Distributed Computing Column 43
Using Social Networks to Overcome Sybil Attacks

Idit Keidar
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
idish@ee.technion.ac.il

We open the new academic year with Haifeng Yu's article on overcoming sybil attacks using social networks. In a *sybil attack*, a malicious user assumes multiple identities, and uses them to pose as multiple users. Sybil attacks are a threat of the new millennium – they arise in Internet-based distributed systems with a dynamic user population. Indeed, such attacks were not a concern in traditional distributed systems, where the set of participating processes was statically pre-defined. Sybil attacks are inherently difficult to deal with in systems where users do not wish to disclose binding private information, like credit card numbers. A recent popular approach for overcoming sybil attacks is using *social networks*. Intuitively, even if a malicious user can create many identities, he will have a hard time getting many honest users to befriend all of them in a social network. Thus, the *graph structure* of a social network can assist in revealing sybil nodes.

In this column, Haifeng Yu presents a tutorial on how social networks can be leveraged to defend against sybil attacks, and a survey of recent suggestions employing this approach. Though Haifeng tackles the problem from a theoretical standpoint, (proving formal bounds etc.), this direction has garnered more attention from the systems community, perhaps because sybil attacks are perceived as a real threat for which social networks can provide a viable solution. Yet it appears that much theory for sybil defense using social networks is yet to be developed. In particular, there is a need for better understanding the graph properties of social networks, and how they can be leveraged to achieve provable bounds on false positives and false negatives in the detection of sybil nodes. Haifeng touches upon some of these directions in the last section. Many thanks to Haifeng for his article!

Call for contributions: I welcome suggestions for material to include in this column, including news, reviews, open problems, tutorials and surveys, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways.

Sybil Defenses via Social Networks: A Tutorial and Survey

Haifeng Yu
School of Computing
National University of Singapore
Republic of Singapore
haifeng@comp.nus.edu.sg

Abstract

The *sybil attack* in distributed systems refers to individual malicious users joining the system multiple times under multiple fake identities. Sybil attacks can easily invalidate the overarching prerequisite of many fault-tolerant designs which assume that the fraction of malicious nodes is not too large. This article presents a tutorial and survey on effective sybil defenses leveraging *social networks*. Since this approach of sybil defenses via social networks was introduced 5 years ago, it has attracted much more attention from the research community than many other alternatives. We will first explain the intuitions and insights behind this approach, and then survey a number of specific sybil defense mechanisms based on this approach, including SybilGuard, SybilLimit, SybilInfer, Gatekeeper, SumUp, Whanau, and Ostra. We will also discuss some practical implications and deployment considerations of this approach.

1 The Sybil Attack Problem

The term *sybil attack* was first coined by Douceur [9] in 2002, after a book titled "Sybil" which describes a patient with sixteen different personalities. Analogously, the sybil attack in distributed systems refers to individual malicious users joining the system multiple times under multiple fake identities (called *sybil identities*). The malicious users can use these sybil identities in a byzantine fashion to effectively launch a wide range of follow-up attacks. For example, these sybil identities can easily "out vote" the honest users in various collaborative tasks, such as in byzantine consensus, in collaborative recommendation systems, and in redundant routing and data replication in DHTs.

While the sybil attack was first discussed [9] in the context of peer-to-peer systems, it is worth noting that it applies to centralized systems as well. For example, Amazon's centralized recommendation system is still susceptible to sybil attacks, and malicious users may launch such attacks to influence the ratings of the products. To effectively thwart sybil attacks, the centralized system needs to be able to bind identities to actual human beings. Such binding is often the most problematic part, since it *by definition* requires verifying private/personal information such as credit card numbers.

Without the system having such a binding capability, the sybil attack problem becomes rather challenging. Simple sybil defenses include using CAPTCHAs, IP address filtering, or computational puzzles. All of these unfortunately can only deter casual attackers. A quick Google search shows that the "market price" for solving 1,000 CAPTCHAs is just a few dollars. IP address filtering such as allowing only one account/identity per IP address will cause serious problems for users behind NATs, and yet still cannot guard against attackers controlling a subnet. Finally, computational puzzles require each identity to solve a computational problem with some controlled "hardness", such as finding the pre-image of an x-bit one-way hash. This has rather limited effectiveness since a determined attacker can easily have much more computational resources than an honest user with an outdated PC. Also the system needs to arrange these puzzles to be solved by all identities simultaneously, otherwise even an attacker with limited computational resources can solve many puzzles over a longer time window.

Because of the above reasons, the sybil attack is usually considered to be rather hard to defend against. In fact, Douceur's paper [9] has already proved some simple results showing that sybil defense is not possible in usual models of distributed systems.

Why is the problem important? Before moving on to effective sybil defenses leveraging special assumptions/properties that are not captured by usual models of distributed systems, it is worthwhile to understand the importance of sybil defense. A grand goal in distributed computing, since the very beginning, has been to tolerate byzantine failures in various contexts (such as in distributed consensus, in quorum systems, or in DHTs). Over the years, the distributed computing community has developed many smart and effective techniques to achieve such a goal. These techniques almost always require an overarching prerequisite, namely, the fraction of the byzantine nodes cannot be too large. As the fraction of byzantine nodes approaches one, there is very little we can do. This overarching prerequisite, unfortunately, becomes invalidated with sybil attacks. Without addressing sybil attacks first, all these previous techniques might not even be relevant when sybil attacks are possible.

This leads to a second question: Why were sybil attacks not addressed from the very beginning, or what has made sybil attacks possible? Traditional distributed systems are often of a *closed* nature. For example, distributed airline reservation systems are closed in the sense that they have a static set of nodes, and to add a new node to the system, some system administrator needs to incorporate that node into the system. Furthermore, the users of the system are bound to real human beings. With the advent of the Internet and its fast growth, many distributed systems today are *open-access*. These open-access systems aim to provide service to any user on the Internet, and the users may sometimes contribute to the system as well. For example, the users may contribute storage resources in peer-to-peer backup systems or contribute votes in recommendation systems. While being open-access enables distributed systems to maximize technology penetration and to leverage user contribution, such property also has made possible sybil attacks.

The importance of the sybil attack problem has been widely recognized by the research community since it was coined in the 2002 paper [9], which has received over 2,000 citations by now on Google Scholar. In the real world, sybil attacks have been observed (e.g., in the Maze peer-to-peer system [17, 26]), and there are even off-the-shell software tools such as Tube Automator[1] and Friend Bomber[2] for launching sybil attacks.

Roadmap. The remainder of this article presents a tutorial and survey on effective sybil defenses leveraging *social networks*. Since this approach of sybil defenses via social networks was introduced 5 years ago, it has attracted much more attention from the research community than many other alternatives. Section 2 explains the system model and the key insights behind the approach. Section 3 discusses the first designs leveraging these insights. A number of other designs will be surveyed in Section 4. Section 5 discusses some practical implications and deployment considerations.

[1]`http://www.tubeautomator.com/`
[2]`http://www.stealthfriendbomber.com/`

2 System Model and Key Insights

This section first formalizes the system model and then explains the key insights behind the approach of leveraging social networks for sybil defense. These insights were initially made and leveraged by SybilGuard [30] and SybilLimit [27, 28], which first introduced this approach. Later these insights became the common foundation for many other sybil defenses via social networks, including those to be surveyed later in this article.

2.1 System model and goal

Consider a distributed system $\mathcal{D}$ for which sybil defense is needed. The system has n honest human beings as honest users, each with one *honest identity* (e.g., one account). Honest identities obey the protocol. The system also has one or more malicious human beings as malicious users, each with one or more identities. To unify terminology, we call all identities created by malicious users as *sybil identities*. Sybil identities are byzantine, and thus may behave and also collude in adversarial ways.

The goal of sybil defense is to allow any given honest identity V to label any other given identity S as either "honest" or "sybil". Both false positives and false negatives are allowed in the labeling process. Completely eliminating false negatives (i.e., labeling sybil nodes as "honest") is never necessary since the distributed system $\mathcal{D}$ should be able to tolerate some fraction of byzantine identities, otherwise it is not robust even without sybil attacks. A sybil defense thus only needs to bound the total number of false negatives below the tolerance threshold of $\mathcal{D}$. For example, for byzantine consensus, this tolerance threshold is usually $n/2$ (since $(n/2)/(n + n/2) = 1/3$). Most applications can also easily live with a small fraction (e.g., 20%) of false positives (i.e., labeling honest nodes as "sybil"). For example in a peer-to-peer backup system, V labeling S as "sybil" simply means that V will not trust S for storing V's data. A 20% false positive rate would mean that V will still trust 80% of the honest nodes, and be able to leverage their resources. This is equivalent to the system capacity being reduced by 20%. Similarly in a recommendation system, V will still use votes from 80% of the honest identities. With this observation, sybil defenses often allow some bounded fraction of false positives as well.

Finally in many sybil defense mechanisms, two honest identities V_1 and V_2 are allowed to label the same S differently. This is clearly different from the system as a whole assigning a single label to S. Allowing labeling to be done from each individual node V's perspective is critical to enable these sybil defenses to break symmetry (discussed later), as well as to do the labeling in a decentralized manner.

2.2 Key insights behind the approach

The social network. The term *social network* has been used by different people to refer to many different things, and thus we want to clarify what exactly we mean by a social network here. The *social network* for a distributed system $\mathcal{D}$ is an undirected graph $\mathcal{G}$ where the vertices of the graph are the identities in $\mathcal{D}$. From now on, we will use the term "node" to refer to both an identity and the corresponding vertex in $\mathcal{G}$. An edge between two nodes in $\mathcal{G}$ corresponds to human-established trust relations between the two corresponding identities in the real world. For example, we may have edges between two colleagues or two relatives. The *honest region* of $\mathcal{G}$ is defined to be $\mathcal{G}$'s subgraph consisting of all the honest nodes and edges among them. We similarly define the *sybil region*. The edges in $\mathcal{G}$ connecting the honest region and the sybil region are called the *attack edges* (Figure 1). Note that the distributed system $\mathcal{D}$ itself may have nothing to do with social networks, and we are not trying to defend against sybil attacks *in* social networks. Rather, given $\mathcal{D}$

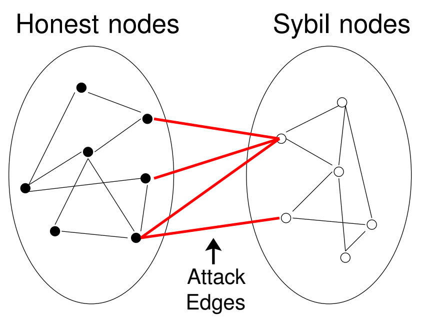

Figure 1: The social network and the attack edges. (From [28].)

(e.g., a DHT), we defend against sybil attacks in $\mathcal{D}$ by leveraging its corresponding social network $\mathcal{G}$. Also, the sybil defense mechanism does not know which edges are attack edges.

Letting the protocol know the social network. Information about the social network $\mathcal{G}$ might not be directly available in $\mathcal{D}$, in which case the users need to provide enough information about $\mathcal{G}$ to the defense mechanism. If the defense mechanism is decentralized (such as SybilGuard and SybilLimit), then each node needs to tell its local instance of the defense protocol who are its neighbors in $\mathcal{G}$. To do so, each pair of neighboring nodes in $\mathcal{G}$ needs to first establish a shared secret symmetric key, via out-of-band means. This key is called the *edge key* since it corresponds to an edge in $\mathcal{G}$. Each node then will feed all its edge keys to its local instance of the defense protocol. The purpose of these edge keys is to allow mutual authentication between a pair of neighbors when they communicate. Otherwise it is possible for one node A to communicate with another node B while faking as B's friend C.

If the defense mechanism is centralized, then the centralized server can directly collect information from each user regarding who are the user's neighbors in $\mathcal{G}$. Note that care still must be taken (e.g., via out-of-band means) to ensure proper authentication. Otherwise, a malicious user may create an account A under the name of B's friend C, and then fool B into forming an edge with A. This can be prevented if B authenticates A via out-of-band means.

The key insights. Sybil defenses via social networks leverage the following three key insights on $\mathcal{G}$ to defend against sybil attacks:

1. The number of attack edges is independent of the number of sybil identities, and is limited by the number of trust relation pairs between malicious users and honest users. This holds since each attack edge corresponds to a distinct edge key established between an honest node and a sybil node controlled by a malicious user. Regardless how many sybil identities that malicious user creates, it can only establish one edge key (via out-of-band means) with the honest user.

2. If the malicious users create too many sybil identities and given that the number of attack edges remains fixed, the *cut* along the attack edges will have a small *quotient*. Here the quotient is the quotient between the number of edges in the cut (i.e., the number of attack edges) and the number of nodes disconnected due to the cut. If we know beforehand that a social network with only honest nodes (e.g., the honest region in $\mathcal{G}$) will not contain a cut with similarly small quotient, then we can tell that $\mathcal{G}$ is "abnormal".

3. Finally, it is necessary to break symmetry in order to properly label nodes. For example, the two regions in Figure 1 might be completely symmetric, in which case we would still not know which

region is honest after detecting the cut with small quotient. Breaking such symmetry is easy. If the local user running the protocol is honest, then the local protocol can safely claim that the region containing the local user is honest. If the local user is malicious, we do not care anyway.

The second insight above relies on an assumption that a social network with only honest nodes will not contain a cut with an excessively small quotient. We will revisit this assumption later.

A strawman design based on cuts. Let us see how one could already use the previous insights to construct a strawman sybil defense:

StrawmanDesign 1. *Given an honest node V, we search for a subgraph $\mathcal{H}$ of $\mathcal{G}$ such that i) $\mathcal{H}$ contains V, ii) $\mathcal{H}$ has n nodes (where n is the total number of honest nodes and let us assume for now that n is known), and iii) the quotient of the minimum quotient cut of $\mathcal{H}$ is not excessively small. V labels all nodes in $\mathcal{H}$ as "honest" and all other nodes as "sybil".*

Clearly there are many possibilities for the $\mathcal{H}$ in StrawmanDesign 1, and the honest region of $\mathcal{G}$ is one such possibility. If we were lucky to find that as $\mathcal{H}$, we would have labeled all nodes correctly. In other possibilities, our labels may be largely incorrect. For example, $\mathcal{H}$ could contain mostly sybil nodes and contain only a rather short path of honest nodes connecting V to the sybil region. Note that since the diameters of social networks tend to be rather small (e.g., "six degrees of separation"), there will likely exist such a short path for almost any V. To avoid such pathological $\mathcal{H}$, intuitively one needs to "grow" $\mathcal{H}$ in a "balanced" way so that $\mathcal{H}$ does not "grow" too much into the sybil region (at least in cases where V is not too close to the attack edges). Formalizing such "balanced growth" and designing a way to achieve it, however, are non-trivial.

3 Sybil Defenses via Social Networks: First Designs

Having discussed the key insights, we continue to elaborate how SybilGuard [30] and SybilLimit [27, 28] leverage these insights for fully decentralized sybil defense with clean formal guarantees. We discuss these two efforts first because they were the first designs leveraging these insights and many concepts/techniques developed there have been later used in other designs. Understanding them will thus facilitate our later survey. SybilLimit is the direct sequel of SybilGuard, and has made SybilGuard obsolete. We will thus only discuss SybilLimit from now on.

3.1 From cuts to mixing time

Random walks and mixing time. As discussed in StrawmanDesign 1, a key to effective sybil defense via social networks is to achieve "balanced growth" with good formal guarantee. It turns out that random walks are a good way to do such "balanced growth" with desirable guarantees. In addition, random walks can be done conveniently in a decentralized way without global knowledge.

To leverage random walks on $\mathcal{G}$, we need to translate our discussion on the quotient of the cuts to discussion on *mixing time*. *Mixing time* describes how fast random walks in a given graph approach the stationary distribution of that graph. Large mixing time means that a random walk needs to be rather long in order to approach the stationary distribution. A connected graph having a cut with a small quotient is guaranteed to have a small conductance and thus a large mixing time.[3] This in turn means that too many

[3]Note that the reverse is not necessarily true.

n	total number of nodes in the honest region of the social network
m	total number of edges in the honest region of the social network
g	total number of attack edges (assumed to be $o(n/t)$)
t	mixing time of the honest region of the social network (e.g., $O(\log n)$)
w	length of the random walks, which is set to t in SybilLimit

Table 1: Key notations.

sybil nodes in $\mathcal{G}$ will increase the social network's mixing time. Again, if we assume that a social network with only honest nodes will have a mixing time that is not excessively large[4], $\mathcal{G}$ will be "abnormal" in terms of mixing time. This central assumption on mixing time is made explicit below:

Assumption 1. *The honest region (i.e., subgraph) of $\mathcal{G}$ has a mixing time no larger than $t(n)$, where $t(n)$ is a function of the size n of the honest region.*

We will write $t(n)$ as t for simplicity. The end guarantees of SybilLimit depend on t, with smaller t yielding better guarantees. The original SybilLimit papers [27, 28] assume that $t = O(\log n)$. This is not actually necessary: Larger t will simply linearly increase the number of sybil nodes incorrectly labeled as "honest" in SybilLimit. This will be clear from the discussion later.

Theoretical evidences. Several theoretical evidences exist to support Assumption 1. The first evidence was based on Kleinberg's widely accepted social network model [14]. Boyd et al. [4] proved that Kleinberg's model has a mixing time of $O(\log n)$ when the graph's base structure is a grid and when the r parameter in the model is 0. When $r = 0$, a node's remote friends are chosen uniformly randomly out of all nodes in the graph. As r increases, the remote friends tend closer and closer to the local node. Later, Flaxman [11] proved that a much wider class of randomly perturbed graphs have $O(\log n)$ mixing time. In particular, these graphs include Kleinberg's model when i) the base structure is a ring and $0 \leq r < 1$, or ii) the base structure is a grid and $0 \leq r < 2$. Note that all these evidences are stronger than what Assumption 1 needs, which does not really require the mixing time to be as small as $O(\log n)$.

Lower bound. Clearly, if the malicious users have introduced so few sybil nodes that $\mathcal{G}$ has a normal mixing time of t, then no approach based on mixing time can possibly tell that those nodes are sybil. It thus has been proved [27] that for any graph $\mathcal{H}$ with n honest nodes and $t = O(\log n)$ mixing time, it is always possible for the adversary to introduce $c \cdot g$ sybil nodes (for any constant c) via g attack edges so that the mixing time of the augmented graph $\mathcal{G}$ is $O(\log(n+cg))$. This means that protocols based on Assumption 1 with $t = O(\log n)$ will label at least $\Omega(g)$ sybil nodes as "honest". One can naturally extend this argument to other t functions.

3.2 The SybilLimit protocol

The technical materials in SybilLimit are mainly about how to use random walks in $\mathcal{G}$ to exploit its "abnormal" mixing time for differentiating sybil nodes from honest nodes, in a decentralized way. The notion of graph "growth" via random walks is implicit rather than explicit here. To understand why random walks enable graph "growth" in a nice way, we need to discuss the concepts of *escaping random walks* and *escaping nodes*. Table 1 summarizes the key notations that will be used in the following.

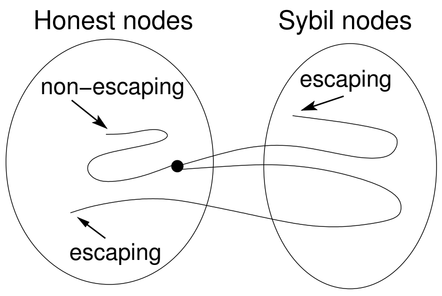

Figure 2: Escaping and non-escaping random walks. (From [28].)

3.2.1 Escaping random walks and escaping nodes

Consider a random walk in $\mathcal{G}$, starting from some given honest node V and with some given length. We say that the random walk is *escaping* if it ever crosses any attack edge (Figure 2). Otherwise it is *non-escaping*. Escaping random walks are bad "growth" since they get into the sybil region. Even if an escaping random walk later comes back to the honest region, it is still bad because the sybil nodes may have manipulated this random walk. Since they are byzantine, sybil nodes may behave arbitrarily, and may not follow the random walk protocol at all. On the other hand, a non-escaping random walk is *entirely* in the honest region, and thus has nice probabilistic properties regardless of the byzantine behavior of the sybil nodes.

The escaping probability of a random walk starting from V is independent of the behavior of the sybil nodes. But it does partly depend on the location of V in the honest region. If V is close to the attack edges, then the escaping probability will likely be higher. The first technical result in SybilGuard/SybilLimit is the following clean theorem regarding the escaping probability:

Theorem 1. *Let g be the total number of attack edges, n be the total number of honest nodes, and assume that all the honest nodes form a connected component. Then the escaping probability of a length-w random walk starting from a uniformly random honest node is at most gw/n.*

While reasoning about the escaping probability of a random walk starting from a given node V is complex, the above theorem assures that the average probability across all V's is at most gw/n. This is true even if the locations of the attack edges are adversarial.

From the average of gw/n, one can easily infer that there can be at most ϵ fraction of the honest nodes whose corresponding probability is above $(gw)/(n\epsilon)$, since otherwise the average would be above gw/n. Those (at most) ϵ fraction of the honest nodes are called *escaping nodes*. The remaining (at least) $1 - \epsilon$ fraction of the honest nodes are called *non-escaping nodes*.[5] A random walk starting from a non-escaping node thus has a well-bounded escaping probability of $O(gw/n)$. If $O(gw/n) = o(1)$, then random walks starting from a non-escaping node are almost never bad "growth".

Only protecting non-escaping nodes. SybilLimit will provide provable guarantees only for the non-escaping nodes. This is not a major concern since that fraction of escaping nodes can be made small. The escaping nodes are likely to be rather close to the attack edges, and it is easy to see why they cannot be protected. For example, if a node has 10 neighbors out of which only one is honest, then from this node's perspectives, the sybil region may look more like "home". The lack of protection also constitutes a penalty and disincentive to this node for making bad friends.

[4]Note that this assumption is stronger than not having cuts with small quotients.

[5]To facilitate understanding, this definition is slightly different from the more stringent definition in [27, 28].

3.2.2 A strawman design based on random walks

We can already design a sybil defense by utilizing the properties discussed so far:

StrawmanDesign 2. *Consider any given non-escaping node V (which must be honest). To label other nodes, V simply performs $\theta(n)$ independent random walks starting from itself. Each such random walk has a length of t, where t is the mixing time of the honest region of $\mathcal{G}$. V labels the last node visited by each of these random walks as "honest". Note that V may label the same node as "honest" multiple times. All other nodes are labeled "sybil". Implicitly, V has "growed" a subgraph (consisting of the nodes labeled as "honest") by roughly growing one node per random walk. Also remember that different V's may assign different labels to the same node, and thus grow the subgraph differently.*

StrawmanDesign 2 actually already offers some nice properties. To see why, imagine that all honest nodes have the same degree, and thus the stationary distribution of the honest region of $\mathcal{G}$ is a uniform distribution. Each length-t non-escaping random walk from V will roughly land on a uniformly random honest node.[6] Now assume that $g = o(n/t)$, which implies that the escaping probability of a random walk starting from V is $O(gt/n) = o(1)$. This in turn means that among the $\theta(n)$ "honest" labels that V assigns, $1 - o(1)$ fraction of those go to honest nodes. Furthermore, each of these labels would roughly be given to a uniformly random honest node. This is sufficient to cover $(1 - \epsilon)$ fraction of all the honest nodes. In other words, V will label most honest nodes as "honest". Next, V may also label some sybil nodes as "honest", due to escaping random walks. Roughly, $O(gt/n)$ fraction of the $\theta(n)$ random walks are escaping. This means that V roughly labels $O(gt/n) \times \theta(n) = O(gt)$ sybil nodes, or $O(t)$ sybil nodes per attack edge, as "honest". Recall that $O(gt/n) = o(1)$, and thus among all the nodes labeled as "honest", only a vanishingly small fraction will be sybil nodes.

StrawmanDesign 2 has two drawbacks. First, in most distributed systems, V would not be interested in labeling all nodes. Rather, V may only want to label those nodes that V happens to interact with. StrawmanDesign 2 does not allow V to do so efficiently: Even if V only wants to label a single node S, V will still need to perform $\theta(n)$ random walks. Second, StrawmanDesign 2 relies on the degrees of the honest nodes being roughly similar. Otherwise the stationary distribution can be far from uniform, and the $\theta(n)$ honest labels may concentrate on a small fraction of honest nodes.

Using StrawmanDesign 2 for obtaining random views. As a side note, if a node performs x independent random walks from itself in StrawmanDesign 2, the node will obtain a subgraph containing roughly x nodes. The above discussion shows that this subgraph actually is a nice random *view* of the system, in the sense that each node in the view is roughly a uniformly random node and the view contains only a vanishingly small fraction of sybil nodes. We will see later that SybilLimit as well as some other sybil defense mechanisms [16, 23] internally uses this approach to obtain random views despite sybil attacks. It is worth noting that there has been some interesting recent efforts (e.g., Brahms [3]) on obtaining nice random views in large-scale systems. These efforts do not need to rely on social networks (and the accompanying assumptions), though they usually do not aim at addressing sybil attacks either.

3.2.3 SybilLimit's approach

To overcome the above drawbacks in StrawmanDesign 2 as a sybil defense mechanism, SybilLimit uses an intersection trick to enable V to label S efficiently. V first performs $\theta(\sqrt{m})$ independent random walks of length t, where m is the total number of edges in the honest region of $\mathcal{G}$. The *tail* of a random walk is defined

[6]This claim can be proved [27] but is not immediately obvious, since it conditions upon the walk being non-escaping.

to be the last edge traversed by the random walk. V thus will have $\theta(\sqrt{m})$ tails from all these random walks, where some tails may be redundant. All these tails essentially form a nice random view for V. S obtains $\theta(\sqrt{m})$ tails similarly. V labels S as "honest" if and only if they have at least one common tail.

For V to label S, the above protocol involves $\theta(\sqrt{m})$ random walks. Also notice that if V later needs to label a second node, V can simply reuse all its tails without doing any more random walks. Usually social networks are rather sparse since the node degrees are by far smaller than the graph size. This implies that $\sqrt{m}$ is often much smaller than n or even close to $\sqrt{n}$. Thus the above design tends to be much more efficient than StrawmanDesign 2, which needs $\theta(n)$ random walks. Also, tails are edges instead of nodes, and the stationary distribution on edges is always a uniform distribution in any graph. This avoids the need to make assumptions on node degrees.

Properties for labeling honest nodes. Let us examine the guarantees offered by the above process. We will assume that $g = o(n/t)$, so that the escaping probability of a random walk starting from a non-escaping node V is $o(1)$. Now if V is a non-escaping node, then V will have roughly $\theta(\sqrt{m})$ tails in the honest region of $\mathcal{G}$. Furthermore, each such tail will roughly be a uniformly random edge in the honest region. The same applies to S's tails, if S is a non-escaping node. Birthday paradox immediately tells us that with good probability, V and S will have a common tail in the honest region.[7] This means that if S is a non-escaping node (and thus must be honest), then V will label S correctly as "honest" with good probability. Since most honest nodes are non-escaping nodes, V will label most honest nodes correctly.

Properties for labeling sybil nodes. Now let us move on to sybil nodes. A non-escaping node V will label a sybil node S incorrectly as "honest" if V and S share some common tail ζ. Separately consider two cases:

- The common tail ζ is in the sybil region. V knows that roughly $o(1)$ fraction of its tails are in the sybil region, but it does not know exactly which ones. For such a tail ζ, the (byzantine) sybil nodes may introduce arbitrary intersections with it. To avoid labeling too many sybil nodes as "honest" due to such ζ, V imposes a quota on each of its tails: For each tail, V will only use it as the common tail for labeling a certain number of nodes as "honest". After the quota is reached, the tail will no longer be used for intersection.

  For example, imagine that we set a quota of $\theta(n/\sqrt{m})$ on each tail. Collectively, the total quota for the honest nodes will be $\theta(n)$ since V has $\theta(\sqrt{m})$ tails in the honest region. Thus one may reasonably hope that the n honest nodes to be labeled will not be negatively affected by such quota. On the other hand, since V has at most $O(gt/n) \cdot \theta(\sqrt{m})$ tails in the sybil region, the total quota for the sybil nodes will be $O(gt/n) \cdot \theta(\sqrt{m}) \cdot \theta(n/\sqrt{m}) = O(gt)$. In other words, V will label $O(gt)$ sybil nodes as "honest" due to those tails in the sybil region.

  The hard technical part here is to prove that the honest nodes to be labeled will indeed not be negatively affected by the quota. Further, without knowing n, there is no easy way to set the quota to $\theta(n/\sqrt{m})$. Instead, SybilLimit uses a floating quota that is a function of the total number of nodes labeled as "honest" so far. These are the most technically involved parts in SybilLimit – interested readers are referred to [27, 28].

- The common tail ζ is in the honest region. Such a tail ζ, which is the tail of some sybil node but is itself in the honest region, is called a *tainted tail*. A tainted tail essentially corresponds to a random walk that starts from a sybil node, crosses some attack edge, and enters the honest region. Since the (byzantine) sybil nodes may not follow the random walk protocol at all, the portion of that walk

[7]Note that this simply lower bounds the intersection probability since we have not considered potential common tails in the sybil region. But we do not want to leverage those common tails since the sybil nodes are byzantine.

in the sybil region can be arbitrary. SybilLimit designs an interesting *random routes* technique (see Appendix A) to bound the total number of tainted tails, across all sybil nodes collectively, within $O(gt\sqrt{m})$. This technique only needs to rely on the property of the random walks (routes) while they are in the honest region.

V's $\theta(\sqrt{m})$ tails in the honest region are roughly uniformly randomly distributed. Thus the expected number of common tails between V's tails and all those $O(gt\sqrt{m})$ tainted tails is roughly $O(gt)$. Applying a Markov inequality shows that with probability $1-\delta$ (for any given positive δ), there are only $O(gt)$ common tails. This means that V will only label $O(gt)$ sybil nodes as "honest" due to these tainted tails.

The main theorem. The following main theorem summarizes SybilLimit's end guarantees:

Theorem 2. *Assume that Assumption 1 holds and also assume that $g = o(n/t)$. For any given constants $\epsilon > 0$ and $\delta > 0$, there exists a set of $(1-\epsilon)n$ honest nodes such that using SybilLimit with the proper parameters will guarantee that for any honest node V in that set, with probability at least $1-\delta$, V labels at most $O(t)$ sybil nodes per attack edge as "honest", and at least $(1-\epsilon)n$ honest nodes as "honest".*

Note that *regardless* of the value of t, as long as $g = o(n/t)$, the theorem guarantees that among all the nodes labeled as "honest", only a vanishingly small fraction will be sybil nodes. Such a guarantee is sufficiently strong for most applications.

From another perspective, the theorem also shows that larger t linearly increases the number of sybil nodes incorrectly labeled as "honest" (i.e., false negatives) per attack edge, if we choose to increase the length of the random walks with t as in SybilLimit. Alternatively, it is also possible to fix the length of the random walks at some $t_0 < t$. In such a case, larger t will result in more false positives (i.e., honest nodes incorrectly labeled as "sybil") instead of false negatives.

3.2.4 Additional discussions on SybilLimit

Estimating unknown parameters. So far the discussion has been implicitly assuming that SybilLimit knows two global parameters: the mixing time (t) and the total number (m) of edges in the honest region of the social network. On the other hand, it may appear that before figuring out which nodes are honest, SybilLimit would not know these two quantities even assuming global knowledge. SybilLimit overcomes this problem in the following way.

In practice, the mixing time t often increases rather slowly with n and can be insensitive to n. For example, if $t = O(\log n)$, then SybilLimit can simply use a t around 20 or 30. This would usually be sufficient to allow for n as large as one million. SybilLimit can always use a pessimistically large t. The *only* drawback of doing so is that the number of false negatives increases with t linearly. Second, SybilLimit provides an interesting *benchmarking technique* to estimate the total number (m) of edges. The technique never over-estimates m, but under-estimation is possible. When under-estimation does occur, the technique ensures that SybilLimit's end guarantees on false positives and false negatives remain intact. Interested readers are referred to [27, 28] for details.

Near optimality. SybilLimit bounds the number of sybil nodes incorrectly labeled as "honest" within $O(t)$ per attack edge. This translates to $O(\log n)$ for $t = O(\log n)$. Recall from Section 3.1 that the lower bound under $t = O(\log n)$ is $\Omega(1)$ incorrectly labeled sybil nodes per attack edge. This means that SybilLimit is only $O(\log n)$ factor away from the optimal, when $t = O(\log n)$. Similar near-optimality arguments can be made under other t functions.

Protocol	*Assumption*	*Main technique*	*Provable end guarantee?*	*Complete decentralized design?*
SybilGuard [30] and SybilLimit [27, 28]	Assumption 1	random walk	√	√
SybilInfer [8]	Assumption 1	random walk	×	×
Gatekeeper [23]	Assumption 1 and 2	breadth-first search and random walk	√	√
SumUp [24]	Assumption 1*	adaptive max flow	√	×
Applying community detection algorithms [25]	not clearly made, but likely similar to Assumption 1	detecting social communities	×	×
Whanau [16]	Assumption 1	random walk and layered-IDs for DHT	√	√
Ostra [18]	not clearly made, but likely similar to Assumption 1	"remove" certain edges based on user feedback	×	×

*Note that [24] only mentions Assumption 1, but we are not sure whether a similar assumption as in Gatekeeper is needed.

Table 2: Summary of six sybil defenses via social networks, besides SybilGuard and SybilLimit. The bottom two are for sybil defenses in specific contexts, while the remaining ones are general. See text for detailed discussions.

Numerical examples. As some concrete numerical examples, it has been shown that on several social networks with around 100,000 to 1,000,000 honest nodes, SybilLimit labels about 95% of the honest nodes as "honest". It also labels about 10 to 20 sybil nodes as "honest" per attack edge.

4 Sybil Defenses via Social Networks: Other Efforts

Since SybilGuard [30] and SybilLimit [27, 28] introduced leveraging social networks to defend against sybil attacks, there have been a number of interesting follow-up research efforts, largely from the systems community. This section discusses and compares six such efforts (Table 2). Among these, four protocols [8, 23, 24, 25] are general sybil defenses, while the remaining two [16, 18] are for sybil defenses in two specific contexts.

We first summarize the commonalities of these six protocols. They all share the foundation of Section 2, and make Assumption 1 (i.e., mixing time) or similar/stronger assumptions on the social network. About half of these protocols use random walks, for which most of the discussion from Section 3.1 to 3.2.2 applies. None of the four general sybil defenses allows efficient pairwise labeling as in SybilLimit, meaning that Section 3.2.3 is unique to SybilLimit. Finally, as summarized in Table 2, two of these six protocols are fully decentralized, while others do not provide a complete decentralized design. Three of the six protocols offer provable end guarantees[8], while others are heuristics having only experimental results. We next discuss the individual protocols in more details.

[8]We note that in a *security* setting, having a provable end guarantee is particularly valuable. After all, the attacker is intelligent and will actively seek the most effective attack strategy specifically designed to foil the given design. Thus only the worst-case matters. Pure experimental methods, regardless of how extensive, may always miss the worst-case unless one explicitly proves that the worst-case is covered.

4.1 SybilInfer [8]

How it works. Same as SybilLimit, the SybilInfer protocol relies on Assumption 1 and uses random walks. SybilInfer first transforms the social network $\mathcal{G}$ to a new graph $\mathcal{G}'$ (or more precisely, a new Markov chain $\mathcal{G}'$), so that the node stationary distribution on $\mathcal{G}'$ is uniform. This is done via a well-known approach which adjusts the transition probability on each edge "$A \to B$" from $1/(\text{degree of } A)$ in $\mathcal{G}$ to $\min(1/(\text{degree of } A), 1/(\text{degree of } B))$ in $\mathcal{G}'$. SybilInfer seems to implicitly assume that if $\mathcal{G}$ has small mixing time, then $\mathcal{G}'$ will have small mixing time too.[9] Let t' be the mixing time of $\mathcal{G}'$.

Consider a given node A in $\mathcal{G}'$. Conceptually, SybilInfer does a sufficient number of length-t' random walks starting from A, and uses those to estimate the distribution of the last node visited by such a random walk. Let us call this distribution as A's *tail distribution*. Now imagine that we obtain such a tail distribution for each node. SybilInfer then conceptually finds a set X of nodes such that the tail distribution of any node in X is roughly a uniform distribution on X. SybilInfer labels all nodes in X as "honest" and all other nodes as "sybil". For the intuition behind this approach, recall the discussion in Section 3.2.1. It is not hard to see that the set of non-escaping nodes will roughly satisfy the requirement on X, when the number of attack edges is not too large.

Formal guarantees. While the intuition works, it seems difficult to prove end guarantees for SybilInfer's design. The original SybilInfer paper [8] does not prove any end guarantee on false positives and false negatives. It only proves that the presence of sybil nodes will increase the mixing time of the graph, and thus affect the probability that a random walk starting from the honest region will end within that region. There is no result on *how much* the probability is affected, how well the events observed by SybilInfer concentrate near this probability, or how truthful the assigned labels are.

Experimental results and the need of global knowledge. Experimentally on a synthetic social network with 1,000 nodes, SybilInfer shows that it significantly outperforms SybilLimit in terms of the fraction of sybil identities among all identities labeled as "honest". Finally, the SybilInfer protocol assumes global knowledge about the social network.[10]

4.2 Gatekeeper [23] and SumUp [24]

Assumptions. The Gatekeeper protocol is an improved and decentralized version of the centralized SumUp protocol. We will focus on Gatekeeper first since it is conceptually cleaner and easier to understand. Gatekeeper relies on a stronger assumption than Assumption 1. It requires the honest region of the social network not only to have small mixing time, but also to be *reasonably balanced*, as defined next. Imagine that we do a breadth-first search from a given honest node A, until the total number of honest nodes *covered* (i.e., touched by the search) reaches $n/2$. Now if we do the above from every honest node A, then an honest node B may be covered up to n times. A social network $\mathcal{G}$ is called *reasonably balanced* if $1 - o(1)$ fraction of the honest nodes are covered $\theta(n)$ times in the above process. Gatekeeper thus needs the following Assumption 2, in addition to Assumption 1.

Assumption 2. *The honest region (i.e., subgraph) of $\mathcal{G}$ is reasonably balanced.*

Different from Assumption 1 which has some supporting theoretical evidences, currently we are not yet aware of such evidences for Assumption 2. On the other hand, the experimental results from [23] do show that Gatekeeper works well on several large social network datasets.

[9]Note that this assumption actually does *not* hold, mathematically. But to allow further discussion, let us assume it did hold. It is perhaps possible to avoid this assumption if we leverage the stationary distribution on edges.

[10]Even though the paper [8] alludes to decentralized designs, it does not provide a complete design that is decentralized.

How it works. Gatekeeper "grows" a subgraph $\mathcal{H}$ within $\mathcal{G}$ in a special breadth-first fashion, starting from some random *roots*. Each root here is selected using a random walk. By Theorem 1 from Section 3.2.1, if this random walk starts from a non-escaping node and under proper setting, such a root will have a good probability of being an honest node. Next Gatekeeper grows a special breadth-first search tree from each root, in the following way. The root first receives $\theta(n)$ *tickets* from Gatekeeper, and each node at a given tree level receives a certain number of tickets from nodes in its parent level. Upon receiving the tickets, a tree node A first consumes one ticket for itself, and then evenly splits the remaining tickets to all nodes that are at the next tree level and to which A is connected. Notice that those nodes are a superset of A's children in the tree. A tree node A with no tickets to give out will not grow the tree further. This means that the resulting tree is likely to be breadth-first near the root, but no longer so afterwards since some of the tree nodes will run out of tickets to further grow the tree. Gatekeeper labels a node B as "honest" if and only if it is *covered* by (i.e., present in) at least some threshold number of trees grown from all those roots.

Clearly, sybil nodes are byzantine and thus may not follow the above ticket distribution protocol at all. This is not a problem since Gatekeeper shows that the total number of tickets that ever get into the hands of the sybil nodes is properly bounded.

Formal guarantees. Gatekeeper proves that if the number (g) of attack edges is $O(n/\log n)$, then it will label most honest nodes as "honest", while only labeling $O(\log g)$ sybil nodes per attack edge as "honest". For large g such as $\theta(n^{1-c})$ where $0 < c < 1$, this guarantee is the same as SybilLimit. For small g such as $g = \theta(1)$, this guarantee is $O(1)$ and is asymptotically better than SybilLimit. Note that because Gatekeeper requires stronger assumption than Assumption 1, the lower bound (Section 3.1) of labeling $\Omega(1)$ sybil nodes per attack edge as "honest" does not directly carry over to Gatekeeper.

Experimental results. Experimentally, Gatekeeper shows that it significantly outperforms SybilLimit in terms of the number of false negatives while having the same false positive rate, under a small number (i.e., 60) of attack edges on a synthetic social network and crawled YouTube and Digg graphs (all with about 500,000 nodes) .

SumUp [24]. Finally, SumUp is the predecessor of Gatekeeper and in some sense can be viewed as a centralized version of Gatekeeper. SumUp assumes global knowledge and focuses on scenarios where only $o(n)$ honest nodes are seeking to be labeled as "honest". It uses adaptive maximum flow on the social network to label most of these $o(n)$ honest nodes correctly as "honest", while only labeling $1 + o(1)$ sybil nodes per attack edge as "honest". This is asymptotically better than SybilLimit's $O(\log n)$ guarantee, under SumUp's setting.

4.3 Applying community detection algorithms [25]

The insights in Section 2 and 3.1 are based on a number of related graph properties of $\mathcal{G}$, such as minimum quotient cuts, vertex expansion, edge expansion, conductance, and mixing time. As shown in Strawman-Design 1 in Section 2.2, a sybil defense mechanism could differentiate honest nodes from sybil nodes by directly searching for a subgraph $\mathcal{H}$ having the desired properties (e.g., having no cuts with small quotient or having small mixing time). This will work if the protocol has global knowledge, and if such search is done in a "balanced" way to avoid those pathological scenarios as discussed in Section 2.2.

Consistent with such insights, Viswanath et al. [25] recently realize that existing community detection algorithms [12] can be used to search for $\mathcal{H}$. These algorithms partition $\mathcal{G}$ into multiple "communities", and then all nodes in the local user's "community" are labeled as "honest". While the term community can have a rather rich meaning, from the algorithmic perspective, "communities" are purely defined based on graph structure. For example for many community detection algoirthms, a "community" is simply any subgraph whose total number of internal edges is at least ρ times the total number of edges connecting that subgraph to the rest of the graph. The human user needs to specify and input ρ as a parameter. The number of "communities" found by these algorithms depends on ρ. If ρ is large, the whole social network will only have one community. As ρ decreases, the social network can be partitioned into more and more "communities". Mathematically, the definition of "community" in these algorithms directly translates to high conductance within the "community" and lower conductance among "communities", which in turn relates to minimum quotient cuts, expansion, and mixing time. Thus it is natural that these algorithms can be used to search for $\mathcal{H}$. As a side note, the original SybilGuard paper [30] mentioned a strawman design by searching for minimum quotient cuts. Searching for such cuts is exactly what many community detection algorithms do. There are also some other community detection algoirthms [12] that use random walks instead of directly searching for cuts.

A key drawback of directly applying community detection algorithms for sybil defense is the lack of formal end guarantees on false positives and false negatives. As explained earlier, it is crucial for the search to be done in a "balanced" way. While community detetction algorithms sometimes indeed tend to search in a "balanced" way, there is no guarantee on this. What further complicates the issue is that community detetection algorithms are seldom designed for adversarial cases, while the topology of the sybil region of $\mathcal{G}$ can be adversarially designed by the attacker. A second issue of using community detection algorithms is that it will require global knowledge, since those algorithms are usually designed for centralized settings.

Experimentally, Viswanath et al. [25] show that applying community detection algorithms on several social network datasets achieves similar results in terms of false positives/negatives as other protocols such as SybilLimit and SybilInfer.

4.4 Whanau [16] and Ostra [18]

The Whanau protocol and the Ostra protocol aim at defending against sybil attacks in some specific contexts. Nevertheless, both of them still leverage social networks, as well as the insights from Section 2. Thus we will discuss them below.

Whanau. Whanau utilizes social networks to defend against sybil attacks in one-hop DHTs. Same as SybilLimit, Whanau relies on Assumption 1 and uses random walks. Whanau creates one (virtual) DHT node for each edge in the social network. It also leverages the fact that the stationary distribution on edges is always a uniform distribution, which corresponds to a uniform distribution on the DHT nodes. Each DHT node A uses random walks on the social network to sample other DHT nodes. Those samples are used to fill up A's finger table and successor table. By Theorem 1, if A is a non-escaping node and under proper setting, those samples will largely be DHT nodes corresponding to honest users. Whanau's design is fully decentralized and comes with formal guarantees on the robustness of the DHT against sybil attacks.

While Whanau defends against sybil attacks in a specific context, its functionality cannot be obtained by directly applying a general sybil defense mechanism such as SybilLimit. To be secure, a DHT needs not only the fraction of byzantine nodes to be properly bounded, but also the IDs of the DHT nodes to be randomly chosen. Random IDs will prevent the attacker from targeting its attack to a certain ID region on the DHT ring. Whanau takes care of this issue by using *layered IDs* – see [16] for details. Of course, one could also alternatively design other techniques for random ID generation (e.g., [22]) and then combine with a general sybil defense mechanism to make DHTs secure.

Ostra. Ostra leverages social networks to prevent malicous users from injecting excessive unwanted user-level communication such as email spams, even in the presence of sybil attacks. The basic conceptual idea is that for a node A to communicate with a node B (e.g., sending B an email), A needs to first discover a path from A to B in the social network. Each edge in the social network has a quota. If the user of B later classifies the email as spam, then the quota of each edge on the path is decremented. An edge with zero quota cannot be used in any path. Clearly, if A sends too many email spams to B, A will eventually run out of paths to B. This holds depsite sybil attacks: Once the quotas on the attack edges drop to zero, no sybil nodes can send email spams to honest nodes any more.

Ostra does not formalize the assumptions it makes on the social network, nor does it provide provable end guarantees. In particular, there is no formal guarantee on the impact of the design on misclassifying good communication (i.e., false positives). The complete design of Ostra assumes global knowledge about the social network.[11]

Ostra is specifically designed to prevent unwanted communication instead of as a general sybil defense mechanism. Different from Whanau's functionality, Ostra's functionality could be roughly obtained by directly applying general sybil defenses such as SybilLimit. All one needs to do is to give a communication quota to each node that is labeled as "honest" in SybilLimit. Since only a limited number of sybil nodes will be labeled as "honest", their total quota will be limited as well.

5 Practical Implications and Deployment Considerations

This section explores several practical implications when we would deploy these social-network-based sybil defenses in real systems, as well as some common confusions. These confusions are largely due to SybilGuard/SybilLimit focusing a lot on clean formal guarantees rather than experimental studies. This has made that first set of papers less accessible to the systems community where many of the practical implications are discussed.

5.1 What can be used as a social network for sybil defense?

Getting a social network with strong trust. Sybil defenses via social networks require the social network to have strong trust embedded. Thus SybilLimit, for example, requires each edge in the social network to correspond to an edge key established out-of-band. This however, necessarily incurs non-trivial overhead on the human users. For experimental evaluation and initial deployment, using an online social network such as Facebook would be much easier. But in Facebook, two users may be online friends even if they do not know each other in real life. A further step along this direction is taken when we use social networks formed by users who just share online contents, such as sharing protected journal entries in LiveJournal. Clearly, the trust in all these weak-trust social networks is not sufficient for sybil defense.

One promising way to overcome this issue is perhaps to infer the amount of trust between pairs of users in these weak-trust social networks, based on the quantity and nature of user interactions. For example, if two online friends on Facebook have had video chats, then their mutual trust is likely to be reasonably strong. Alternatively, a likely strong mutual trust can be inferred if they have writing chats regularly or if they have chatted a lot about personal/private topics. After inferring which edges have strong trust, we can then simply use the social network formed by those edges.[12] While it is easy to imagine the effectiveness of this natural approach, a convincing evaluation (especially for writing a paper) would require a large-scale user study.

All social networks are different. Weak-trust social networks can easily have quite different graph properties from strong-trust social networks, and even strong-trust social networks (e.g. DBLP vs. kinship graphs)

[11]Even though the paper [18] alluded to decentralized designs, it did not provide a complete design that is decentralized.

[12]One could even infer *quantitive* trust, and create *multiple* edges between two users depending on the quantitive value.

can be rather different from each other. Given the paucity of strong-trust social network datasets, researchers have used a wide range of different kinds of social networks in their research on this topic. We should always be careful about interpreting the results from all these social networks, since every social network is different. For example, some researchers have posed the question of whether attack edges are indeed hard to establish [25], given that edges are easy to form in online social networks [2]. Such questions are perhaps best asked against specific social networks. Similarly, the questions in the next section should perhaps also be answered with respect to individual social networks. It is unlikely that fixed rigid answers apply to *all* possible social networks.

5.2 Do social networks really have small mixing time?

While Assumption 1 (i.e., social networks having small mixing time) has some supporting theoretical evidences, it is natural to ask whether it indeed holds in practice. It has been shown [28] that SybilLimit works well on Friendster, LiveJournal, and DBLP datasets. Later based on additional social network datasets, some research efforts [16, 23, 24] support this assumption, while others [19, 25] rebut this assumption. Interestingly, the two camps of efforts have even used some common datasets. Before going further, one should keep in mind that strictly speaking, it does not make sense to ask whether social networks have small mixing time. Exactly how small is small? Sybil defenses via social networks often offer an end guarantee degrading linearly with the mixing time, and there does not exist a "threshold" breaking point. Thus the real question to ask is what the mixing time is, quantitatively.

Removing low-degree nodes? Both camps of research efforts [16, 19, 23, 24, 25, 28] have used real social network datasets. A major factor leading to the different conclusions is whether low-degree nodes are removed from the datasets. Experiments supporting Assumption 1 usually remove those low-degree nodes, while experiments rebutting it do not. For example, the *pre-processing* procedure in SybilLimit removes all nodes with degree below 5 from the social network. There is general consensus that removing these low-degree nodes reduces the mixing time of these datasets significantly. It is tempting to assert that removing these nodes "manipulates" the datasets and thus yields wrong results. Such a claim can be further supported by the fact that in certain low-degree graphs, removing nodes with degree below 5 can reduce the graph size by more than half. This is quite bad on the surface, since as argued in [19], a majority of the nodes would be "denied joining the service".

Interestingly, removing low-degree nodes was first introduced in SybilLimit with a proper usage scenario in mind. This usage scenario has unfortunately been largely missed in the arguments from *both* camps of research efforts, perhaps because the scenario was only described in SybilLimit's technical report [27]. Specifically, SybilLimit expects a new user to establish a minimum number of edges (e.g., 5) with existing users. By imposing this minor requirement on the users, SybilLimit essentially slightly "influences" the social network so that there are no nodes with low degree.[13] A further point is that even those users with degree below 5 (called *trial users*) are not "denied joining the service". A trial user can still properly label other nodes — it only needs to find a single regular user whom it trusts and then simply adopts that regular user's labels on other nodes. Being able to label other nodes is sufficient for the trial user to receive good service, such as finding honest nodes to back up its data in a peer-to-peer backup system or verifying the votes cast by other nodes in a voting system. The only issue is that a trial user will not be properly labeled as "honest" by other nodes and thus cannot *contribute* by for example, providing backup space or casting a

[13]Note that a user still has disincentives to establish edges with malicious nodes to meet this requirement, since users near the attack edges may not be protected by SybilLimit any more.

vote[14].

In summary, imposing some minor user requirements will eliminate low-degree nodes in practice, which in turn often reduces the mixing time significantly. This will enable these sybil defenses to offer stronger guarantees. Of course, it would be even better if we can obtain stronger guarantees without such user requirements.

Do we really need small mixing time? Given the discussion on social networks' mixing time in practice, a related question should be kept in mind: Do we really need small mixing time? In their pursuit of theoretical elegance, SybilGuard and SybilLimit made this clean assumption on mixing time, which has been inherited by most later efforts. On the other hand, mixing time is a worst-case measure. It describes the needed length of a random walk starting from the worst starting node to mix well into even the worst region of the graph. Given that sybil defenses allow false positives and false negatives anyway, such a clean but worst-case assumption may not be necessary, and can probably be replaced with weaker but less clean assumptions.

For example, the recently proposed concept of expected conductance [13] allows one to capture the "average" conductance of a graph. It might be possible for these sybil defenses to only rely on assumptions regarding such "average" conductance or "average" mixing time of the social network. As another example, Leskovec et al. [15] show that the conductance of social networks they studied (without removing low-degree nodes) is often rather small and is often around the order of magnitude of 0.01 or 0.001. But they also observe that such small conductance is usually caused by small-sized *whiskers* (of around 100 nodes) each having one edge connecting it to a *core*. The core can have for example, 50% of all the nodes. The mixing time is large since the random walk needs to mix well into every whisker. On the other hand, SybilLimit might only need the random walks to mix well into the core.

5.3 Will targeted sybil attacks [25] break these defenses in practice?

Via extensive experiments, Viswanath et al. [25] have recently identified in practice a *targeted sybil attack* where the attack edges concentrate near a given honest node V. They show that such attack causes many false positives and false negatives in the labels that V assigns to other nodes. In turn, they suggest that such a targeted attack reveals a fundamental limitation of sybil defenses via social networks.

As discussed in Section 3.2.1, honest nodes near the attack edges are fundamentally handicapped since the sybil region may look more like "home" to them. Thus it is unlikely that any defense mechanism can protect *all* honest nodes. One can however, provide nice provable guarantees to $(1-\epsilon)$ fraction of the honest nodes, as in Theorem 1 and 2. Note that these theorems hold even if the locations of the attack edges are adversarial — their locations will only affect which nodes belong to the $(1-\epsilon)$ fraction.

It is perhaps clear at this point that Viswanath et al.'s conclusion stems from only looking at the guarantees for that handicapped node V. A targeted sybil attack can always cause problem to a *given* honest node V and make V handicapped, but it is not possible to simultaneously target all honest nodes (otherwise the number of attack edges is simply not fixed any more). The defense mechanism still protects $(1-\epsilon)$ fraction of the honest nodes, except that V is not in that fraction. This also highlights the critical importance for each honest node to assign its own labels to other nodes, as in SybilLimit. Targeted sybil attacks would have succeeded if the system simply picked a few trusted honest nodes, and had all nodes adopt the labels assigned by those few trusted nodes.

[14]Usually in distributed systems, casting a vote helps other users instead of helping the user casting the vote. This is rather different from for example, democratic election.

6 Sybil Defenses Not Leveraging Social Networks

There are also a number of interesting sybil defenses proposed recently which do not leverage social networks. We only intend to provide a concise list here. In the following, only the first effort is for general sybil defense, and all others are for sybil defenses in specific contexts.

- Bazzi and Konjevod [1] propose a very interesting and elegant sybil defense mechanism based on network coordinates. Their approach offers provable guarantees under certain assumption on the network position of the attacker.
- DSybil [31] uses user feedbacks to defend against sybil attacks in recommendation systems.
- DCast [29] uses *pairwise entry fees* to defend against sybil attacks in overlay multicast with rational players.
- Danezis et al. [7] propose exploiting the *bootstrap tree* of the DHT for defending against sybil attacks in DHT routing.
- Cheng and Friedman [5] investigate how to prevent sybil nodes from boosting the attacker's reputation in reputation systems. Their effort can be viewed as a substantial generalization of Feldman et al.'s work [10] for sybil defense in the same context, which uses max-flow in the reputation network.
- Sybil defenses have also been explored in sensor networks [20, 21].

7 Conclusions and Future Directions

This article has provided a tutorial on sybil defense mechanisms via social networks, which have attracted much attention in the past 5 years. We have also surveyed and discussed a number of interesting designs in this area, as well as some practical implications. Interesting future research directions on this topic include:

- On the practical side, it would be interesting to push these designs into real systems and real deployments. A critical step here is to find an appropriate social network to use, and we feel that inferring trust based on user interactions (Section 5.1) can be quite promising.
- On the theoretical side, SybilLimit is already near-optimal if we leverage only mixing time. But there could be other or additional graph properties that one can leverage. The desirable property should be significantly affected by a sybil attack and there should be no easy way for the attacker to avoid affecting that property. From a theoretical perspective, it would be very interesting to design good algorithms to leverage such a property, while providing formal guarantees.
- Beyond sybil attacks and beyond social networks, the insights on attack edges, cuts, and mixing time may find applications elsewhere. For example, these insights might apply to PageRank and help it to be robust against sybil webpages [6]. One could also imagine detecting email spams based on the email graph and its connectivity property.

8 Acknowledgments

I thank all my collaborators on our sybil attack related research efforts, including Abraham Flaxman, Phillip B. Gibbons, Michael Kaminsky, Chenwei Shi, and Feng Xiao. I thank Binbin Chen for inspiring discussions and many helpful comments on this article. Finally, I thank Idit Keidar for inviting me to write this article for the Distributed Computing Column, as well as for her many helpful comments on this article.

References

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

[2] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *WWW*, Apr. 2009.

[3] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine Resilient Random Membership Sampling. *Computer Networks*, 53(13), 2009.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *IEEE INFOCOM*, 2005.

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

[6] A. Cheng and E. Friedman. Manipulability of PageRank under Sybil Strategies. In *Proceedings of the First Workshop of Networked Systems*, 2006.

[7] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant DHT routing. In *ESORICS*, 2005. Springer-Verlag LNCS 3679.

[8] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

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

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

[11] A. Flaxman. Expansion and Lack Thereof in Randomly Perturbed Graphs. In *International Workshop on Algorithms and Models for the Web-Graph*, 2006.

[12] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5), 2010.

[13] M. Gurevich and I. Keidar. Correctness of Gossip-Based Membership under Message Loss. *SIAM Journal on Computing*, 39(8), 2010.

[14] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *ACM STOC*, 2000.

[15] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, 2008.

[16] C. Lesniewski-Laas and M. F. Kaashoek. Whanau: A Sybil-proof Distributed Hash Table. In *NSDI*, Apr. 2010.

[17] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the Maze P2P file-sharing system. In *IEEE ICDCS*, 2007.

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

[19] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *IMC*, 2010.

[20] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: Analysis & defenses. In *ACM/IEEE IPSN*, 2004.

[21] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE S & P*, 2005.

[22] C. Scheideler. How to spread adversarial nodes? Rotate! In *STOC*, 2005.

[23] N. Tran, J. Li, L. Subramanian, and S. Chow. Optimal Sybil-resilient Node Admission Control. In *INFOCOM*, Apr. 2011.

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

[25] B. Viswanath, A. Post, K. Gummadi, and A. Mislove. An Analysis of Social Network-based Sybil Defenses. In *SIGCOMM*, August 2010.

[26] M. Yang, Z. Zhang, X. Li, and Y. Dai. An empirical study of free-riding behavior in the Maze P2P file-sharing system. In *IPTPS*, 2005.

[27] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. Technical Report TRA2/08, National University of Singapore, School of Computing, March 2008. http://www.comp.nus.edu.sg/~yuhf/sybillimit-tr.pdf.

[28] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. *IEEE/ACM Transactions on Networking*, June 2010. Preliminary version appeared in IEEE Symposium on Security and Privacy 2008.

[29] H. Yu, P. B. Gibbons, and C. Shi. Brief Announcement: Sustaining Collaboration in Multicast despite Rational Collusion. In *PODC*, 2011.

[30] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. *IEEE/ACM Transactions on Networking*, June 2008. Preliminary version appeared in ACM SIGCOMM Conference 2006.

[31] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal Sybil-Resistance for Recommendation Systems. In *IEEE Symposium on Security and Privacy*, May 2009.

A Bounding the Number of Tainted Tails in SybilLimit

This section explains how SybilLimit uses *random routes* to bound the total number of tainted tails, which was deferred from Section 3.2.3.

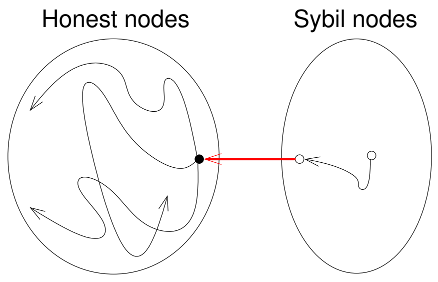

Figure 3: Getting unlimited number of tainted tails.

Secure random walks. To begin with, SybilLimit needs to prevent sybil nodes from claiming arbitrary edges in the honest region as their tails. This is relatively easy to achieve. When a node initiates a random walk from itself, it includes its identity as well as a non-negative hop count in the message. This message propagates along the random walk, with the hop count incrementing at each hop. At each hop, the message is authenticated using the corresponding edge key, so that a node only accepts message from its neighbors. Let node A be the node receiving the message with a hop count of t, along a certain incoming edge ζ. A now knows that ζ is the tail of that random walk. A then locally binds ζ to the identity included in the message. Later A will vouch for that identity when ζ is used as the common tail for intersection.

If the random walk started from an honest node and is non-escaping, then all nodes along the walk obey the above protocol. If the random walk is instead escaping, then A may be a sybil node. But as explained in Section 3.2.3, we will not need the help from this escaping random walk to ensure tail intersection anyway. The last possibility is that the random walk started from a sybil node. If this walk ever crosses the attack edge and enters the honest region, it may result in a tainted tail. SybilLimit leverages the fact that once this random walk crosses the attack edge, it will be in the hands of good nodes and thus will have properly maintained hop counts. The honest nodes can also easily ensure that the hop count of the random walk is non-negative when it first crosses the attack edge, by dropping all random walks with negative hop counts and without knowing which edges are attack edges. All these will guarantee that the honest region portion of this random walk has a length of at most t.

Now a sybil node can only claim an edge ζ in the honest region as its tail, if there exists a random walk that starts from some attack edge and reaches ζ within t hops. This by itself still does not yet bound the total number of tainted tails, since a sybil node can initiate unlimited number of random walks, and each random walk may result in a different tainted tail (Figure 3). To solve this problem, we first describe a simple quota-based strawman design that is easier to understand than SybilLimit's design:

StrawmanDesign 3. *Each edge in the graph enforces a quota on the total number of times that the edge can be crossed by all random walks collectively. Once the quota is reached, random walks that want to traverse that edge will simply be dropped.*

To see why StrawmanDesign 3 works, imagine that each node in the social network performs a random walk of length t starting from itself. Consider any given directed edge in the graph (where we treat each undirected edge as two directed edges). Each of the random walks may traverse this edge for 0, 1, or multiple times. One can show that the total number of times that all the random walks traverse this edge is $O(t \log t)$ with probability at least $1 - O(1/t)$.[15] Now since each node actually does $\theta(\sqrt{m})$ random walks in SybilLimit, we will limit the total number of random walks that cross any given edge within

[15] More specifically, let X denote the total number of traversal times. It can be shown that $E[X] \leq t$ and $\Pr[X > O(t \log t)] < \delta/t$ for any given $\delta > 0$. Note that different hops in each random walk are correlated, but still the above property can be obtained by reasoning about X_i's, which is the total number of times that all the random walks traverses the edge at their ith hops.

$O(\sqrt{m} \cdot t \log t)$. Doing so will not hurt honest nodes much since the probability that a uniformly random honest node's random walk encountering a full quota along the way is only δ. On the other hand, the total number of tainted tails will be bounded within $O(\sqrt{m} \cdot gt \log t)$, since each attack edge has a quota of $O(\sqrt{m} \cdot t \log t)$.

SybilLimit's design based on random routes. Instead of enforcing a quota, SybilLimit uses *random routes* in place of random walks to bound the number of tailed tails within $O(\sqrt{m} \cdot gt)$, which is better than $O(\sqrt{m} \cdot gt \log t)$. When a random walk reaches a node, the node flips a coin on the fly to choose a uniformly random edge to continue the walk. For doing *random routes*, each node instead maintains a *randomized routing table*, which is initiated at start-up time and remains fixed until the neighbors of the node change. This routing table is a uniformly random one-to-one mapping between incoming edges and outgoing edges. When a random route reaches a node, the node looks up the routing table to determine where the route should continue. Since each node in SybilLimit will need to do $\theta(\sqrt{m})$ random routes, these random routes will be done in $\theta(\sqrt{m})$ independent *instances*, where a node maintains an independent route table for each instance.

These routing tables introduce strong correlation among multiple random routes in a given instance — if two random routes in a given instance ever cross the same edge, they merge and stay together for ever. This means that in Figure 3, in a given instance, all the random routes crossing the attack edge will merge and stay merged. It is thus not hard to see that now each attack edge can result in at most t tainted tails in a given instance. Those tainted tails are essentially the tails of t overlapping random routes that traverse 1, 2, . . ., t honest nodes, respectively. With g attack edges and $\theta(\sqrt{m})$ instances, the total number of tainted tails is $O(\sqrt{m} \cdot gt)$.

SybilLimit still wants individual random routes to have the same probabilistic behavior as individual random walks, so that the properties on the tails of the honest nodes still hold. Clearly, if a random route does not encounter any node more than once, its behavior is already the same as a random walk. If a random route encounters some node more than once, that node will use additional independent routing tables for those extra routing decisions, so that the random route will have the same probabilistic behavior as individual random walks.

One may wonder whether the earlier arguments on $O(\sqrt{m} \cdot gt)$ tainted tails still hold with all these additional routing tables. Consider the example in Figure 3, and let A and B be the honest node and sybil node incidental to the attack edge, respectively. Imagine that a random route crosses the attack edge "$B \rightarrow A$" at hop count $h_1 \geq 0$. If that random route always remains in the honest region, then our previous arguments clearly still hold. But if it returns to the sybil region at hop count h_2 ($h_2 > h_1$), then it may come back to B and then to A at hop count h_3, forcing A to use a second independent routing table this time. Note that here h_1 and h_3 may be manipulated by the sybil nodes. But the honest nodes will determine the value of $(h_2 - h_1)$, which corresponds to the number of edges (i.e., tainted tailed) in the honest region that the random route traverses between its two encounters with A.

Assume for now that the random route does not encounter A for a third time. Then as long as $h_3 \geq h_2$, the number of tainted tails will be at most $(t - h_3) + (h_2 - h_1) \leq t$. To ensure $h_3 \geq h_2$, A first obtains an upper bound h'_2 on h_2. This is done by having the nodes on the random route propagate backward along the route the maximum hop count h'_2 they see, before the random route encounters A a second time. The sybil nodes can never bring down h'_2 below h_2, since the hop count of h_2 has already appeared before the random route goes back to the sybil region. Now A simply requires that $h_3 \geq h'_2$, otherwise it drops the random route. Note that if the random route has only traversed honest nodes, then h_3 will equal h'_2. It is easy to generalize the above arguments to cases where the random route encounters A more than two times.