DSybil: Optimal Sybil-Resistance for Recommendation Systems

Haifeng Yu*, Chenwei Shi†, Michael Kaminsky‡, Phillip B. Gibbons§ and Feng Xiao¶

*School of Computing, National University of Singapore, Singapore. Email: haifeng@comp.nus.edu.sg
†School of Computing, National University of Singapore, Singapore. Email: shichen@comp.nus.edu.sg
‡Intel Research Pittsburgh, Pittsburgh, USA. Email: michael.e.kaminsky@intel.com
§Intel Research Pittsburgh, Pittsburgh, USA. Email: phillip.b.gibbons@intel.com
¶School of Computing, National University of Singapore, Singapore. Email: xiaof@comp.nus.edu.sg

Abstract

Recommendation systems can be attacked in various ways, and the ultimate attack form is reached with a sybil attack*, where the attacker creates a potentially unlimited number of* sybil identities *to vote. Defending against sybil attacks is often quite challenging, and the nature of recommendation systems makes it even harder.*

This paper presents DSybil*, a novel defense for diminishing the influence of sybil identities in recommendation systems. DSybil provides strong provable guarantees that hold even under the worst-case attack and are optimal. DSybil can defend against an unlimited number of sybil identities over time. DSybil achieves its strong guarantees by i) exploiting the heavy-tail distribution of the typical voting behavior of the honest identities, and ii) carefully identifying whether the system is already getting "enough help" from the (weighted) voters already taken into account or whether more "help" is needed. Our evaluation shows that DSybil would continue to provide high-quality recommendations even when a million-node botnet uses an optimal strategy to launch a sybil attack.*

1. Introduction

Recommendation systems recommend objects to users based on other users' reported prior experience with those objects (i.e., *votes*). There are numerous real world examples for recommendation systems, such as Netflix (for movie rating), Amazon (for book rating), Razor [52] (for collaborative spam filtering), Digg [23], and Credence [60][1]. By casting misleading votes, malicious users or *malicious identities* can potentially mislead a recommendation system. In addition to casting such votes, an adversary seeking to magnify its influence can bribe honest users or even compromise honest users' computers to obtain *bribed identities* or *stolen identities* to cast more votes. The ultimate form is reached with a *sybil attack* [26], where the attacker creates a potentially unlimited number of fake identities (i.e., *sybil identities*) to vote. Off-the-shell software such as Tube Automator [58] and Friend Bomber [31] can readily launch such sybil attacks on recommendation systems.

In this paper, we will use the term sybil identities to refer to all malicious/bribed/stolen/sybil identities. Defending against sybil identities is usually quite challenging, and the nature of recommendation systems makes it even harder. For example, it is known [11, 17, 29] that DHTs can be made secure as long as the fraction of sybil identities is below $1/4$. On the other hand, recommendation systems have significantly lower "tolerance": Because on average only a small fraction (e.g., 1% or 0.1%) of the honest users will vote on an object, even a relatively small number of sybil identities can out-vote these honest voters.

Most existing sybil defense mechanisms [12, 16, 21, 56, 62] cannot provide such strong guarantees. For example, there is evidence [51] that botnet sizes can easily reach 10^5. Even (optimistically) assuming that the sybil defenses can enforce one sybil identity per bot, out-voting the sybil identities created from such a botnet can require a system with 10 to 100 million (i.e., 10^5 divided by 1% or 0.1%) honest users! By leveraging social networks, some recent sybil defenses [44, 57, 62, 63] can nullify the effect of all bots outside the system's social network. But they can still fail to provide guarantees that match recommendation systems' low tolerance threshold (see Section 2).

Trust-based approaches and previous efforts. While directly bounding the fraction of sybil identities is unlikely to be effective for recommendation systems, it is possible to reduce the influence of sybil identities gradually over time based on their historical behavior. For example, a user Alice may trust another identity Bob less if Bob voted for objects that Alice found to be bad. If one can sufficiently diminish the influence of all the sybil identities, then the sybil attack becomes ineffective, for all practical purposes.

Leveraging trust (built over history) in recommendation systems is in no way a new idea. Because trust maps nicely

1. Digg is a popular web site where users "digg" news stories that they consider interesting. A news story with many "diggs" may then be recommended to other users. Similarly, Credence allows p2p file sharing users to vote whether a downloaded file is corrupted. Such votes can then guide other users' downloading.

to human intuition and real life experience, there have been many efforts to leverage trust for defending against malicious users (and occasionally sybil identities). Although simple at the intuitive level, rigorously reasoning about the numerous design alternatives and their end guarantees is always rather tricky.

Because of such difficulty, most previous efforts (e.g., [20, 22, 32, 34, 36, 37, 40, 46, 49, 60]) only explore heuristics without hard guarantees. The proposed techniques are evaluated only against synthetic attacks or attacks observed in measurement traces. It is never clear how robust these systems are against an intelligent adversary launching an optimal attack. We argue that such heuristics can only result in an endless arms race between the attacker and the system designer. On the other hand, there has also been a large body [3, 5, 6, 13, 14, 18, 30, 47, 53, 54] of theoretical work on how to make robust recommendations with provable guarantees. However, to overcome the difficulty in rigorous reasoning about design alternatives and end guarantees, all of them need to make strong restrictive (and sometimes unrealistic) assumptions. For example, some [5, 53, 54] assume that in each round all users vote and that they vote sequentially in some fixed order. In summary, it is still an open question how to deal with a limited number of malicious identities, let alone a potentially unlimited number of sybil identities.

Our goal and approach. This paper aims to answer the following central question:

Is it possible to sufficiently diminish the influence of (a potentially unlimited number of) sybil identities based on historical behavior?

In answering this question, we aim for provable guarantees that hold even under the worst-case attack. To avoid overly restrictive assumptions, we leverage the fact that we do *not* need to optimize for the worst-case voting behavior of the honest identities. After all, the honest identities do not aim to defeat our system. Specifically, we exploit the heavy-tail distribution of the typical voting behavior of the honest identities, and show that this single property can enable a design with strong guarantees.

Our contributions and results. This paper presents *DSybil*, a novel defense for diminishing the influence of sybil identities in recommendation systems. DSybil targets application scenarios such as Digg and Credence where the objects to be recommended are either good or bad. DSybil has the following salient features:

- DSybil can defend against an unlimited number of sybil identities over time.
- The *loss* (i.e., number of bad recommendations) is $O(\mathscr{D}\log M)$ even under the worst-case attack, where M is the maximum number of sybil identities voting on any one object and $\mathscr{D}$ is the *dimension* of the objects (see definition below).
- We prove that the $O(\mathscr{D}\log M)$ loss is optimal. We further show that different from scenarios such as byzantine consensus and DHTs, for a recommendation system to tolerate more sybil identities, the *lifespan* of the honest users is far more important than their *population*.
- DSybil provides a *growing defense*: If the user has used DSybil for some time when the attack starts, the loss will be significantly smaller than the loss under the worst-case attack.

DSybil achieves its strong guarantees partly by optimizing only for cases where $\mathscr{D}$ is small. Roughly speaking, given a set of objects from which we need to make recommendations, the *dimension* $\mathscr{D}$ is the minimum number of honest users that can *cover* a significant fraction (e.g., 60%) of the good objects in the set.[2] Here a user *covers* an object if the user has voted for that object. The value of $\mathscr{D}$ is determined by the voting behavior of the honest users. Our study of large-scale datasets from real-world recommendation systems shows that $\mathscr{D}$ tends to be rather small in practice, due to the heavy-tail vote distribution of the (honest) users. We show that as long as the distribution is heavy-tail, the dimension is likely to be rather small.

Even with small dimension, how to design a robust recommendation system is far from obvious. The central challenge is the tension between i) giving trust to honest identities for their helpful voting, and ii) avoiding giving trust to those sybil identities who are behaving like honest identities (but who could later use the trust to inflict loss). As one would imagine, any design here will necessarily be a double-edged sword: Not giving trust to sybil identities in some cases will unavoidably cause the system not to give trust to some honest identities in other cases. The crux, then, is how to strike the optimal balance between the two factors. In DSybil, we clearly identify whether the user is already getting "enough help" from those identities that the user trusts. If yes, DSybil will not grow the trust to anyone (including honest identities). We are able to show that this design leads to an optimally small loss.

We have implemented DSybil as a toolkit in Java. Our experimental results based on a one year crawl of Digg show that under practical parameters and without taking the growing defense into account, the fraction of bad recommendations by DSybil is only around 12% even when each object has up to 10 billion sybil voters and (on average) only 1,239 honest voters. Taking the growing defense into account, the fraction of bad recommendations further drops to around 6% (or 4%) if the user had been using DSybil for a day (or month) when the attack starts. Assuming that an average honest user's lifespan (of using DSybil) is one year, and assuming that the adversary starts attacking at a random point in time, $364/365 \approx 99.7\%$ (or $11/12 \approx 91.7\%$) of the honest users will have used DSybil for a day (or month)

2. DSybil does not know or try to determine $\mathscr{D}$.

when the attack starts.

With its strong robustness against sybil identities, DSybil needs to use only some basic mechanism to loosely limit M. Specifically, whenever the number of votes on any object exceeds a certain threshold (e.g., 1 billion), DSybil will start imposing a recurring computational puzzle on each voting identity. In our example setting with a recurring 1-minute computational puzzle per week, an adversary would need a million-node botnet to cast the aforementioned 10 billion votes on an object.[3]

2. Related Work

Sybil defenses not leveraging social networks. One way to defend against sybil attacks in recommendation systems is to leverage generic sybil defenses that can bound the number of sybil identities accepted/admitted (which in turn limits the number of sybil voters). The simplest generic sybil defense is for a trusted central authority to verify credentials (e.g., credit card or passport) that are unique to actual human beings. Unfortunately, this often leads to privacy concerns and scares away users. SMS verification also incurs privacy concerns, and does not work well if the attacker has access to cheap phone accounts. Graphical puzzles requiring human efforts (such as CAPTCHAs [59]) can be reposted on the adversary's web site to be solved by clients seeking access to that web site. Charging a fee for each identity can undermine the exact reason why online recommendation systems are popular. Additionally, all the above approaches require a central server, which may not be available (e.g., in Credence). Researchers have also proposed a limited number of decentralized sybil defense mechanisms, including mechanisms based on IP addresses/prefixes [21], resource challenges [16, 56], or network coordinates [12]. All these approaches only provide limited protection and are fundamentally vulnerable to powerful adversaries attacking from botnets.

Sybil defenses leveraging social networks. To better defend against sybil attacks, recent sybil defenses (such as SybilGuard [63], SybilLimit [62], Ostra [44], SumUp [57]) leverage the *social network* among the users. Here an edge connecting an honest user and a malicious user is called an *attack edge*. Because each edge involves human-established trust, it is difficult for the adversary to introduce an excessive number of attack edges. Given g attack edges, if the number of sybil identities behind the attack edges is much larger than g, it will result in a small *quotient cut* in the graph (where the quotient is between g and the number of sybil identities). This graph property translates to poor expansion and large mixing time.

Leveraging this graph property (i.e., expansion and mixing time) can be powerful. If one is willing to assume global knowledge of the continuously-changing social network (i.e., one node maintains an up-to-date copy of the entire social network graph), then simply running an approximation algorithm [42] for minimal quotient cut will bound the number of sybil identities accepted within $O(g \log n)$, where n is the number of honest identities. Also assuming global knowledge and further focusing on scenarios where only $o(n)$ honest identities are seeking to be accepted, SumUp [57] uses adaptive maximum flow on the social network to bound the number of sybil identities (voters) accepted within $g + o(g)$.[4] SybilLimit [62], in contrast, avoids the need for any global knowledge by using a decentralized secure random route technique. It bounds the number of sybil identities accepted within $O(g \log n)$, while provably accepting nearly all honest nodes.

Unfortunately, a lower bound in [62] implies that any social-network-based sybil defense (including all of the systems mentioned above) will be insufficient for recommendation systems, whose tolerance threshold is rather low (e.g., 1% or 0.1%). Specifically, the lower bound proves that exploiting the graph property (i.e., expansion and mixing time) can at the best help us to bound the number of sybil identities accepted within the order of g, and g can easily be larger than the number of honest voters. For example, suppose that each node in the social network has a degree of 10, and optimistically assume that we accept all honest voters and only g sybil voters. Then as long as the adversary can compromise 0.1% (or 0.01%) of the honest identities, the sybil identities can out-vote those 1% (or 0.1%) honest voters. A recent study claims that 40% of the PCs in the U.S. are infected by botnets [1]. While it is quite unlikely for all of them to belong to the same botnet, it is not hard to imagine that many botnets can easily exceed the 0.1% or 0.01% threshold, and out-vote the honest voters.

In contrast, DSybil (which targets recommendation systems only) uses a completely different approach that i) does not require a social network, ii) is not subject to the lower bound on social network defenses, and iii) can in fact tolerate a million-node botnet given just thousands of honest voters.

Sybil attacks in reputation systems. Related to recommendation systems, *reputation systems* (such as Ebay) are also vulnerable to sybil attacks. A reputation system computes the numerical reputation of individual identities, based on pair-wise feedback among the identities. A number of mechanisms [19, 28, 35, 55] have been proposed to prevent sybil identities from artificially boosting the attacker's reputation. Compared to DSybil, these efforts do not explore how to use such reputation to make recommendations (for

3. With a million-node botnet, the adversary may be able to launch serious DDoS attacks on various components of the system. How to defend [25] against such DDoS attacks is beyond the scope of this paper.

4. In addition, as a secondary optimization, SumUp leverages feedback and trust, but its heuristics are without provable guarantees on end-to-end loss.

objects) with provable guarantees.

Classic machine learning algorithms and extensions. From a theoretical perspective, DSybil's recommendation algorithm provides strong guarantees for a variant of the sleeping-experts-based [38] adversarial multiarmed bandit (MAB) problem [3] in machine learning. In this problem, the algorithm needs to make a recommendation out of a pool of objects in each round, based on the votes from the "experts." *Sleeping-experts* means that an expert might not participate in all rounds, while *MAB* means that the algorithm obtains only partial-feedback [18] (i.e., no feedback is provided for objects not chosen/recommended). These two aspects of the problem make it particularly challenging.

There are existing results [3] on the MAB problem that assume all experts participate (vote) in all rounds. This assumption rarely holds when the experts are real users in a recommendation system. Researchers [5, 6, 53, 54] have also applied MAB-related techniques to recommendation systems, while inheriting the previous assumption (or making even stronger assumptions). Some efforts [13, 14, 30, 47] have investigated how to make recommendations based on votes from sleeping-experts who do not always participate. These efforts all assume complete feedback where the "goodness" of all objects (even non-recommended ones) are revealed after each round. For our applications, doing so would require the user to test all objects, which defeats the exact purpose of providing recommendations.

To the best of our knowledge, the only work that discusses the sleeping-experts-based MAB problem is [38]. The proposed algorithm has exponential complexity and linear loss (in terms of the number of sybil identities). We are able to design a linear-complexity logarithmic-loss algorithm for this problem by exploiting the voting pattern of honest users.

Other theoretical work on recommendation systems. There have been many other theoretical efforts on recommendation systems [2, 4, 7, 8, 9, 10, 27, 39]. They largely deal with finding good objects out of a fixed set of objects, while our model involves multiple rounds where each round has its own set of objects. The notion of trust (across multiple rounds) only becomes relevant in our model. Near the end of [8], Awerbuch et al. discuss an algorithm for multiple rounds. The loss, however, is linear with the number of sybil identities.

3. System Model and Attack Model

Target applications/scenarios. Recommendation systems are a generic concept, and the details of different recommendation systems can be dramatically different. As a result, solutions suitable for one scenario (e.g., Netflix) may very well be inappropriate in other contexts (e.g., Digg). DSybil does not intend to capture all possible recommendation systems. Rather, we focus on scenarios where i) the objects to be recommended are either good or bad (but different users may have different subjective opinions regarding whether an object is good); ii) the lifespan of the users is not overly short so that they can build trust; and iii) the users aim to find some good objects to consume instead of exhaustively finding all good objects.

In terms of real world applications, DSybil captures the requirements of p2p file rating systems such as Credence [60] and news story recommendation systems such as Digg [23]. In both cases, the objects are binary in terms of "goodness." A typical user of Credence or Digg may download many files or read many news stories (e.g., 20 per day). Also, a user in Credence is usually only interested in downloading one good (i.e., non-corrupted) version of, for example, a particular song. The user does not aim to exhaustively find all good versions. Similarly, a typical Digg user may wish to read a certain number of interesting news stories on a given day, without finding out all the interesting stories appearing in that day. On the other hand, DSybil cannot yet capture Netflix or Amazon where rather fine-grained (e.g., 5-star scales) recommendations are expected.

In the remainder of this section, we formally define DSybil's system model and attack model.

Objects and rounds. DSybil recommends *objects* (e.g., news stories in Digg or files for downloading in Credence) to users, based on *votes* (see later) from other users. Depending on a user's prior behavior, DSybil may recommend different objects to different users. From now on, our discussion will always be with respect to a given DSybil user (called Alice). We assume that Alice follows the protocol honestly.

With respect to Alice, each object is either *good* or *bad*. In each *round*, Alice would like to pick one object out of a specific set (U) of objects to *consume* (e.g., read the story or download the file). For example, U may be the set of all stories submitted over the past day in Digg or all the different versions of mp3 files with the same song title in Credence. We assume that the fraction of good objects in U is at least p that is bounded away from 0. For example, measurement studies [43] of the file pollution level in Kazaa show that the fraction of good versions of the songs studied ranges between 30% to 100%, under attackers with commercial interests. After each round, Alice provides *feedback* to DSybil regarding whether the consumed object is good. This feedback can be in the form of a vote but does *not* have to be. For example, Alice may provide feedback but choose not to cast a vote that other users will see (e.g., due to the sensitivity of the object). Sometimes implicit feedback is possible (e.g., based on how much time Alice spends reading the news story).

DSybil can be configured to recommend either a single object or a set of k (e.g., 5 or 10) objects in each round. If multiple objects are recommended, Alice picks one of them to consume. As an example, one can view the set as a web page displaying excerpts of k recommended news stories

in Digg. Also, if Alice wants to consume multiple objects from a given set U, we can always model that as multiple rounds with the same U (after removing the objects already consumed).

We use *loss*, defined as the total number of bad objects consumed by Alice, to measure the "goodness" of the recommendation system. Since DSybil is randomized, we will only be concerned with the expectation on loss, where the expectation is taken over the random coin flips in the algorithm. The loss is also dependent on the attack strategy of the adversary. When we say *loss under the worst-case attack*, we mean the expected loss under the worst-case attack strategy. We expect that in order to retain users, a recommendation system needs to achieve a rather small per-round loss (e.g., 0.1 or 0.05) that can be much smaller than $1-p$. To achieve this, DSybil leverages the votes from other users.

Votes. A *vote* claims a certain object to be good, and this is the only kind of vote DSybil uses. In this sense, all votes in DSybil are "positive votes". We will later prove that in our setting "negative votes" can never help to reduce loss, and thus DSybil does not use them. A user does not have to use DSybil (or follow DSybil's recommendations) in order to vote. For example, a Linux fan may read all Linux-related news stories on Digg and then vote for the interesting ones. Similarly, a round as defined earlier is not a voting round. Namely, the votes on the objects in Alice's round were cast (by other DSybil and non-DSybil users) independent of when Alice starts the round.

Attack model. Because we consider sybil attacks, we will use the term *identities* instead of users from now on. An identity is not necessarily tied to any human being or computer in the real world. An identity is either *honest* or *sybil*. A sybil identity can be an identity owned by a malicious user, or it can be a bribed/stolen identity, or it can be a fake identity obtained through a sybil attack. We assume that DSybil can associate each vote with the identity casting that vote. For example, this can be achieved by requiring an identity to log into his/her account before voting. Alternatively, each identity may have a *locally* generated public/private key pair, and signs each vote using the private key. (The adversary can freely generate an unlimited number of accounts or key pairs for the sybil identities.) Each identity may cast at most one vote on any given object; otherwise, its votes will be ignored.

A sybil identity is byzantine and may collude with all other sybil identities. We allow the sybil identities to know which objects are good/bad and also the votes cast by honest identities. We do not make assumptions on the total number of sybil identities over time, which can be infinite. We assume that the number of sybil identities voting on any given object is at most M, where M (e.g., 10^{10}) can be orders of magnitude larger than the number of voting honest identities (e.g., 1000).

N	# of guides
N'	# of non-guides
M	maximum # of sybil voters on any object
W	maximum # of honest voters on any object
p	each round has at least p fraction of good objects
$\mathscr{D}_f$	the f-fractional dimension
f	the fraction of good objects covered by critical guides

Table 1. Key notations in this paper. DSybil does not know or need to know the values of any of these parameters.

A round is called *attack-free* if there are no sybil identities voting in that round. DSybil does not know which rounds (if any) are attack-free.

Honest identities may have different "tastes" and thus may have different (subjective) opinions on the same object. We assume that there are N honest identities (called *guides*) with the *same* or *similar* "taste" as Alice. Specifically, having the *same taste* as Alice means that the guide never votes on bad objects. (On the other hand, the guide may or may not vote on any given good object.) Having *similar taste* as Alice means that the guide seldom votes on bad objects. The existence of such guides is a necessary assumption for virtually all recommendation systems—if Alice has such an esoteric taste that no one else shares her taste, no recommendation system can help Alice. Let N' denote the number of honest identities that are not guides (called *non-guides*). We will focus on cases where $M > N+N'$. Unless otherwise mentioned, we will pessimistically treat the N' non-guides as byzantine (and potentially colluding with sybil identities). DSybil does not know or try to determine which identities are guides, non-guides, or sybil. In particular, the extra loss due to a guide being bribed or compromised is equivalent to M increasing by 1 (unless removing the guide increases the dimension—see later).

We say that an object is *covered* by an identity if that identity votes for that object. Let $\mathscr{U}$ denote the set of objects that has ever appeared in any of Alice's rounds. Given the votes on the objects in $\mathscr{U}$, we defined $\mathscr{U}$'s *f-fractional dimension* (denoted as $\mathscr{D}_f$) as the smallest number of guides that can cover f fraction of the *good* objects in $\mathscr{U}$. We will be concerned only with f that is not too small (e.g., 0.5). We call these $\mathscr{D}_f$ guides as *critical guides*.[5] We call those objects that are covered by at least one critical guide as *guided*. Other objects (even if they are covered by some other guide) are called *unguided*. For any given set of identities, their *voting pattern* fully specifies which identities vote for which objects. By definition, $\mathscr{D}_f$ is only affected by the voting pattern of the guides, which are honest. Calculating $\mathscr{D}_f$ can be reduced to Set-Covering and is thus NP-hard. DSybil does not know which objects are guided or which identities are critical guides. DSybil does not know/calculate $\mathscr{D}_f$ either. Table 1 summarizes the key notations in this paper.

5. It is possible to have different sets of $\mathscr{D}_f$ guides where every set can cover f fraction of the good objects in $\mathscr{U}$. In such case, we simply pick an arbitrary set and call (only) those guides in that set as *critical guides*.

4. Leveraging Trust: The Obvious, the Subtle, and the Challenge

Even with small $\mathscr{D}_f$ values, designing a robust recommendation system is far from trivial. This section discusses some of the issues.

The obvious. The notion of trust appeals naturally to human intuition. For example, we all know that Alice should trust Bob more (less) if Alice finds that Bob voted for good (bad) objects. Also, objects with more votes should probably be recommended over objects with fewer votes.

The subtle. How to implement this "obvious" notion is in fact not so obvious, however, as one starts asking quantitative and more specific questions. First, how should we assign seed trust (i.e., initial trust) to new identities? Because the adversary can continuously introduce new identities (by *whitewashing*), giving positive seed trust to each new identity may already allow the adversary to inflict infinite loss over time. One might introduce some "trial period" so that an identity is given seed trust only after, for example, voting for 10 good objects. Unfortunately, knowing this, a sybil identity could always first vote for 10 good objects and then immediately cheat. Notice that these first 10 "correct" votes would not benefit Alice at all, since identities in the trial period are not used for recommendations.

Second, how exactly should we grow the trust of the identities? If Bob has cast 4 "correct" votes while Cary has cast 12, should the trust to Cary be 3 ($= 12/4$) times or 256 ($= 2^{(12-4)}$) times the trust to Bob? In particular, should we (and how can we) avoid growing the trust of identities casting "correct" but "non-helpful" votes? For example, a sybil identity may vote for a good object that already has many votes from other users. Such a "correct" vote is "non-helpful" because the object will most likely be selected anyway even without this additional vote. To prevent sybil identities from gaining trust "for free" by casting such votes, one may try to determine the "amount of contribution" from different voters. Doing so, however, can be rather tricky. For example, some researchers [53, 54] propose taking the voting order into account. The first voter for the object gains the most trust, while later ones gain less and less trust. However, if the adversary knows that a certain object will attract a lot of votes by the time that Alice selects, the sybil identities can all rush and cast the first batch of votes on that object. These votes are still "non-helpful" because the object will attract enough votes anyway.

Finally, how should we recommend objects based on the (potentially conflicting) votes from the various identities with different trust? Should two votes from two identities each with x trust be considered equivalent to one vote from an identity with $2x$ trust? Should we deterministically choose the object with the most votes, or use randomized recommendation where the probability is proportional to the number of votes? Should we take negative votes into account?

The challenge. The above numerous (and subtle) design alternatives are not simply different "trade-offs": Because the end user Alice has only a single objective (minimizing loss), different designs indeed have different "goodness." However, it is simply impossible to exhaustively enumerate and compare them all. Thus we aim to directly design a defense with optimal loss. Doing so will answer all the design questions once and for all. This is also a key difference between our approach and previous trust-based recommendation systems [20, 32, 34, 36, 40, 46, 49], where it is always unclear whether there exist better designs. The key challenge now, of course, is to design a defense whose loss is optimal. In the rest of this paper, we will not extensively explore other design alternatives, though one should keep in mind that we will prove that no alternatives are (asymptotically) better.

5. DSybil Recommendation Algorithm

The main component of DSybil's defense is its optimal and elegant recommendation algorithm. The algorithm can be run either by individual users (independent of whether other users are using DSybil), or by a central server making recommendations to individual users (where the server will run different instances for different users). We will later show that some natural alternatives (or "optimizations"/"tweaks") to our algorithm will easily break its optimality. This implies that achieving the optimality is far from trivial—many of the design choices in our algorithm are not arbitrary and in fact, may be necessary for optimality.

We describe the algorithm below in a progressive fashion. Section 5.1 first discusses an algorithm that assumes i) the total number of sybil identities is M; ii) the values of M, N', and $\mathscr{D}_f$ are known; iii) DSybil recommends, and Alice consumes, only one object from each round; and iv) Alice has the same taste as the critical guides. Next, Section 5.2 improves the algorithm so that it can tolerate an unlimited number of sybil identities, as long as the number of sybil voters on any given object is bounded within M. Furthermore, the improved algorithm no longer needs to know M, N', or $\mathscr{D}_f$. Finally, Section 5.3 shows that the remaining assumptions can be naturally relaxed as well. For simplicity, our proofs will assume that if the f-fractional dimension of $\mathscr{U}$ is $\mathscr{D}_f$, then in every round, f fraction of the good objects are guided (i.e., covered by at least one of the $\mathscr{D}_f$ critical guides). It is not difficult to extend our results to weaker but less concise conditions.

5.1. At Most M Sybil Identities Total

Algorithm description. Our recommendation algorithm (Figure 2) maintains a real-valued *trust*, initialized to some

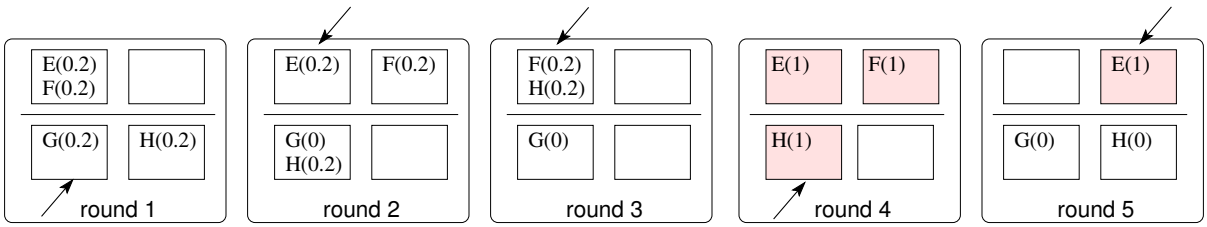

Figure 1. An example execution of the algorithm in Figure 2 with $S = 0.2$, $c = 1$, $\alpha = 5$, and $\beta = 0$. The top two objects in each round are good, and the bottom two are bad. There are 2 honest identities E and F, and 2 sybil identities G and H. The figure annotates which identities vote for which objects, and the trust of those identities. Shaded objects are overwhelming. Objects pointed to by arrows are recommended objects, consumed by Alice. A random object is recommended in rounds 1–3. An arbitrary overwhelming object is recommended in rounds 4 and 5.

Make recommendation:
determine which objects are overwhelming;
return an arbitrary overwhelming object if there is one;
otherwise return a uniformly random non-overwhelming obj;

After Alice's feedback on the recommended object O:
if O is good *and* if O is non-overwhelming
  multiply the trust of all voters for O by α;
if O is bad
  multiply the trust of all voters for O by β;

Figure 2. Recommendation algorithm executed in each round. Each identity starts with S trust. The parameters satisfy $0 < S < c$, $\alpha > 1$, and $0 \leq \beta < 1$. See Corollary 2 for possible values.

seed value $S > 0$, for each identity. The algorithm is with respect to Alice—different Alices will run different instances of the algorithm and thus have different trust for the same identity. An object is *overwhelming* if the total trust of the identities voting for the object is at least some constant c, where $c > S$. If a round has overwhelming objects (called an *overwhelming round*), the algorithm simply recommends an arbitrary overwhelming object; otherwise it recommends a uniformly random non-overwhelming object. Here "arbitrary" means that the overwhelming object can be chosen in some arbitrary way (even adversarially).[6] Notice that our (optimal) algorithm does not further distinguish different overwhelming objects or different non-overwhelming objects.

Alice then consumes the object and provides feedback. If the object is good *and* if the object is non-overwhelming, the algorithm multiplies the trust of all identities voting for that object by some constant $\alpha > 1$. If the object is bad, the algorithm multiplies the trust by some constant β where $0 \leq \beta < 1$. A nonzero β serves to allow guides with similar but non-identical taste as Alice (Section 5.3). Table 2 summarizes DSybil's parameters.

Algorithm intuition. Figure 1 illustrates an example execution of the algorithm. Initially when all the identities have small trust, the algorithm recommends random objects in each round. Because trusts grow exponentially whenever Alice consumes a good non-overwhelming object, overwhelming rounds will quickly arise. Overwhelming objects can always be consumed "safely"—even if the object turns out to be bad, the algorithm will benefit because it confiscates a substantial amount of trust from the sybil identities. Overwhelming rounds may still be followed by non-overwhelming rounds, when for example, some identities with large trust that were actively voting are no longer voting. Regardless of the mixture of non-overwhelming and overwhelming rounds, conditioned upon the dimension of the objects being small, we will be able to later prove that the number of non-overwhelming rounds is small.

c	the threshold for an object to be overwhelming
α	the multiplicative factor when increasing trust
β	the multiplicative factor when decreasing trust
S	seed trust of each identity (only used in the simplified algorithm in Figure 2)
S_Σ	total seed trust, if any, given out in each round (only used in the full algorithm in Figure 3)

Table 2. The (only) parameters in DSybil's algorithm.

Our algorithm does not increase the trust of any identity when a good overwhelming object is consumed. Intuitively in such cases, Alice is already getting "enough help" from the voters and thus does not need to increase any one's trust. This addresses a key issue raised in the previous section: Namely, under such design, sybil identities casting "correct" but "non-helpful" votes will never be rewarded with additional trust. On the other hand, such design is unavoidably a double-edged sword, and can prevent us from increasing the trust to guides. In particular, it makes the "choking attack" possible, where the sybil identities can "choke" honest identities so that the honest identities can never gain trust. One key challenge addressed by the algorithm is how to strike the optimal balance among these factors.

Guarantees against the worst-case attack. To facilitate our formal arguments, we define λ to be the number of guided good non-overwhelming objects consumed by Alice. We use G_n and G_o to denote the number of good

6. This leads to an interesting effect—in overwhelming rounds, one can lay another (arbitrary) recommendation algorithm on top of our algorithm, and yet the combined algorithm is guaranteed to remain optimal.

non-overwhelming objects and good overwhelming objects consumed by Alice, respectively. We similarly define B_n and B_o for bad non-overwhelming objects and bad overwhelming objects, respectively. We will show that λ is properly bounded, which in turn bounds G_n, B_n, B_o, and eventually Alice's loss (i.e., $B_n + B_o$).

Consider a guided good non-overwhelming object consumed by Alice. After the object is consumed, our algorithm will increase the trust of at least one critical guide. Furthermore, no identity can ever have a trust value larger than $\alpha \cdot c$. Otherwise the identity would have a trust value of at least c before the last trust increase, which is impossible. Because every identity starts with a trust of S, we can multiply the trust of a critical guide by α for at most $\lceil \log_\alpha(\alpha c/S) \rceil$ times before the trust reaches $\alpha \cdot c$. Thus, Alice can consume at most $\mathscr{D}_f \lceil \log_\alpha(\alpha c/S) \rceil$ guided good non-overwhelming objects and $\lambda \le \mathscr{D}_f \lceil \log_\alpha(\alpha c/S) \rceil$. This means that with small $\mathscr{D}_f$, λ will be small as well.

Whenever Alice consumes a guided good non-overwhelming object, the object is a uniformly random object from all the (non-overwhelming) objects in that non-overwhelming round. Since the fraction of guided good objects in a round is at least $p \cdot f$, we can view λ as the number of successes when repeating an experiment of at least pf success probability for $B_n + G_n$ times. According to the mean of the geometric distribution, we have $E[B_n + G_n] \le \frac{1}{pf}\lambda$. By a similar argument, we also have $E[G_n] = \frac{1}{f}\lambda$.

Finally, we want to reason about B_o. Sybil identities can only increase their trust by voting for good non-overwhelming objects. Each such object enables the adversary to gain less than $c\alpha - c$ additional trust. On the other hand, whenever Alice consumes a bad overwhelming object, the trust of the voters for that object will be multiplied by β. Thus, the algorithm confiscates at least $c - c\beta$ trust from the sybil identities (which includes the non-guides). Because the total confiscated trust will never be larger than the trust that the sybil identities can possibly gain, we have $c \cdot (1-\beta) \cdot B_o \le c \cdot (\alpha - 1) \cdot G_n + (M + N') \cdot S$. Alice's loss (i.e., $B_n + B_o$) can then be shown to be properly bounded:

Theorem 1: Let L be the loss of the algorithm in Figure 2, against an adversary attacking with M sybil identities total. Then regardless of the adversary's strategy:

$$\begin{cases} \lambda \le \mathscr{D}_f \lceil \log_\alpha(\alpha \cdot \frac{c}{S}) \rceil \\ E[L] \le \lambda \cdot \left(\frac{\alpha - 1}{f(1-\beta)} + \frac{1-p}{pf} \right) + \frac{M+N'}{1-\beta} \cdot \frac{S}{c} \end{cases} \quad (1)$$

Proof sketch: Our earlier discussion already shows that $\lambda \le \mathscr{D}_f \lceil \log_\alpha(\alpha c/S) \rceil$. Additionally, we have $L = B_n + B_o$ and :

$$\begin{cases} E[G_n] = \frac{1}{f}\lambda \\ E[B_n + G_n] \le \frac{1}{pf}\lambda \\ (1-\beta) \cdot c \cdot B_o \le (\alpha - 1) \cdot c \cdot G_n + (M + N') \cdot S \end{cases}$$

After Alice's feedback on the recommended object O:
if O is good *and* if O is non-overwhelming
  for each of the x voters for O with zero trust (if any), set their trust to S_Σ/x;
  for each voter for O, multiply its trust by α;
if O is bad
  multiply the trust of all voters for O by β;

Figure 3. Recommendation algorithm executed in each round. Each identity starts with 0 trust. The parameters satisfy $S_\Sigma > 0$, $\alpha > 1$, and $0 \le \beta < 1$. See Corollary 4 for possible values.

Solving the above equations yields the desired results. □

The above theorem guarantees Alice's expected loss, where the expectation is taken over the random coin flips in the algorithm. It is also possible to prove a high probability guarantee on the loss, via standard Chernoff bounds [45] on G_n and $(B_n + G_n)$. We omit the details for space limitations.

We will take $\beta = 0.5$ to accommodate guides with non-exact taste as Alice (Section 5.3). It is possible to find the optimal values for α, S, c. For simplicity, however, we use $\alpha = 2$ and $S/c = \mathscr{D}_f/(2M + 2N')$, which is sufficient to achieve the best result asymptotically. Notice that here we need to know N', M, and $\mathscr{D}_f$ in order to properly set S/c.

Corollary 2: Setting $\alpha = 2$, $\beta = 0.5$, and $S/c = \mathscr{D}_f/(2M + 2N')$, we have

$$E[L] \quad \le \quad \mathscr{D}_f + \mathscr{D}_f \frac{1+p}{pf} \left\lceil \log_2 \left(\frac{4(M+N')}{\mathscr{D}_f} \right) \right\rceil \quad (2)$$

For p and f constants bounded away from 0, this becomes $O\left(\mathscr{D}_f \log(M/\mathscr{D}_f)\right)$ asymptotically.

5.2. At Most M Sybil Voters on Each Object

Algorithm description. Figure 3 presents the improved algorithm that can tolerate an unlimited number of sybil identities, as long as the number of sybil identities voting for any given object is at most M. The part for making recommendations is the same as in Figure 2 and thus is not shown. Different from earlier, here each identity starts with 0 trust. If Alice consumes a good non-overwhelming object, then if there are x voters with 0 trust voting for that object,[7] each such voter will be given a seed trust of S_Σ/x. Here S_Σ is some positive constant. Notice that our optimal algorithm does not use any "trial period" for new identities.

Algorithm intuition. Having identities start with 0 trust is critical for dealing with an unlimited number of sybil identities (over time), because otherwise even the seed trust assigned to sybil identities will grow unbounded. The intuition for giving out seed trust only when Alice consumes a good non-overwhelming object is similar as before. Namely,

7. When $\beta = 0$, a voter with 0 trust may either be an uninitialized voter, or can be a voter that voted on a bad object. We do not need to distinguish the two cases.

if the object is overwhelming, then Alice is already getting "enough help." Same as before, such design is unavoidably a double-edged sword, and can prevent guides from obtaining seed trust.

We will still be able to show that the number of good non-overwhelming objects consumed (i.e., G_n) is limited. Thus, the total seed trust given out to sybil identities will also be limited. Instead of giving each identity the same seed trust, the algorithm simply enforces a limit S_Σ on the total seed trust given out each time. How exactly to distribute S_Σ to the various voters for the object can be rather flexible. All we need to ensure is that each critical guide receives some seed trust that is not too small (without knowing who they are). Our algorithm in Figure 3 distributes S_Σ evenly to all voters with 0 trust, which ensures that the seed trust of a critical guide is at least $S_\Sigma/(W+M)$. Here $W \le N+N'$ is the maximum number of honest users voting on any object. Notice that by doing so, the algorithm no longer needs to know the values of N, N', M, or $\mathscr{D}_f$.

Guarantees against the worst-case attack. The guarantees of the algorithm in Figure 3 can be proved using similar arguments as before (a high probability result can also be obtained):

Theorem 3: Let L be the loss of the algorithm in Figure 3, against an adversary attacking with at most M sybil identities voting for any given object. Then regardless of the adversary's strategy:

$$\begin{cases} \lambda \le \mathscr{D}_f \lceil \log_\alpha(\alpha \cdot c \cdot (W+M)/S_\Sigma) \rceil \\ E[L] \le \lambda \cdot \frac{1}{pf} \cdot \left(p \cdot \frac{S_\Sigma/c+\alpha-1}{1-\beta} + 1 - p \right) \end{cases} \tag{3}$$

Proof sketch: By similar arguments as in Theorem 1, we have:

$$\begin{cases} \lambda \le \mathscr{D}_f \lceil \log_\alpha(\alpha \cdot c \cdot (W+M)/S_\Sigma) \rceil \\ E[G_n] = \frac{1}{f}\lambda \\ E[B_n+G_n] \le \frac{1}{pf}\lambda \\ (1-\beta)\cdot c \cdot B_o \le (S_\Sigma + (\alpha-1)\cdot c)\cdot G_n \end{cases}$$

Solving the above equations yields the desired results. □

It is possible to find the optimal values for α and S_Σ/c, but that will make them dependent on p and M (which are unknown). Thus, we simply use $\alpha = 2$ and $S_\Sigma/c = 0.5$. These values are sufficient to give us the best result asymptotically.

Corollary 4: Setting $\alpha = 2$, $\beta = 0.5$, and $S_\Sigma/c = 0.5$, we have

$$E[L] \le (1+2p)/(pf)\cdot\lambda_0, \tag{4}$$

$$\text{where } \lambda_0 = \mathscr{D}_f \cdot \lceil \log_2(4(W+M)) \rceil \tag{5}$$

Because $W \le N+N' \le M$, for p and f constants bounded away from 0, $E[L]$ becomes $O\left(\mathscr{D}_f \log M\right)$ asymptotically.

A growing defense. A salient property of DSybil is that its defense against attacks grows over time. Namely, if there are some initial attack-free rounds, then Alice's total loss (including the loss in the attack-free rounds) can be significantly smaller than the loss from Corollary 4.[8] For example, in later experiments under $M = 10^{10}$, Alice's per-round loss drops from 12% to 4% if Alice has been using DSybil for a month before the adversary starts attacking.

This growing defense comes from the following three factors. First, to inflict a maximum loss, the adversary needs to ensure that each critical guide receives no more than $S_\Sigma/(W+M)$ seed trust. In an attack-free round, the number of voters for any given object is likely to be significantly smaller than $W+M$. Thus, in those rounds the seed trust given to a critical guide can be much larger than $S_\Sigma/(W+M)$. Such effect can be significant because the identities are likely to be assigned seed trust during the earlier rounds (instead of later rounds). Second, to inflict a maximum loss, the sybil identities need to vote for every good non-overwhelming object consumed (without causing the objects to become overwhelming), so that they can gain trust and later maximize B_o. If Alice has already consumed some good non-overwhelming objects before the attack starts, the sybil identities lose some "opportunities" to gain trust.

Finally, the maximum loss in Corollary 4 is reached only when for every guided good non-overwhelming object consumed, there is only one critical guide voting for that object. As one would imagine, in practice, there can easily be multiple critical guides voting. Whenever this happens in an attack-free round, the trust of multiple critical guides will increase, and λ will be smaller. After the attack starts, the adversary will be able to use a *choking attack* to always prevent this from happening. Namely, whenever multiple critical guides vote for a good non-overwhelming object, the sybil identities can all vote for that object and make it overwhelming. This "chokes" the critical guides so that they will not have the opportunity to simultaneously gain trust.

One can easily modify Corollary 4 to calculate a bound on the loss when the attack does not start from the very first round. Specifically, let s_i be the trust of the ith critical guide when the attack starts (for $1 \le i \le \mathscr{D}_f$). To bound the loss incurred after the attack starts, we simply replace Equation 5 with:

$$\lambda_0 = \sum_{i=1}^{\mathscr{D}_f} \left\lceil \log_2 \left(\frac{2}{\max(s_i/c, 1/(2W+2M))} \right) \right\rceil \tag{6}$$

Adding such loss to the loss incurred during the attack-free rounds will then give us the total loss.

5.3. Further Extensions

Consuming multiple objects in a round. Our discussion so far assumes that DSybil recommends one object in each round and Alice consumes the recommended object. In some cases, Alice may want to consume more than one object. For example, Alice may want to read 20 new stories among

8. Recall that DSybil does not know which rounds are attack-free.

the set U of news stories in the past day. As mentioned in Section 3, we can readily model this as 20 rounds. Let the set of the objects in these 20 rounds be U_1 through U_{20}. Then we can set $U_1 = U$, $U_2 = U \setminus \{O\}$ where O is the object already consumed in round 1, and so on.

The only complication here is that if O is a guided good object, then the fraction of good objects and guided good objects in U_2 may now be below p and pf, respectively. To be precise, let v be the number of objects Alice consumes from the original set U and imagine that we model the process as v rounds where Alice consumes one object per round. Let p' and f' be the lower bound on the fraction of good objects and guided good objects in any of these v rounds. Let $u = |U|$. A straight-forward calculation then shows:

$$p' = \frac{u \cdot p - v}{u - v} \quad \text{and} \quad f' = \frac{u \cdot p \cdot f - v}{u \cdot p - v} \tag{7}$$

Applying Corollary 4 then immediately shows

$$E[L] \quad \leq \quad (1+2p')/(p'f') \cdot \lambda_0 \tag{8}$$

Notice that these results assume that $u \cdot p \cdot f > v$, which explains why f cannot be overly small.

Recommending multiple objects in a round. Our algorithm can also be easily extended to recommend k ($k > 1$) objects in each round, among which Alice can select one to consume (as mentioned in Section 3). Alice may have different ways to pick the object out of the k objects recommended. We assume that her choice is at least as good as picking a uniformly random object out of the k objects. This is rather reasonable since otherwise Alice should just toss a coin each time. Thus it suffices to consider the worst case where Alice just picks a uniformly random object to consume.

To recommend k objects in a round with at least k overwhelming objects, we can simply recommend an arbitrary k overwhelming objects. If a round has no overwhelming objects, we will recommend k uniformly random (non-overwhelming) objects. The boundary condition where a round has k' ($1 \leq k' < k$) overwhelming objects is slightly tricky. The simplest method is to recommend only the k' overwhelming objects. Doing so will preserve all the guarantees from Section 5.1 and 5.2 without any change. The drawback is, of course, that the algorithm may occasionally recommend $k' < k$ objects in a round.

It is possible to remove this minor limitation (if necessary), by recommending all k' overwhelming objects together with $k - k'$ uniformly random non-overwhelming objects. Doing so, however, will break the proofs in Section 5.1 and 5.2 for the following reason. Each round has at least a $p \cdot f$ fraction of guided good objects. Our earlier proofs leverage the fact that if the round has no overwhelming objects, then a uniformly random non-overwhelming object is a guided good non-overwhelming object with probability of at least pf. Now because the round has k' overwhelming objects (which can very well be guided good objects), the fraction of guided good objects among the non-overwhelming objects can be smaller than pf. However, we expect such effect to be negligible because $|U|$ is usually much larger (e.g., on the order of tens or hundreds in Digg) than k and k' (e.g., 5).

Disagreements between Alice and the critical guides. DSybil targets contexts where users can be classified into types and each type has the same or similar "taste." Our analysis so far has been assuming that Alice has the same taste as the critical guides (i.e., the critical guides never vote on bad objects). (Whether Alice has the same or similar taste as the other guides is irrelevant.) Next, we show that a small number of votes from the critical guides on bad objects will not increase the loss of our algorithm excessively, as long as β is not too close to 0.

Imagine that a critical guide votes on a bad object O and Alice consumes O. This will increase Alice's loss in two ways: i) if not for the vote from the critical guide, Alice might not consume this bad object, and ii) the trust of the critical guide will be decreased (i.e., multiplied by β). The extra loss from the first part is at most 1. For the second part, with $\alpha = 2$ and $\beta = 0.5$, DSybil only needs to multiply the trust of the critical guide by α one additional time in order to compensate. Thus, the effect is simply to increase λ_0 by 1. In turn, this translates to an extra loss of $O(1)$ (by Corollary 4).

6. A Deeper Look

DSybil's algorithm is optimal. We will prove that DSybil's elegant recommendation algorithm is (asymptotically) optimal, by proving a lower bound on loss. To make our lower bound as strong as possible, we will allow "negative votes" that were not included in our system model. We allow sybil identities to cast negative votes in arbitrary ways. For guides, we optimistically assume that if a guide ever appears (casts positive votes) in a round, it will cast negative votes on all the bad objects in that round. Obviously, this maximizes the information regarding which objects are bad and makes our lower bound stronger. Our lower bound construction depends on the value of p. Although our algorithm focuses on constant p within $(0,1)$, to be as complete as possible, we will also consider $p \to 0$ and $p \to 1$ when proving the lower bound. Doing so will help to reveal that the $1/p$ factor in DSybil's loss is somewhat fundamental. For the lower bound, we continue to assume that if the f-fractional dimension is $\mathscr{D}_f$, then in every round, f fraction of the good objects need to be guided. Notice that this strengthens the lower bound, because it imposes additional restrictions on our construction. The following theorem presents the lower bound (see the appendix for proof).

Theorem 5: For any given p (in the following 3 cases), nonnegative integer M, positive integer $\mathscr{D}$, and any recommendation algorithm, we can always construct a sequence

of rounds, objects, and votes where i) the fraction of good objects in each round is at least p, ii) there are at most M sybil identities voting for each object, iii) for all $f \in (0,1]$ the f-fractional dimension, $\mathscr{D}_f$, of the objects is $\mathscr{D}$, and iv) the algorithm will incur at least

- $\frac{1}{2} \cdot \mathscr{D}_f \cdot \lfloor \log_2 M \rfloor$ expected loss if $p = 0.5$.
- $\frac{1}{4} \cdot \mathscr{D}_f \cdot \left\lfloor \frac{1}{p} \right\rfloor \cdot \left\lfloor \frac{\log_2 M}{\log_2(1/p)} \right\rfloor$ expected loss if $0 < p < 0.5$.
- $\frac{1}{2} \cdot \mathscr{D}_f \cdot \left\lfloor \frac{\log_2 M}{\log_2(3/(1-p))} \right\rfloor$ expected loss if $0.5 < p < 1$.

For constant p, this becomes $\Omega(\mathscr{D}_f \log M)$.

This lower bound (asymptotically) matches DSybil's guarantee in Corollary 4. In addition, it is worth noting that for $0 < p < 0.5$, the $1/p$ term is present in the lower bound as well (except for an extra logarithmic term $\log_2(1/p)$ in the denominator, which however is dominated by $1/p$). This means that the $1/p$ term in DSybil's loss (Corollary 4) cannot be avoided, unless we make additional assumptions in the model.

For the scenario where the total number of sybil identities is M (as in Section 5.1 and Corollary 2), using a rather similar proof as for Theorem 5, we can prove that the lower bound there is the same as in Theorem 5 except that the term "$\log_2 M$" should be replaced by "$\log_2(M/\mathscr{D}_f)$". Again, this asymptotic lower bound $\Omega(\mathscr{D}_f \log(M/\mathscr{D}_f))$ matches the guarantee from Corollary 2.

DSybil's algorithm may be necessary for optimality. We next show that some natural alternatives (or "optimizations"/"tweaks") to our algorithm will easily break its optimality. This means that many of the design choices in DSybil may actually be necessary for optimality. Some of the following alternative designs (or similar designs) have been used in previous recommendation algorithms (which are not necessarily designed to defend against sybil attacks) [20, 32, 40, 46, 49, 60].

- If DSybil increased the trust additively instead of multiplicatively, then one can easily show that DSybil's loss would be linear with respect to M instead of logarithmic.
- In a non-overwhelming round, currently DSybil simply returns a uniformly random object (i.e., it does not distinguish different objects with different votes). If, instead, the algorithm returned the object whose voters have the largest combined trust, then the loss would be linear with respect to M under the following attack. Consider a sequence of rounds where each round has exactly one good object and one bad object. Initially, all the M sybil identities vote on the good object, until Alice consumes the first good object. Each sybil identity now has a positive trust value that is no smaller than the trust of any honest identity. In addition, no identity has trust value larger than $\alpha S_\Sigma / M$. For each of the next $M/(W+1)$ rounds, the adversary uses $(W+1)$ sybil identities to vote on the bad object. As long as $(\alpha S_\Sigma / M) \cdot (W+1) < 1$ (which easily holds), no objects will be overwhelming. Since the good object has at most W voters while the bad object has $(W+1)$ voters (each with either the same or larger trust), the algorithm will recommend bad objects in all these $M/(W+1) = \Omega(M)$ rounds (for constant W).
- Consider a modified version of our algorithm where in each round, the algorithm recommends each object O with certain probability. The probability is proportional to the total trust of the voters on O. Consider the same attack as above except that after Alice consumes the first good object, for each of the next $\sqrt{M/W}$ rounds, the adversary uses $\sqrt{MW}$ sybil identities to vote on the bad object. This means that the probability mass on the bad object will be at least $\sqrt{M/W}$ times the probability mass on the good object. Based on the mean of geometric distributions, one can easily show that the expected number of bad objects consumed (before consuming any further good objects) will be $\Omega(\sqrt{M/W}) = \Omega(\sqrt{M})$ (for constant W).
- Currently DSybil only increases the trust of the voters on non-overwhelming good objects. Imagine that we instead simply increase the trust of the voters on any good objects. Consider an adversary that initially has all M sybil identities vote on all the good objects. After $\Theta(\log(M \cdot c / S_\Sigma))$ rounds, all of the sybil identities will have a trust of c. Then, each of them can cause 1 loss (i.e., making some bad object overwhelming), resulting in a total loss of $\Omega(M)$. Even if we further optimize and pick the overwhelming object whose voters have the largest combined trust, the loss will still reach $\Omega(M/W) = \Omega(M)$ (for constant W).

Lifespan and population. Like other systems, a recommendation system's robustness against malicious behavior comes from the help provided by the honest identities (or votes from the guides in our case). The number of malicious identities that other systems (e.g., majority voting, byzantine consensus, and DHTs) can tolerate is usually directly proportional to the number of honest identities (i.e., their *population*). In contrast, the following will use our lower bound results to show that to tolerate more sybil identities in a recommendation system, the *lifespan* (or more precisely the number of votes cast throughout the identity's lifetime) of the honest identities is far more important than the *population*.

Our upper bound and lower bound both have a $\mathscr{D}_f$ multiplicative term. Given a set $\mathscr{U}$ of objects, $\mathscr{D}_f$ tends to be inversely proportional to the lifespan but is independent of the population of the honest identities. For example, if we assume that each guide votes on y random objects in $\mathscr{U}$, then $\mathscr{D}_f$ will be $\Theta(|\mathscr{U}|/y)$ for any constant $f < 1$. Increasing the lifespan y by x times will thus reduce loss by a multiplicative

factor of x.

To see how population may affect the loss, imagine for now that we fix the voting pattern and the lifespan of all honest identities. As an example, let us replicate each honest identity into x distinct new identities where each new identity casts the same votes as the original identity. Doing so obviously increases the population by x times. Using similar arguments as in Theorems 3 and 5 and assuming $x < M$, one can show that both the upper bound and the lower bound now become $\Theta(\mathscr{D}_f \log(M/x)) = \Theta(\mathscr{D}_f(\log M - \log x))$. In other words, the loss is reduced only by some additive logarithmic term of x.

7. Loosely Bounding M

The only remaining missing piece of the DSybil defense is to design a means to loosely bound M, the number of sybil identities voting on any given object. DSybil uses simple computational puzzles to bound M; the recommendation algorithm's strong guarantees mask the key drawbacks of computation puzzles, as discussed in this section. Alternatively, one could apply sophisticated approaches such SybilLimit [62] and SumUp [57] (that require a social network) to better bound M. However, DSybil's logarithmic loss implies that the extra benefit gained will likely be limited in DSybil's context.

Why computational puzzles suffice. The first problem with computational puzzles is that the adversary can be much more resourceful than a typical user. DSybil's logarithmic loss helps to mitigate this problem. Second, even with recurring computational puzzles, the adversary can still abandon old identities, reclaim resources, and create new identities. This will result in an unlimited number of sybil identities over time (though they are not simultaneously active). Fortunately, our algorithm only requires the number of sybil voters on any given object to be bounded. Notice that an object is likely to have some limited lifetime. For example, most objects in p2p file sharing systems (e.g., Credence) are only active for a few weeks, and a news story on Digg gets most of its votes within the first few days. We can thus naturally impose a limited-duration (e.g., a few weeks) *voting window* for each object during which votes are permitted on the object. Given a voting window, computational puzzles can be used to bound M, as shown next.

Computational puzzles in DSybil. With its logarithmic loss, DSybil can afford to start actively bounding M only when the maximum number of votes on individual objects exceeds some large threshold (e.g., 1 billion). As a comparison point, the total number of PCs in the world was only around 1 billion in 2007 [50]. Thus, only during rather serious attacks will such a threshold be exceeded.

After the threshold is exceeded, DSybil periodically (e.g., at the beginning of every calendar week) releases a fresh puzzle seed. For centralized application scenarios such as Digg, the server can readily release a fresh seed weekly. For decentralized cases, DSybil simply uses the concatenation of the weekly closing prices of some common stocks as the seed [41], which are readily available on the Internet to all users. The seed, together with an identity's unique name, will instantiate a computational puzzle (e.g., of 1 minute) for the identity. Solving the puzzle will allow the identity to vote in the *next* calendar week (on an arbitrary number of objects), except that solving the very first week's puzzle enables voting in the first two weeks. The puzzle does not need to be solved online and the solution can be submitted anytime during the week.

To understand the guarantees, consider an object with a one-week voting window, which can span at most two calendar weeks. To cast 10 billion votes for the object, the adversary needs to solve 5 billion 1-minute puzzles at least in one of the two weeks. Assuming that an average bot is up 50% of the time, doing so would roughly require a million-node botnet (i.e., $(5 \times 10^9)/(0.5 \times 7 \times 24 \times 60) \approx 10^6$).

In the above design, we chose not to use a one-time computational puzzle for each identity (e.g., at registration time), because the adversary could then hoard identities and M could be unbounded. We did not use a recurring puzzle for each vote to avoid delays when voting. For similar reasons, in DSybil solving the puzzle gives voting privileges for the next week instead of the current week.

DDoS attacks. An adversary controlling a million-node botnet may be able to launch severe DDoS attacks on various components of the system. Even simply casting 10 billion votes may already overload the system. Dealing with such attacks is still an active research area [25] and is beyond the scope of this paper. DSybil is quite simple at the implementation level, and it is unlikely to be the bottleneck component. As future work, we intend to incorporate DSybil into real-world recommendation systems (e.g., Digg) and study the system's robustness against DDoS.

8. Evaluation

We have implemented DSybil as a toolkit in Java. In this section, we first validate DSybil's key assumption on small $\mathscr{D}_f$, based on a number of real-world datasets (Sections 8.1 and 8.2). Then, we demonstrate DSybil's end-to-end guarantees and its growing defense (Section 8.3).

8.1. Starting Point: Small Dimensions in Digg

DSybil targets applications, such as Digg, where objects are either good or bad. Thus, our study starts with a crawled dataset (called `digg`) of the news stories (together with user votes) from Aug 2007 to July 2008 on digg.com. This large-scale dataset has a total of 496,622 users, 36,103 objects, and 44,741,196 votes. All the votes are positive votes, because

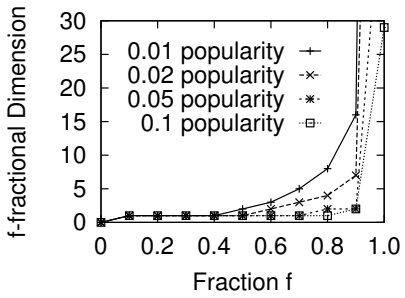

Figure 4. Dimension in digg.

Digg does not show the negative votes (i.e., the "buries"). In any case, DSybil does not use negative votes. Although Digg is not free of attacks [24], there is evidence that the scale of the attack is limited (at least before 2009). For example, researchers [57] found that the fraction of "suspicious objects" on Digg is only around 0.25%. Thus we expect the dataset to be "clean enough" for our experiments.

The setting. Consider an imaginary Digg user Alice. Recall that $\mathscr{D}_f$ is the smallest number of guides that can cover f fraction of the good objects in $\mathscr{U}$. For this experiment, we construct $\mathscr{U}$ to contain all the objects in digg. Because the dataset only contains positive votes, it does not tell us which objects Alice will consider as bad. We will pessimistically require the $\mathscr{D}_f$ guides to cover f fraction of *all* the objects in $\mathscr{U}$. Because a guide does not vote on bad objects, doing so would ensure that the $\mathscr{D}_f$ guides cover at least f fraction of the good objects.

To determine $\mathscr{D}_f$, we further need to know which users are guides (i.e., with the same "taste" as Alice). Such information is not available from digg. To overcome this, let Alice's *popularity* be the fraction of honest users who are guides. We vary Alice's popularity and for each popularity value x, we pick a random x fraction of the users in the dataset as guides. For example, with a 0.02 popularity, there will be one user with the same taste as Alice in every 50 users. In order to be able to benefit from other users' votes, we expect Alice to have some minimal level of popularity. In other words, if Alice has a rather esoteric taste, the help she can obtain from other users will be minimal. For example in digg, each object on average has 1239 votes. With lower than 0.01 popularity, on average fewer than 12 votes out of these 1239 votes are from users with the same taste as Alice. Thus, we consider popularities of at least 0.01. Finally, because determining $\mathscr{D}_f$ is NP-hard, we will use a greedy heuristic and all our results are thus pessimistic upper bounds on $\mathscr{D}_f$.

Dimension in digg. Figure 4 plots how $\mathscr{D}_f$ changes with f, under different popularity values. Even for a small popularity of 0.01, the dimension is quite small (below 5) for any $f \leq 0.6$. Section 8.3 later will show that under $M = 10^{10}$, such dimension translates to 4%–12% per-round loss in some specific scenarios. These small dimension values are not obvious: If all users in digg were to have cast an equal number of votes, $\mathscr{D}_{0.6}$ would have been at least $\frac{0.6\times 36{,}103}{44{,}741{,}196/496{,}622} \approx 240$ instead of below 5. Thus, the small dimension is due to some users casting significantly more votes than others. Our experiments further show (not included in Figure 4) that these results are robust: The dimension remains small (around 5) even after removing the 100 heaviest voters from the dataset.[9] This is easy to understand in hindsight—with 0.01 popularity, only 1 user out of every 100 users is a guide after all.

The dimension increases quickly as f approaches 1.0, even under 0.1 popularity. This is due to the "coupon collection" effect. Namely, as $f \to 1.0$, it becomes harder and harder to cover additional objects because many of the objects covered are redundant. This also shows why it is critical for DSybil not to rely on $\mathscr{D}_{1.0}$ being small.

Finally, our digg dataset is one-year long, and one may wonder how $\mathscr{D}_f$ will potentially grow for an Alice using Digg for more than a year. $\mathscr{D}_f$ can increase due to the "churn" of the guides, where existing guides leave the system and new guides are needed to cover objects in later rounds. However, there are strong reasons to believe that $\mathscr{D}_f$ will likely increase *at most* linearly with Alice's lifespan. For example, if Alice's lifespan reaches two years, the worst case would be for all the existing guides to leave the system after one year. Assuming the new guides in the second year have a similar heavy-tail voting pattern as the old guides, this can at most double the dimension. On the other hand, as long as the dimension increases at most linearly, Alice's per-round loss will never increase. Thus, our later per-round loss results are likely to be upper bounds for any Alice with a lifespan of at least a year.

8.2. Generalization: Small $\mathscr{D}_f$ is Fundamental

Supplementary datasets. We are interested in generalizing our earlier findings from digg. There are a number of other Digg-like websites (such as youtube.com, reddit.com, mixx.com, motorpulse.com, tribalwar.com, dotnetkicks.com) that involve user voting and recommendation of new stories or video clips. The websites youtube.com, reddit.com, and mixx.com only provide the total number of votes on each object and do not show who the voters are. Thus, we are unable to use them. The remaining three websites (motorpulse.com, tribalwar.com, and dotnetkicks.com) have a rather small numbers of users and objects. Among these three, we crawled only the largest one, dotnetkicks.com (a news site for .NET-related techniques), and use it as a supplementary dataset (Table 3). This dataset is about 2 orders of magnitude smaller than digg. We also use three

9. Our notion of "heaviest voters" is different from the notion of "top Digg users" from http://socialblade.com/digg/topusers.html. "Heaviest voters" is purely based on vote count, while "top Digg Users" are those who have submitted many stories that are later promoted by Digg.

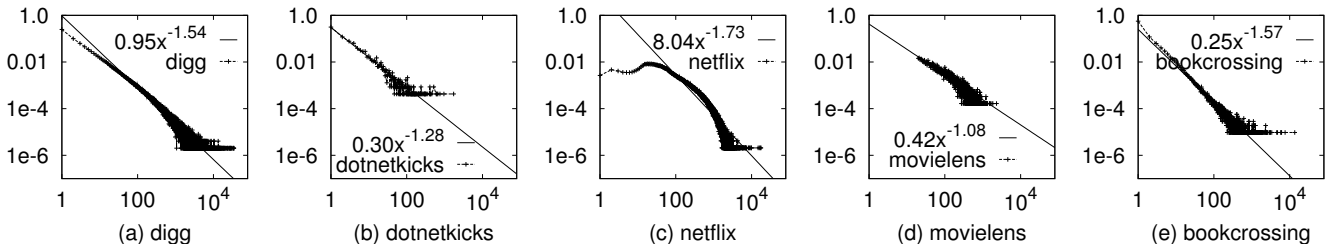

Figure 5. Pareto fittings. In all figures, x-axis is the # of votes cast, and y-axis is the fraction of users casting x votes.

dataset	# users	# objects	# votes
dotnetkicks	2,339	2,252	45,236
netflix	480,189	17,770	100,480,507
movielens	6,040	3,952	1,000,209
bookcrossing	105,283	341,133	1,149,780

Table 3. Additional supplementary datasets.

publicly available movie/book rating datasets, netflix [48], movielens [33], and bookcrossing [15] (Table 3). Finally, while Credence [60] is one of our target applications, we were not able to obtain Credence's dataset.

Our thesis. Ideally, one would directly study the dimensions in the supplementary datasets. Unfortunately, as mentioned above, dotnetkicks has an overly small number of users/objects. The other datasets contain fine-grained scalar/numerical ratings, so their dimension is not even well-defined. Instead, we use these datasets in the following way.

We will focus on the fundamental cause of small dimensions in digg. Figure 5(a) plots the distribution of the number of votes cast by individual users in digg. The figure shows a rather straight line on log-log scale, indicating that the distribution is heavy-tail (or more precisely Pareto). Let T be the number of good objects in $\mathscr{U}$. With a Pareto vote distribution, an honest user will cast exactly i votes (on i random objects out of the T objects) with probability $a \cdot i^{-b}$. Here a is a normalization factor and is a function of b and T.

Our thesis is that such heavy-tail distribution is what fundamentally causes the small dimensions in digg. The intuition is that a small number of users near the "heavy tail" can often cover a substantial fraction of all the objects (as long as these users' voting patterns, as in those in digg, do not have excessive correlation among themselves). Based on this thesis, instead of directly validating small dimensions, we will validate the following *sufficient* (but not *necessary*) conditions for small dimensions:

1) The vote distribution is Pareto, and
2) A Pareto vote distribution usually implies small dimension.

We will use the four datasets in Table 3 to validate the first statement, and use simulation/analysis to validate the second one. Notice that such a decomposition can be powerful—even if one rather pessimistically dismisses the representativeness of all our datasets, as long as the heavy-tail distribution (which is rather universal/fundamental) applies, the dimension will likely to be small. Additionally, because we are concerned only with the "heavy-tailness" of the vote distribution from the honest identities, we do not expect the (byzantine) votes from the sybil identities to affect such "heavy-tailness".

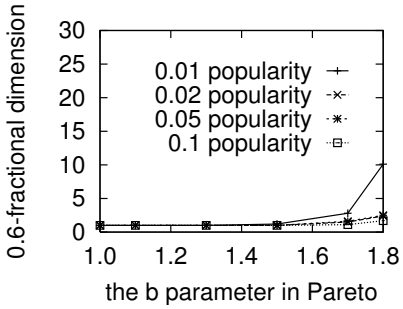

Figure 6. Dimension resulting from Pareto.

Is vote distribution Pareto? Figures 5(b) through 5(e) fit the four datasets to Pareto, with b values ranging from 1.08 to 1.73. The flat components at the tails are due to the finite number of users in the datasets. We ignore the flat parts when fitting to Pareto. This makes the tails "lighter" (i.e., b larger) and makes our later results pessimistic. The figure shows that all datasets except netflix have quite good fit. Pareto does not fit netflix well at the beginning. But since dimension is mainly determined by the "tail" portion, we do not expect the beginning portion to be critical.

Does Pareto imply small dimension? We construct synthetic datasets with synthetic voting patterns for the guides based on given Pareto distributions, and then determine the dimension of the datasets. In our construction, each guide votes for a random i objects, with i drawn from the given Pareto distribution. To allow a direct comparison with digg, we set $T = 36,103$ and the total number of honest users to be $492,622$. We again consider different popularity values for Alice. For space limitations, we only present the results under $f = 0.6$ (Figure 6). Obviously, the dimension only becomes smaller for $f < 0.6$. We consider $b \in [1.0, 1.8]$ since the b values in our fittings fall within such a range. If b is excessively large, Pareto's tail is simply not "heavy" any more for practical purposes. The figure shows that in all cases,

the dimension remains small. This confirms that a Pareto vote distribution implies small dimension (with reasonable b ranges). Independent of these experimental results, we have also obtained similar results directly via analytical methods. We omit the details of the analysis due to space limitations.

8.3. Loss under the Worst-case Attack

Even though we have implemented DSybil, we *explicitly choose not to focus on studying DSybil's loss against individual attack strategies experimentally* (though we provide some simple examples in the next section). As emphasized in Section 1, a (human) attacker will always try to find and use the single most effective strategy, which is dependent on the algorithm being attacked. It is not meaningful to discuss "typical" attack strategies. On the other hand, we do not know either how to construct the worst-case attack against DSybil (otherwise we could inject such attack experimentally). Thus, we directly use Equation 8 to (pessimistically) upper bound Alice's loss during the worst-case attack, based on the dimension values in the datasets. Using this upper bound can only make our results worse (since the upper bound may not be tight).

The setting. We consider the following scenario for a imaginary Digg user Alice. We pessimistically use 0.01 as Alice's popularity. There are 36,103 news stories in `digg`, and we partition them into 361 sets with 100 objects per set. Each set thus roughly corresponds to a day in this one-year dataset. Since `digg` does not have negative votes, for simplicity, we assume all of the objects to be good. (Additionally, most of them are voted for by at least one guide already.) We construct one round corresponding to each set. Each round contains all the 100 good objects in the set, and additionally another 100 bad objects. We assume that Alice wants to consume 20 objects in each round (i.e., read 20 news stories every day). We model this as 20 rounds where Alice consumes one object per round (as described in Section 5.3).

We use the voting pattern of the guides in `digg` to determine $\mathscr{D}_f$. Other information in the dataset, such as the voting patterns of the non-guides, is irrelevant. Similarly, we do not need to specify the voting patterns of the sybil identities or the votes on the bad objects, since Equation 8 holds under all possibilities. We plug $\mathscr{D}_f$ into Equation 8 with $u = 200$, $p = 0.5$, $v = 20$, and $W = 30,720$ (observed from the dataset). Since Equation 8 holds for $\forall f > \frac{v}{u \cdot p} = 0.2$, we compute the bound on $E[L]$ using the f value (> 0.2) that minimizes the bound. Notice that DSybil's expected loss ($E[L]$) under the worst-case attack is already fixed given the `digg` dataset. But precisely determining $E[L]$ is challenging. By varying the f value in Equation 8, we are simply finding the best upper bound for $E[L]$ (i.e., the upper bound that is the tightest). DSybil itself does not need to search for such f because f is not a parameter in DSybil.

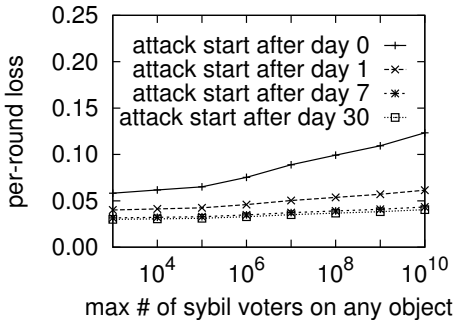

Figure 7. Alice's per-round loss when the attack starts after Alice has used DSybil for a given number of days (assuming the worst-case attack).

DSybil's loss under the worst-case attack. The "attack after day 0" curve in Figure 7 illustrates how Alice's per-round loss changes with M. For this scenario, per-round loss is the fraction of bad new stories recommended to (and read by) Alice. The figure shows that the per-round loss is below 6% under $M = 1000$, and is only around 12% even if M reaches 10 billion. Putting it another way, even if individual objects have up to 10^{10} votes from the sybil identities, only roughly 2.4 objects are bad out of the 20 objects recommended by DSybil every day. In comparison, an average good object only has 12 votes from the guides and 1,239 votes from the guides and non-guides combined. Section 7 showed that a million-node botnet will likely be needed to cause M to reach 10^{10}. Higher popularity values for Alice will result in smaller loss. For example, for 0.1 popularity and $M = 10^{10}$, the per-round loss is 3.8% (not shown in the figure).

Growing defense. Next, we illustrate the growing defense of DSybil, when the adversary starts attacking after Alice has used DSybil for some time. (We still consider the worst-case attack, and here the worst-case attack means the attack that incurs the largest loss among all attacks that start after some initial attack-free rounds.) For this experiment, our DSybil toolkit first processes the dataset for some number of attack-free rounds, and then outputs the total loss so far, together with the trust values of all identities. These values, together with the dimension of the remaining rounds, enables us to use Equations 6 and 8 to bound the loss under the worst-case attack.

To show the robustness of the results against different voting patterns of the non-guides, we perturb their votes in `digg` in various ways. We have experimented with various ways to do such perturbation (e.g., changing a non-guide's vote on a good object to a vote on some random bad object with certain probability). We observe that the impact of all the perturbations we tried are always negligible. We find that the fundamental reason is that to have a non-trivial impact, the voting pattern of the non-guides needs to be carefully constructed to mislead DSybil. Different from the votes from the sybil identities, the "noise" from the non-guides can

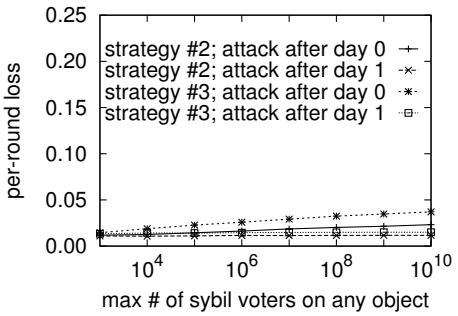

Figure 8. Alice's per-round loss under example attack strategies.

rarely be "strategically misleading". Because of this, this section (and the next section as well) presents our results based only on the following specific perturbation: For each vote from a non-guide, we change it to a vote for a random bad object with probability of 0.5 (other probabilities yield even lower loss).

Figure 7 illustrates Alice's per-round loss when the attack starts after Alice has used DSybil for a given number of days (each day corresponds to 20 rounds). The results show that DSybil's growing defense is rather prominent. Even if Alice has used DSybil for only one day before the attack starts, her per-round loss dramatically drops from around 12% to 6% (under $M = 10^{10}$). If Alice has used DSybil for a month, her per-round loss further drops to around 4%. Assuming that an average honest user's lifespan (of using DSybil) is one year, and assuming that the attack starts at a random point of time, $364/365 \approx 99.7\%$ of the users will have used DSybil for a day when the attack starts. We have also experimented with higher popularity values. For example, for 0.1 popularity and $M = 10^{10}$, the per-round loss will be 0.70% if the attack starts after day 1.

Figure 7 further shows that with some initial attack-free rounds, Alice's loss becomes less sensitive to M. This is within expectation. Namely, larger M will result in smaller seed trust for the critical guides. But the initial attack-free rounds are likely to have already assigned seed trust (whose value is independent of M) to most of the critical guides.

8.4. Loss under Example Attack Strategies

To illustrate DSybil's behavior under attack, we present DSybil's loss under some example attack strategies, as observed experimentally on our DSybil implementation. We do not intend to be extensive or exhaustive here, since the previous section already studied DSybil's loss under the worst-case attack. We continue with the exact same setting from the previous section (with a pessimistic popularity of 0.01). As the baseline, our experiments show that when there is no attack, Alice's per-round loss is 1.3%. This non-zero loss is essentially caused by those non-guides who have different taste from Alice's. In the next, we consider three example strategies.

Strategy #1 is a naive one: For every bad object in each round, the adversary simply uses M identities to vote on it. The identities used on different objects (and in different rounds) can either be the same or different. Our experiments show that such attack has no effect on loss. This is easy to understand—these sybil identities will never get any seed trust in DSybil.

For Strategy # 2, on the first day of attack, the adversary creates M sybil identities and split them into two equal sets X_1 and Y_1. Each identity in X_1 votes for all good objects in that day, and each identity in Y_1 votes for all bad objects. In the second day, X_1 is split into two equal-sized sets X_2 and Y_2. Additionally, identities in Y_1 are replaced with $M/2$ fresh sybil identities. Half of fresh sybil identities are added to X_2 and the remaining half are added to Y_2. Same as before, identities in X_2 vote for all good objects, while identities in Y_2 vote for bad objects. Such process is repeated for all the remaining days.

For Strategy #3, initially, the adversary creates 100 sets (since there are 100 good objects and 100 bad objects in each day) of sybil identities where each set has M identities. The execution (361 days) is broken into multiple segments, where each segment consists of $x+y$ days. For the first x days of each segment, each of the 100 sets of M identities votes for a distinct good object in each day. In the next y days of each segment, each set votes for a distinct bad object in each day. Our experiments show that using $x = 5$ and $y = 1$ incurs the largest loss, and thus we use such setting. We have also performed some simplified analysis (details omitted), which indeed predicts that such setting will incur the largest loss.

Figure 8 plots DSybil's per-round loss under Strategy #2 and #3, when the attack starts after day 0 or 1. It is clear that the loss is below the loss under the worst-case attack in Figure 7.

9. Conclusion

This paper presented *DSybil*, a novel defense for diminishing the influence of sybil identities in recommendation systems. DSybil has provable guarantees that are optimal. Our evaluation showed that DSybil's loss would remain small even under a potential sybil attack launched from a million-node botnet.

Acknowledgment. We thank Avrim Blum, Wee Sun Lee, Siddhartha Srinivasa, and Nathan Ratliff for very helpful discussions on related issues in the machine learning context. We thank Brad Karp for useful discussions about attacks on Digg and Wikipedia. We thank Avrim Blum and the anonymous reviewers for helpful comments on this paper. This work is partly supported by NUS Young Investigator Award R-252-000-334-123.

References

[1] B. Acohido and J. Swartz. Botnet scams are exploding. March 16, 2008, USA Today. http://www.usatoday.com/tech/news/computersecurity/2008-03-16-computer-botnets_N.htm.

[2] N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. Tell me who I am: An interactive recommendation system. In *ACM SPAA*, 2006.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1), 2003.

[4] B. Awerbuch, Y. Azar, Z. Lotker, B. Patt-Shamir, and M. R. Tuttle. Collaborate with strangers to find own preferences. In *ACM SPAA*, 2005.

[5] B. Awerbuch and T. P. Hayes. Online collaborative filtering with nearly optimal dynamic regret. In *ACM SPAA*, 2007.

[6] B. Awerbuch and R. D. Kleinberg. Competitive collaborative learning. *J. Comp. Sys. Sci.*, 74(8), 2008.

[7] B. Awerbuch, A. Nisgav, and B. Patt-Shamir. Asynchronous active recommendation systems. In *OPODIS*, 2007.

[8] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Collaboration of untrusting peers with changing interests. In *ACM Electronic Commerce*, 2004.

[9] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Adaptive collaboration in peer-to-peer systems. In *ICDCS*, 2005.

[10] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Improved recommendation systems. In *ACM-SIAM SODA*, 2005.

[11] B. Awerbuch and C. Scheideler. Towards a scalable and robust DHT. In *ACM SPAA*, 2006.

[12] R. Bazzi and G. Konjevod. On the establishment of distinct identities in overlay networks. In *ACM PODC*, 2005.

[13] A. Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1), 1997.

[14] A. Blum and Y. Mansour. From external to internal regret. *J. Machine Learning Research*, 8, 2007.

[15] http://www.informatik.uni-freiburg.de/~cziegler/BX/.

[16] N. Borisov. Computational puzzles as sybil defenses. In *IEEE P2P*, 2006.

[17] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *USENIX OSDI*, 2002.

[18] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[19] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *ACM P2PEcon*, 2005.

[20] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *ACM WIDM*, 2005.

[21] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM CCS*, 2002.

[22] M. Dell'Amico and L. Capra. Sofia: Social filtering for robust recommendations. In *IFIPTM*, 2008.

[23] http://www.digg.com/.

[24] An interview with Digg top user. http://www.invesp.com/blog/social-media/an-interview-with-digg-top-user.html, 2008.

[25] C. Dixon, T. Anderson, and A. Krishnamurthy. Phalanx: Withstanding multimillion-node botnets. In *USENIX NSDI*, 2008.

[26] J. Douceur. The Sybil attack. In *IPTPS*, 2002.

[27] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *ACM STOC*, 2002.

[28] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM Electronic Commerce*, 2004.

[29] A. Fiat, J. Saia, and M. Young. Making Chord robust to byzantine attacks. In *ESA*, 2005.

[30] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *ACM STOC*, 1997.

[31] http://www.stealthfriendbomber.com/.

[32] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Trans. on Sensor Networks*, 4(3), 2008.

[33] http://www.grouplens.org/.

[34] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. Technical report, Purdue Univ., 2007. http://www.cs.purdue.edu/homes/zagedj/docs/reputation_survey.pdf.

[35] J. Hopcroft and D. Sheldon. Manipulation-resistant reputations using hitting time. In *WAW*, 2007.

[36] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW*, 2003.

[37] A. Kast. The digg recommendation engine. http://digg.com/whitepapers/recommendationengine, 2008.

[38] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. In *ACM COLT*, 2008.

[39] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *IEEE FOCS*, 1998.

[40] S. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *WWW*, 2004.

[41] H. H. Lee, E.-C. Chang, and M. C. Chan. Pervasive random beacon in the internet for covert coordination. In *Information Hiding*, 2005.

[42] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *FOCS*, 1988.

[43] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in P2P file sharing systems. In *IEEE INFOCOM*, 2005.

[44] A. Mislove, A. Post, K. Gummadi, and P. Druschel. Ostra: Leveraging trust to thwart unwanted communication. In *USENIX NSDI*, 2008.

[45] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[46] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. on Internet Technology*, 7(4), 2007.

[47] A. Nakamura. Learning specialist decision lists. In *ACM COLT*, 1999.

[48] http://www.netflixprize.com/.

[49] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. on Internet Technology*, 4(4), 2004.

[50] Microsoft wants to double the number of PCs in the world by 2015. http://www.informationweek.com/news/windows/microsoft_news/showArticle.jhtml?articleID=199200360, 2007.

[51] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging. In *USENIX HotBots*, 2007.

[52] http://razor.sourceforge.net/.

[53] P. Resnick and R. Sami. The influence limiter: Provably manipulation-resistant recommender systems. In *ACM RecSys*, 2007.

[54] P. Resnick and R. Sami. The information cost of manipulation-resistance in recommender systems. In *ACM RecSys*, 2008.

[55] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *SWSA ISWC*, 2003.

[56] H. Rowaihy, W. Enck, P. Mcdaniel, and T. La-Porta. Limiting sybil attacks in structured peer-to-peer networks. In *INFOCOM*, 2007.

[57] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *USENIX NSDI*, 2009.

[58] http://www.tubeautomator.com/.

[59] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Telling humans and computers apart. In *IACR Eurocrypt*, 2003.

[60] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *USENIX NSDI*, 2006.

[61] A. Yao. Probabilistic Computations: Towards a Unified Measure of Complexity. In *IEEE FOCS*, 1977.

[62] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *IEEE S&P*, 2008.

[63] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against sybil attacks via social networks. *IEEE/ACM Transactions on Networking*, 16(3), June 2008.

Appendix

We prove Theorem 6 below first, and later use it to prove Theorem 5.

Theorem 6: For any p (in the following 3 cases), M, and any recommendation algorithm, we can always construct a sequence of rounds, objects, and votes with the following properties: The fraction of good objects in each round is at least p, there are *at most M sybil identities total*, $\mathscr{D}_f = 1$ for all $0 < f \leq 1$, and the algorithm will incur at least

- $\frac{1}{2}\lfloor \log_2 M \rfloor$ expected loss if $p = 0.5$.
- $\frac{1}{4} \cdot \left\lfloor \frac{1}{p} \right\rfloor \cdot \left\lfloor \frac{\log_2 M}{\log_2(1/p)} \right\rfloor$ expected loss if $0 < p < 0.5$.
- $\frac{1}{2} \cdot \left\lfloor \frac{\log_2 M}{\log_2(3/(1-p))} \right\rfloor$ expected loss if $0.5 < p < 1$.

Proof sketch: In all constructions below, there are M sybil identities and exactly one honest identity (which is a guide). When we say "remove an identity", we mean that identity will never vote again in future rounds.

Construction for $p = 0.5$. We construct a sequence of rounds where each round has exactly one good object and one bad object. In the first round, half of the $M+1$ identities (including the guide) vote for the good object and against the bad object. The other half vote for the bad object and against the good. Since the two objects and the votes on them are completely symmetric, no algorithm can achieve an expected loss of below 0.5 in this round. We then remove those $(M+1)/2$ sybil identities voting for the bad object. In the second round, the remaining identities again split into two equal-sizes subsets, and repeat the previous procedure. We can construct $\lfloor \log_2(M+1) \rfloor \geq \lfloor \log_2 M \rfloor$ such rounds before running out of sybil identities. This then incurs at least $\frac{1}{2}\lfloor \log_2 M \rfloor$ expected loss for any algorithm.

Construction for $0 < p < 0.5$. Let $u = \lfloor 1/p \rfloor$. The first round contains one good object and $u-1$ bad objects. We partition all $M+1$ identities into u equal-sized sets. The set of identities containing the guide will all vote for the good object and against all other objects. Each of the other sets will vote for one (distinct) bad object and against all other objects. We replicate such construction u times to obtain a total of u rounds (i.e., the objects in different rounds are the same). We call these u rounds as the first *phase*.

In the first round, because all objects are symmetric, the best thing that any algorithm can do is to pick a random object. (This argument can also be made rigorous by a trivial invocation of Yao's Theorem [61].) If a bad object happens to be selected, then the algorithm can "blacklist" that object (potentially together with the set of identities voting for it). Without loss of generality, we assume the algorithm will never select that bad object in future rounds, since otherwise the expected loss can only be larger. In the second round, the remaining $u-1$ objects are all still symmetric, which implies that the best thing that any algorithm can do is to pick a random object out of those $u-1$ objects. On the other hand, if a good object happens to be selected in the first round, the algorithm will never need to incur any loss in the remaining rounds in the first phase.

To formalize, let random variable X denote the total loss during the first phase. The event $X = i$ (for $1 \leq i \leq u-1$) happens when the algorithm picks bad objects in the first i rounds, and then picks the good object in the $(i+1)$th round. Thus we have $Pr[X = i] = \frac{u-1}{u} \cdot \frac{u-2}{u-1} \cdot \ldots \cdot \frac{u-i}{u-i+1} \cdot \frac{1}{u-i} = 1/u$, and also:

$$E[X] \quad = \quad \sum_{j=0}^{u-1} j \cdot Pr[X = j] = \frac{u-1}{2} = \frac{1}{2}\left\lfloor \frac{1}{p} \right\rfloor - \frac{1}{2} \geq \frac{1}{4}\left\lfloor \frac{1}{p} \right\rfloor$$

After the first phase, we remove all identities except the set containing the guide. We again split the remaining $\lfloor (M+1)/u \rfloor$ identities into u subsets, and then construct the second phase in the same way. One can easily show that we can construct $\lfloor \log_u(M+1) \rfloor \geq \left\lfloor \frac{\log_2 M}{\log_2(1/p)} \right\rfloor$ phases before running out of sybil identities. This then yields a lower bound of $\frac{1}{4} \cdot \lfloor \frac{1}{p} \rfloor \cdot \left\lfloor \frac{\log_2 M}{\log_2(1/p)} \right\rfloor$ on expected loss.

Construction for $0.5 < p < 1$. Here we will need to construct the next round *adaptively* based on the algorithm's choices in the previous rounds. Let $u = \lceil \frac{2}{1-p} \rceil$. The first round contains exactly one bad object and $u-1$ good objects. We partition all identities into u equal-sized sets. The set of identities containing the honest identity will all vote against the single bad object and vote for the $u-1$ good objects. Each of the other $u-1$ sets will vote against one (distinct) good object and vote for all other objects.

Because all objects are symmetric in the first round, the best thing that any algorithm can do is to pick a random object. Let the object picked be O. If O is bad, then we construct a trivial round and replicate it $u/2-1$ times as the next $u/2-1$ rounds. The trivial round will contain only one good object and all identities vote for it. If O is good, then we remove the set of (sybil) identities voting against O. The second round will contain exactly one bad object and $u-2$ good objects. Similar as the first round, the set of identities containing the honest identity will vote against the single bad object and vote for the $u-2$ good objects. Each of the other $u-2$ sets will vote against one (distinct) good object and vote for all other objects. All objects are thus again symmetric in the second round. We continue such process for $u/2$ rounds. One can easily confirm that the fraction of good objects in all the rounds is at least p.

We call the above $u/2$ rounds as the first *phase*. The probability that the algorithm never picks the bad object during the first phase is $\frac{u-1}{u} \cdot \frac{u-2}{u-1} \cdot \ldots \cdot \frac{u-(u/2)}{u-(u/2)+1} = \frac{1}{2}$. Thus with probability $\frac{1}{2}$, the algorithm will pick the bad object and incur 1 loss. The expected loss thus is $\frac{1}{2}$.

Finally, after the first phase, we remove all identities except the subset containing the honest identity. We again split the remaining identities into u subsets and repeat the process. One can easily show that we can construct $\lfloor \log_u(M+1) \rfloor \geq \left\lfloor \frac{\log_2 M}{\log_2(3/(1-p))} \right\rfloor$ phases before running out of sybil identities.

This then yields an expected loss of at least $\frac{1}{2} \cdot \left\lfloor \frac{\log_2 M}{\log_2(3/(1-p))} \right\rfloor$. $\square$

Proof for Theorem 5: We simply use Theorem 6 to construct $\mathscr{D}$ independent executions and then concatenate them together. Because there is at least one good object per round and one honest identity per execution (different in each execution), we have that $\mathscr{D}_f$, the smallest number of guides that can cover an $f > 0$ fraction of the good objects in every round, is $\mathscr{D}$. $\square$