SybilInfer: Detecting Sybil Nodes using Social Networks

George Danezis
Microsoft Research,
Cambridge, UK.
gdane@microsoft.com

Prateek Mittal
University of Illinois at Urbana-Champaign,
Illinois, USA.
mittal2@uiuc.edu

Abstract

SybilInfer is an algorithm for labelling nodes in a social network as honest users or Sybils controlled by an adversary. At the heart of SybilInfer lies a probabilistic model of honest social networks, and an inference engine that returns potential regions of dishonest nodes. The Bayesian inference approach to Sybil detection comes with the advantage label has an assigned probability, indicating its degree of certainty. We prove through analytical results as well as experiments on simulated and real-world network topologies that, given standard constraints on the adversary, SybilInfer is secure, in that it successfully distinguishes between honest and dishonest nodes and is not susceptible to manipulation by the adversary. Furthermore, our results show that SybilInfer outperforms state of the art algorithms, both in being more widely applicable, as well as providing vastly more accurate results.

1 Introduction

The Peer-to-peer paradigm allows cooperating users to enjoy a service with little or no need for any centralised infrastructure. While communication [24] and storage [4] systems employing this design philosophy have been proposed, the lack of any centralised control over identities opens such systems to Sybil attacks [8]: a few malicious nodes can simulate the presence of a very large number of nodes, to take over or disrupt key functions of the distributed or peer-to-peer system. Any attempt to build fault-tolerance mechanisms is doomed since adversaries can control arbitrary fractions of the system nodes. This Sybil attack is further made practical through the use of the existing large number of compromised networked machines (often called *zombies*) being part of bot-nets.

Similar problems plague Web 2.0 applications that rely on collaborative tagging, filtering and editing, like Wikipedia [31], or `del.icio.us` [28]. A single user can register under different pseudonyms, and bypass any elections of velocity check mechanism that attempts to guarantee the quality of the data through the plurality of contributors. On-line forums, starting with USENET [19], to contemporary blogs or virtual worlds like Second Life [22] always have to deal with the issue of disruption in the discussion threads, with persistent abusers coming back under different names. All these are forms of Sybil attacks at the high-level application layers.

There are two schools of Sybil defence mechanisms, the *centralised* and *decentralised* ones. Centralised defences assume the existence of an authority that is capable of doing admission control for the network [3]. Its role is to rate limit the introduction of 'fake' identities, to ensure that the fraction of corrupt nodes remains under a certain threshold. The practicalities of running such an authority are very system-specific and in general it would have to act as a Public Key Certification Authority as well as a guardian of the moral standing of the nodes introduced – a very difficult problem in practice. Such centralised solutions are also at odds with the decentralisation guiding principle of peer-to-peer systems.

Decentralised approaches recognise the difficulty in having a single authority vouching for nodes, and distribute this task across all nodes of the system. The first such proposal is Advogato [14], which aimed to reduce abuse in on-line services, followed by a proposal to use introduction graphs of Distributed Hash Tables [6] to limit the potential for routing disruption in those systems. The state of the art SybilGuard [27] and SybiLimit [26] propose the use of social networks to mitigate Sybil attacks. As we will see, SybilGuard suffers from high false negatives, while SybilLimit makes unrealistic assumptions about the knowledge of number of honest nodes in the network. In both cases the systems Sybil detection strategies are based on heuristics that are not optimal.

Our key contribution is to propose SybilInfer, a method for detecting Sybil nodes in a social network, that makes use of all information available to the defenders. The formal model underlying our approach casts the problem of detecting Sybil nodes in the context of Bayesian Inference: given a set of stated relationships between nodes, the task is to label nodes as honest or dishonest. Based on some simple and generic assumptions, like the fact that social networks are fast mixing [18], we sample cuts in

the social graph according to the probability they divide it into honest and dishonest regions. These samples not only allow us to label nodes as honest or Sybil attackers, but also to associate with each label output by our algorithm a degree of certainty.

The proposed techniques can be applied in a wide variety of settings where high reliability peer-to-peer systems, or Sybil-resistant collaborative mechanisms, are required even under attack:

- Secure routing in Distributed Hash Tables motivated early research into this field, and our proposal can be used instead of a centralised authority to limit the fraction of dishonest nodes, that could disrupt routing [3].
- In public anonymous communication networks, such as Tor [7], our techniques can be used to eliminate the potential for a single entity introducing a large number of nodes, and de-anonymize users' circuits. This was so far a key open problem for securely scaling such systems.
- Leader Election [2] and Byzantine agreement [12] mechanisms that were rendered useless by the Sybil attack can again be of use, after Sybil nodes have been detected and eliminated from the social graph.
- Finally, detecting Sybil accounts is a key step in preventing false email accounts used for spam, or preventing trolling and abuse of on-line communities and web-forums. Our techniques can be applied in all those settings, to fight spam and abuse [14].

SybilInfer applies to settings where a peer-to-peer or distributed system is somehow based on or aware of social connections between users. Properties of natural social graphs are used to classify nodes are honest or Sybils. While this approach might not be applicable to very traditional peer-to-peer systems [24], it is more an more common for designers to make distributed systems aware of the social environment of their users. Third party social network services [29, 30], can also be used to extract social information to protect systems against sybil attacks using SybilInfer. Section 5 details deployment strategies for SybilInfer and how it is applicable to current systems.

We show analytically that SybilInfer is, from a theoretical perspective, very powerful: under ideal circumstances an adversary gains no advantage by introducing into the social network any additional Sybil nodes that are not 'naturally' connected to the rest of the social structure. Even linking all dishonest nodes with each other (without adding any Sybils) changes the characteristics of their social sub-graph, and can under some circumstances be detected. We demonstrate the practical efficacy of our approach using both synthetic scale-free topologies as well as real-world LiveJournal data. We show very significant security improvements over both SybilGuard and SybilLimit, the current state of the art Sybil defence mechanisms. We also propose extensions that enable our solution to be implemented in decentralised settings.

This paper is organised in the following fashion: in section 2 we present an overview of our approach that can be used as a road-map to the technical sections. In section 3 we present our security assumptions, threat model, the probabilistic model and sampler underpinning SybilInfer; a security evaluation follows in section 4, providing analytical as well as experimental arguments supporting the security of the method proposed along with a comparison with SybilInfer. Section 5 discusses the settings in which SybilInfer can be fruitfully used, followed by some conclusions in section 6.

2 Overview

The SybilInfer algorithm takes as an input a social graph G and a single known good node that is part of this graph. The following conceptual steps are then applied to return the probability each node is honest or controlled by a Sybil attacker:

- A set of traces T are generated and stored by performing special random walks over the social graph G. These are the only information retained about the graph for the rest of the SybilInfer algorithm, and their generation is detailed in section 3.1.
- A probabilistic model is then defined that describes the likelihood a trace T was generated by a specific honest set of nodes within G, called X. This model is based on our assumptions that social networks are fast mixing, while the transitions to dishonest regions are slow. Given the probabilistic model, the traces T and the set of honest nodes we are able to calculate $\Pr[T|X \text{ is honest}]$. The calculation of this quantity is the subject of section 3.1 and section 3.2.
- Once the probabilistic model is defined, we use Bayes theorem to calculate for any set of nodes X and the generated trace T, the probability that X consists of honest nodes. Mathematically this quality is defined as $\Pr[X \text{ is honest}|T]$. The use of Bayes theorem is described in section 3.1.
- Since it is not possible to simply enumerate all subsets of nodes X of the graph G, we instead sample from the distribution of honest node sets X, to only get a few $X_0, \ldots, X_N \sim \Pr[X \text{ is honest}|T]$. Using those representative sample sets of honest nodes, we can calculate the probability any node in the system is honest or dishonest. Sampling and the approximation of the sought marginal probabilities are the subject of section 3.3.

The key conceptual difficulty of our approach is the definition of the probabilistic model over the traces T, and

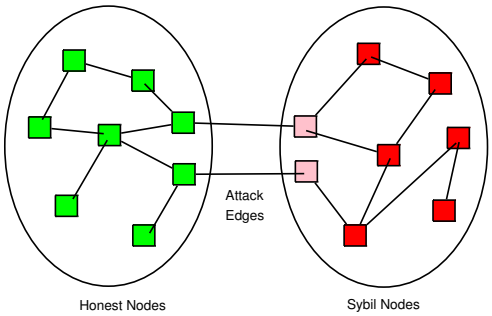


Figure 1. Illustration of Honest nodes, Sybil nodes and attack edges between them.

its inversion using Bayes theorem to define a probability distribution over all possible honest sets of nodes X. This distribution describes the likelihood that a specific set of nodes is honest. The key technical challenge is making use of this distribution to extract the sought probability each node is honest or dishonest, that we achieve via sampling. Section 3, describes in some detail how these issues are tackled by SybilInfer.

3 Model & Algorithm

Let us denote the social network topology as a graph G comprising vertices V, representing people and edges E, representing trust relationships between people. We consider the friendship relationship to be an undirected edge in the graph G. Such an edge indicates that two nodes trust each other to not be part of a Sybil attack. Furthermore, we denote the friendship relationship between an attacker node and an honest node as an *attack edge* and the honest node connected to an attacker node as a *naive node* or *misguided node*. Different types of nodes are illustrated in figure 1. These relationships must be understood by users as having security implications, to restrict the promiscuous behaviour often observed in current social networks, where users often flag strangers as their friends [23].

We build our Sybil defence around the following assumptions:

1. At least one honest node in the network is known. In practise, each node trying to detect Sybil nodes can use itself as the apriori honest node. This assumption is necessary to break symmetry: otherwise an attacker could simply mirror the honest social structure, and any detector would not be able to distinguish which of the two regions is the honest one.

2. Social networks are fast mixing: this means that a random walk on the social graph converges quickly to a node following the stationary distribution of the graph. Several authors have shown that real-life social networks are indeed fast mixing [16, 18].

3. A node knows the complete social network topology (G) : social network topologies are relatively static, and it is feasible to obtain a global snapshot of the network. Friendship relationships are already public data for popular social networks like Facebook [29] and Orkut [30]. This assumption can be relaxed to using sub-graphs, making SybilInfer applicable to decentralised settings.

Assumptions (1) and (2) are identical to those made by the SybilGuard and SybilInfer systems. Previously, the authors of SybilGuard [27] observed that when the adversary creates too many Sybil nodes, then the graph G has a small cut: a set of edges that together have small stationary probability and whose removal disconnects the graph into two large sub-graphs.

This intuition can be pushed much further to build superior Sybil defences. It has been shown [20] that the presence of a small cut in a graph results in slow mixing which means that fast mixing implies the absence of small cuts. Applied to social graphs this observation underpins the key intuition behind our Sybil defence mechanism: the *mixing between honest nodes in the social networks is fast, while the mixing between honest nodes and dishonest nodes is slow*. Thus, computing the set of honest nodes in the graph is related to computing the bottleneck cut of the graph.

One way of formalising the notion of a bottleneck cut, is in terms of *graph conductance (Φ)* [11], defined as:

$$\Phi = \min_{X \subset V : \pi(X) < 1/2} \Phi_X,$$

where Φ_X is defined as:

$$\Phi_X = \frac{\Sigma_{x \in X} \Sigma_{y \notin X} \pi(x) P_{xy}}{\pi(X)},$$

and $\pi(\cdot)$ is the stationary distribution of the graph. Intuitively for any subset of vertices $X \subset V$ its conductance Φ_X represents the probability of going from X to the rest of the graph, normalised by the probability weight of being on X. When the value is minimal the bottleneck cut in the graph is detected.

Note that performing a brute force search for this bottleneck cut is computationally infeasible (it is actually NP-Hard, given its relationship to the sparse-cut problem). Furthermore, finding the exact smallest cut is not as important as being able to judge how likely any cut is, to be dividing nodes into an honest and dishonest region. This probability is related to the deviation of the size of any cut from what we would expect in a natural, fast mixing, social network.

3.1 Inferring honest sets

In this paper, we propose a framework based on Bayesian inference to detect approximate cuts between honest and Sybil node regions in a social graph and use those to infer the labels of each node. A key strength of our approach is that it, not only associates labels to each node, but also finds the correct probability of error that could be used by peer-to-peer or distributed applications to select nodes.

The first step of SybilInfer is the generation of a set of random walks on the social graph G. These walks are generated by performing a number s of random walks, starting from each node in the graph (i.e. a total of $s \cdot |V|$ walks.) A special probability transition matrix is used, defined as follows:

$$P_{ij} = \begin{cases} \min(\frac{1}{d_i}, \frac{1}{d_j}) & \text{if } i \to j \text{ is an edge in G} \\ 0 & \text{otherwise} \end{cases},$$

where d_i denotes the degree of vertex i in G.

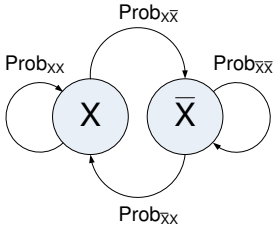
This choice of transition probabilities ensures that the stationary distribution of the random walk is uniform over all vertices $|V|$. The length of the random walks is $l = O(\log |V|)$, which is rather short, while the number of random walks per node (denoted by s) is a tunable parameter of the model. Only the starting vertex and the ending vertex of each random walk are used by the algorithm, and we denote this set of vertex-pairs, also called the *traces*, by T.

Now consider any cut $X \subset V$ of nodes in the graph, such that the a-prior honest node is an element of X. We are interested in the probability that the vertices in set X are all honest nodes, given our set of traces T, i.e. $P(X = \text{Honest}|T)$. Through the application of Bayes theorem we have an expression of this probability:

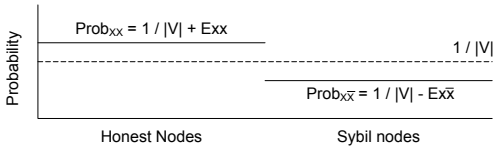
$$P(X = \text{Honest}|T) = \frac{P(T|X = \text{Honest}) \cdot P(X = \text{Honest})}{Z},$$

where Z is the normalization constant given by: $Z = \Sigma_{X \subset V} P(T|X = \text{Honest}) \cdot P(X = \text{Honest})$. Note that Z is difficult to compute because it involves the summation of an exponential number of terms in the size of $|V|$. Only being able to compute this probability up to a multiplicative constant Z is not an impediment. The a-prior distribution $P(X = \text{Honest})$ can be used to encode any further knowledge about the honest nodes, or can simply be set to be uniform over all possible cuts.

Bayes theorem has allowed us to reduce the initial problem of inferring the set of good nodes X from the set of traces T, to simply being able to assign a probability to each set of traces T given a set of honest nodes X, i.e. calculating $P(T|X = \text{Honest})$. Our only remaining theoretical task is deriving this probability, given our model's assumptions.



(a) A schematic representation of transition probabilities between honest X and dishonest $\bar{X}$ regions of the social network.



(b) The model of the probability a short random walk of length $O(\log |V|)$ starting at an honest node ends on a particular honest or dishonest (Sybil) node. If no Sybils are present the network is fast mixing and the probability converges to $1/|V|$, otherwise it is biased towards landing on an honest node.

Figure 2. Illustrations of the SybilInfer models

Note that since the set X is honest, we assume (by assumption (2)) fast mixing amongst its elements, meaning that a short random walk reaches any element of the subset X uniformly at random. On the other hand a random walk starting in X is less likely to end up in the dishonest region $\bar{X}$, since there should be an abnormally small cut between them. (This intuition is illustrated in figure 2(a).) Therefore we approximate the probability that a short random walk of length $l = O(\log |V|)$ starts in X and ends at a particular node in X is given by $\text{Prob}_{XX} = \Pi + E_{XX}$, where Π is the stationary distribution given by $1/|V|$, for some $E_{XX} > 0$. Similarly, we approximate the probability that a random walk starts in X and does not end in X is given by $\text{Prob}_{X\bar{X}} = \Pi - E_{X\bar{X}}$. Notice that $\text{Prob}_{XX} > \text{Prob}_{X\bar{X}}$, which captures the property that there is fast mixing amongst honest nodes and slow mixing between honest and dishonest nodes. The approximate probabilities Prob_{XX} and $\text{Prob}_{X\bar{X}}$ and their likely gap from the ideal $1/|V|$ are illustrated in figure 2(b).

Let N_{XX} be the number of traces in T starting in the honest set X and ending in same honest set X. Let $N_{X\bar{X}}$ be the number of random walks that start at the honest set X and end in the dishonest set $\bar{X}$. $N_{\bar{X}\bar{X}}$ and $N_{\bar{X}X}$ are defined similarly. Given the approximate probabilities of transitions from one set to the other and the counts of such

transitions we can ascribe a probability to the trace:

$$P(T|X = \text{Honest}) = (\text{Prob}_{XX})^{N_{XX}} \cdot (\text{Prob}_{X\bar{X}})^{N_{X\bar{X}}} \cdot (\text{Prob}_{\bar{X}\bar{X}})^{N_{\bar{X}\bar{X}}} \cdot (\text{Prob}_{\bar{X}X})^{N_{\bar{X}X}},$$

where $\text{Prob}_{\bar{X}\bar{X}}$ and $\text{Prob}_{\bar{X}X}$ are the probabilities a walk starting in the dishonest region ends in the dishonest or honest regions respectively.

The model described by $P(T|X = \text{Honest})$ is an approximation to reality that is suitable enough to perform Sybil detection. It is of course unlikely that a random walk starting at an honest node will have a uniform probability to land on all honest or dishonest nodes respectively. Yet this simple probabilistic model relating the starting and ending nodes of traces is rich enough to capture the "probability gap" between landing on an honest or dishonest node, as illustrated in figure 2(b), and suitable for Sybil detection.

3.2 Approximating E_{XX}

We have reduced the problem of calculating $P(T|X = \text{Honest})$ to finding a suitable E_{XX}, representing the 'gap' between the case when the full graph is fast mixing (for $E_{XX} = 0$) and when there is a distinctive Sybil attack (in which case $E_{XX} >> 0$.)

One approach could be to try inferring E_{XX} through a trivial modification of our analysis to co-estimate $P(X = \text{Honest}, E_{XX}|T)$. Another possibility is to approximate E_{XX} or Prob_{XX} directly, by choosing the most likely candidate value for each configuration of honest nodes X considered. This can be done through the conductance or through sampling random walks on the social graph.

Given the full graph G, Prob_{XX} can be approximated as $\text{Prob}_{XX} = \frac{\Sigma_{x\in X}\Sigma_{y\in X}\Pi(x)P^l_{xy}}{\Pi(X)}$, where P^l_{xy} is the probability that a random walk of length l starting at x ends in y. This approximation is very closely related to the conductance of the set X and $\bar{X}$. Yet computing this measure would require some effort.

Notice that Prob_{XX}, as calculated above, can also be approximated by performing many random walks of length l starting at X and computing the fraction of those walks that end in X. Interestingly our traces already contain random walks over the graph of exactly the appropriate length, and therefore we can reuse them to estimate a good Prob_{XX} and related probabilities. Given the counts $N_{XX}, N_{X\bar{X}}, N_{\bar{X}X}$ and $N_{\bar{X}\bar{X}}$:

$$\text{Prob}_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|}$$

and

$$\text{Prob}_{\bar{X}\bar{X}} = \frac{N_{\bar{X}\bar{X}}}{N_{\bar{X}\bar{X}} + N_{\bar{X}X}} \cdot \frac{1}{|\bar{X}|},$$

and, $\text{Prob}_{X\bar{X}} = 1 - \text{Prob}_{XX}$ and $\text{Prob}_{\bar{X}X} = 1 - \text{Prob}_{\bar{X}\bar{X}}$.

Approximating Prob_{XX} through the traces T provides us with a simple expression for the sought probability, based simply on the number of walks starting in one region and ending in another:

$$\begin{aligned} P(T|X = \text{Honest}) \quad = \quad & (\frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|})^{N_{XX}} \cdot \\ & (\frac{N_{X\bar{X}}}{N_{X\bar{X}} + N_{XX}} \cdot \frac{1}{|\bar{X}|})^{N_{X\bar{X}}} \cdot \\ & (\frac{N_{\bar{X}\bar{X}}}{N_{\bar{X}\bar{X}} + N_{\bar{X}X}} \cdot \frac{1}{|\bar{X}|})^{N_{\bar{X}\bar{X}}} \cdot \\ & (\frac{N_{\bar{X}X}}{N_{\bar{X}X} + N_{\bar{X}\bar{X}}} \cdot \frac{1}{|X|})^{N_{\bar{X}X}}, \end{aligned}$$

This expression concludes the definition of our probabilistic model, and contains only quantities that can be extracted from either the known set of nodes X, or the set of traces T that is assigned a probability. Note that we do not assume any prior knowledge of the size of the honest set, and it is simply a variable $|X|$ or $|\bar{X}|$ of the model. Next, we shall describe how to sample from the distribution $P(X = \text{Honest}|T)$ using the Metropolis-Hastings algorithm.

3.3 Sampling honest configurations

At the heart of our Sybil detection techniques lies a model that assigns a probability to each sub-set of nodes of being honest. This probability $P(X = \text{Honest}|T)$ can be calculated up to a constant multiplicative factor Z, that is not easily computable. Hence, instead of directly calculating this probability for any configuration of nodes X, we will attempt instead to sample configurations X_i following this distribution. Those samples are used to estimate the marginal probability that any specific node, or collections of nodes, are honest or Sybil attackers.

Our sampler for $P(X = \text{Honest}|T)$ is based on the established Metropolis-Hastings algorithm [10] (MH), which is an instance of a Markov Chain Monte Carlo sampler. In a nutshell, the MH algorithm holds at any point a sample X_0. Based on the X_0 sample a new candidate sample X' is proposed according to a probability distribution Q, with probability $Q(X'|X_0)$. The new sample X' is 'accepted' to replace X_0 with probability α:

$$\alpha = \min(\frac{P(X'|T) \cdot Q(X_0|X')}{P(X_0|T) \cdot Q(X'|X_0)}, 1)$$

otherwise the original sample X_0 is retained. It can be shown that after multiple iterations this yields samples X according to the distribution $P(X|T)$ irrespective of the way new candidate sets X' are proposed or the initial state of the algorithm, i.e. a more likely state X will pop-out more frequently from the sampler, than less likely states.

A relatively naive strategy can be used to propose candidate states X' given X_0 for our problem. It relies on

simply considering sets of nodes X' that are only different by a single member from X_0. Thus, with some probability p_{add} a random node $x \in \bar{X}_0$ is added to the set to form the candidate $X' = X_0 \cup x$. Alternatively, with probability p_{remove}, a member of X_0 is removed from the set of nodes, defining $X' = X_0 \cap x$ for $x \in X_0$. It is trivial to calculate the probabilities $Q(X'|X_0)$ and $Q(X'|X_0)$ based on p_{add}, p_{remove} and using a uniformly at random choice over nodes in X_0, $\bar{X}_0$, X' and $\bar{X}'$ when necessary.

A key issue when utilizing the MH algorithm is deciding how many iterations are necessary to get independent samples. Our rule of thumb is that $|V| \cdot \log |V|$ steps are likely to guarantee convergence to the target distribution P. After that number of steps the coupon collector's theorem states that each node in the graph would have been considered at least once by the sampler, and assigned to the honest or dishonest set. In practice, given very large traces T, the number of nodes that are difficult to categorise is very small, and a non-naive sampler requires few steps to produce good samples (after a certain burn in-period that allows it to detect the most likely honest region.)

Finally, given a set of N samples $X_i \sim P(X|T)$ output by the MH algorithm it is possible to calculate the marginal probabilities any node is honest. This is key output of the SybilInfer algorithm: given a node i it is possible to associate a probability it is honest by calculating: $\Pr[i \text{ is honest}] = \frac{\sum_{j \in [0, N-1)} I(i \in X_j)}{N}$, where $I(i \in X_j)$ is an indicator variable taking value 1 if node i is in the honest sample X_j, and value zero otherwise. Enough samples can be extracted from the sampler to estimate this probability with an arbitrary degree of precision.

More sophisticated samplers would make use of a better strategy to propose candidate states X' for each iteration. The choice of X' can, for example, be biased towards adding or removing nodes according to how often random walks starting at the single honest node land on them. We expect nodes that are reached often by random walks starting in the honest region to be honest, and the opposite to be true for dishonest nodes. In all cases this bias is simply an optimization for the sampling to take fewer iterations, and does not affect the correctness of the results.

4 Security evaluation

In this section we discuss the security of SybilInfer when under Sybil attack. We show analytically that we can detect when a social network suffers from a Sybil attack, and correctly label the Sybil nodes. Our assumptions and full proposal are then tested experimentally on synthetic as well as real-world data sets, indicating that the theoretical guarantees hold.

4.1 Theoretical results

The security of our Sybil detection scheme hinges on two important results. First, we show that we can detect whether a network is under Sybil attack, based on the social graph. Second, we show that we are able to detect Sybil attackers connected to the honest social graph, and this *for any* attacker topology.

Our first result states that:

> THEOREM A. In the absence of any Sybil attack, the distribution of $P(X = \text{Honest}|T)$, for a given size $|X|$, is close to uniform, and all cuts are equally likely ($E_{XX} \approxeq 0$).

This result is based on our assumption that a random walk over a social network is fast mixing meaning that, after $\log(|V|)$ steps, it visits nodes drawn from the stationary distribution of the graph. In our case the random walk is performed over a slightly modified version of the social graph, where the transition probability attached to each link ij is:

$$P_{ij} = \begin{cases} \min(\frac{1}{d_i}, \frac{1}{d_j}) & \text{if } i \to j \text{ is an edge in G} \\ 0 & \text{otherwise} \end{cases},$$

which guarantees that the stationary distribution is uniform over all nodes (i.e. $\Pi = \frac{1}{|V|}$). Therefore we expect that in the absence of an adversary the short walks in T to end at a network node drawn at random amongst all nodes $|V|$. In turn this means that the number of end nodes in the set of traces T, that end in the honest set X is $N_{XX} = \lim_{|T_X| \to \infty} \frac{|X|}{|V|} \cdot |T_X|$, where T_X is the number of traces in T starting within the set $|X|$. Substituting this in the equations presented in section 2.1 and 2.2 we get:

$$\text{Prob}_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|} \Rightarrow \quad (1)$$

$$\Pi + E_{XX} = \frac{N_{XX}}{N_{XX} + N_{X\bar{X}}} \cdot \frac{1}{|X|} \Rightarrow \quad (2)$$

$$\frac{1}{|V|} + E_{XX} = \frac{(|X|/|V|) \cdot |T_X|}{|T_X|} \cdot \frac{1}{|X|} \Rightarrow \quad (3)$$

$$E_{XX} = 0 \quad (4)$$

As a result, by sufficiently increasing the number of random walks T performed on the social graph, we can get E_{XX} arbitrarily close to zero. In turn this means that our distribution $P(T|X = \text{Honest})$ is uniform for given sizes of $|X|$, given our uniform a-prior $P(X = \text{Honest}|T)$.

In a nutshell by estimating E_{XX} for any sample X returned by the MH algorithm, and testing how close it is to zero we detect whether it corresponds to an attack (as we will see from theorem B) or a natural cut in the graph. We can increase the precision of the detector arbitrarily by increasing the number of walks T.

Our second results relates to the behaviour of the system under Sybil attack:

THEOREM B. Connecting *any* additional Sybil nodes to the social network, through a set of corrupt nodes, lowers the dishonest sub-graph conductance to the honest region, leading to slow mixing, and hence we expect $E_{XX} > 0$.

First we define the dishonest set $\bar{X}_0$ comprising all dishonest nodes connected to honest nodes in the graph. The set $\bar{X}_S$ contains all dishonest nodes in the system, including nodes in $\bar{X}_0$ and the Sybil nodes attached to them. It must hold that $|\bar{X}_0| < |\bar{X}_S|$, in case there is a Sybil attack. Second we note that the probability of a transition between an honest node $i \in X$ and a dishonest node $j \in \bar{X}$ cannot increase through Sybil attacks, since it is equal to $P_{ij} = \min(\frac{1}{d_i}, \frac{1}{d_j})$. At worst the corrupt node will increase its degree by connecting Sybils which has only the potential to decrease this probability. Therefore we have that $\sum_{x\in\bar{X}_S}\sum_{y\notin\bar{X}_S} P_{xy} \leq \sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0} P_{xy}$. Combining the two inequalities we get that:

$$\frac{\sum_{y\notin\bar{X}_S} P_{xy}}{|\bar{X}_S|} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0} P_{xy}}{|\bar{X}_0|} \Leftrightarrow \quad (5)$$

$$\frac{\sum_{y\notin\bar{X}_S} \frac{1}{|V|} P_{xy}}{|\bar{X}_S|\frac{1}{|V|}} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0} \frac{1}{|V|} P_{xy}}{|\bar{X}_0|\frac{1}{|V|}} \Leftrightarrow \quad (6)$$

$$\frac{\sum_{y\notin\bar{X}_S} \pi(x) P_{xy}}{\Pi(\bar{X}_S)} < \frac{\sum_{x\in\bar{X}_0}\sum_{y\notin\bar{X}_0} \pi(x) P_{xy}}{\Pi(\bar{X}_0)} \Leftrightarrow \quad (7)$$

$$\Phi(\bar{X}_S) < \Phi(\bar{X}_0). \quad (8)$$

This result signifies that independently of the topology of the adversary region the conductance of a sub-graph containing Sybil nodes will be lower compared with the conductance of the sub-graph of nodes that are simply compromised and connected to the social network. Lower conductance in turn leads to slower mixing times between honest and dishonest regions [20] which means that $E_{XX} > 0$, even for very few Sybils. This deviation is subject to the sampling variation introduced by the trace T, but the error can be made arbitrarily small by sampling more random walks in T.

These two results are very strong: they indicate that, in theory, a set of compromised nodes connecting to honest nodes in a social network, would get no advantage by connecting any additional Sybil nodes, since that would lead to their detection. Sampling regions of the graph with abnormally small conductance, through the use of the random walks T, should lead to their discovery, which is the theoretical foundation of our technique. Furthermore we established that techniques based on detecting abnormalities in the value of E_{XX} are *strategy proof*, meaning that there is no attacker strategy (in terms of special adversary topology) to foil detection.

4.2 Practical considerations

Models and assumptions are always an approximation of the real world. As a result, careful evaluation is necessary to ensure that the theorems are robust to deviations from the ideal behaviour assumed so far.

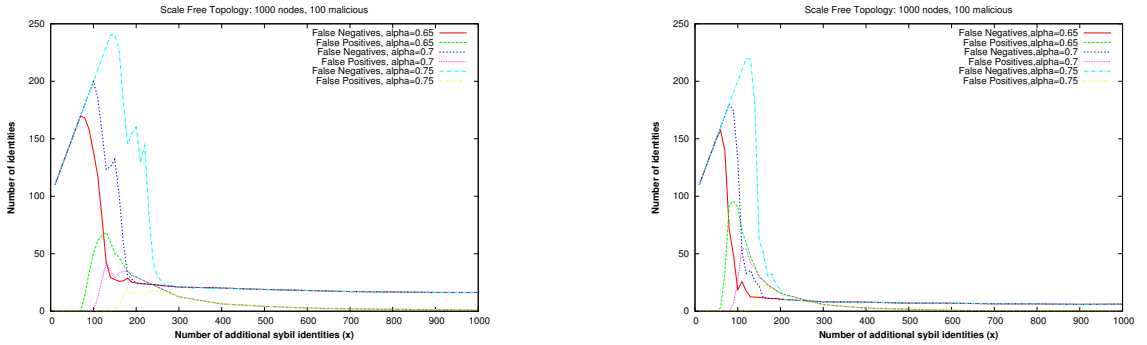
The first practical issue concerns the fast mixing properties of social networks. There is a lot of evidence that social networks exhibit this behaviour [18], and previous proposals relating to Sybil defence use and validate the same assumption [27, 26]. SybilInfer makes an further assumption, namely that the modified random walk over the social network, that yields a uniform distribution over all nodes, is also fast mixing for real social networks. The probability $P_{ij} = \min(\frac{1}{d_i}, \frac{1}{d_j})$, depends on the mutual degrees of the nodes i and j, and makes the transition to nodes of higher degree less likely. This effect has the potential to slow down mixing times in the honest case, particularly when there is a high variation in node degrees. This effect can be alleviated by removing random edges from high degree nodes to guarantee that the ratio of maximum and minimum node degree in the graph is bounded (an approach also used by SybilLimit.)

The second consideration also relates to the fast mixing properties of networks. While in theory fast mixing networks should not exhibit any small cuts, or regions of abnormally low conductance, in practice they do. This is especially true for regions with new users that have not had the chance to connect to many others, as well as social networks that only contain users with particular characteristics (like interest, locality, or administrative groups.) Those regions yield, even in the honest case, sample cuts that have the potential to be mistaken as attacks. This effect forces us to consider a threshold E_{XX} under which we consider cuts to be simply false positives. In turn this makes the guarantees of schemes weaker in practice than in theory, since the adversary can introduce Sybils into a region undetected, as long as the set threshold E_{XX} is not exceeded.

The threshold E_{XX} is chosen to be $\alpha \cdot E_{XXmax}$, where $E_{XXmax} = \frac{1}{|X|} - \frac{1}{|V|}$, and α is a constant between 0 and 1. Here α can be used to control the tradeoff between false positives and false negatives. A higher value of alpha enables the adversary to insert a larger number of sybils undetected but reduces the false positives. On the other hand, a smaller value of α reduces the number of Sybils that can be introduced undetected but at the cost of higher number of false positives.

Given these practical considerations, we can formulate a weaker security guarantee for SybilInfer:

THEOREM C. Given a certain "natural" threshold value for E_{XX} in an honest social network, a dishonest region performing a Sybil attack will exceed it after introducing a certain number of Sybil nodes.



(a) Average degree compromised nodes

(b) Low degree compromised nodes

Figure 3. Synthetic Scale Free Topology: SybilInfer Evaluation as a function of additional Sybil identities (ψ) introduced by colluding entities. False negatives denote the total number of dishonest identities accepted by SybilInfer while false positives denote the number of honest nodes that are misclassified.

This theorem is the result of Theorem B that demonstrates that the conductance keeps decreasing as the number of Sybils attached to a dishonest region increases. This in turn will slow down the mixing time between the honest and dishonest region, leading to an increasingly large E_{XX}.

Intuitively, as the attack becomes larger, the cut between honest and dishonest nodes becomes increasingly distinct, which makes Sybil detection easier. It is important to note that as more Sybils are introduced into the dishonest region, the probability of the whole region being detected as an attack increases, not only the new Sybil nodes. This provides strong disincentives to the adversary from performing larger Sybil attacks, since even previously undetected malicious nodes might be flagged as Sybils.

4.3 Experimental evaluation using synthetic data

We first experimentally demonstrate the validity of Theorem C using synthetic topologies. Our experiments consist of building synthetic social network topologies, injecting a variable number of Sybil nodes, and applying SybilInfer to establish how many of them are detected. A key issue we explore is the number of introduced Sybil nodes under which Sybil attacks are not detected.

Social networks exhibit a scale-free (or power law) node degree topology [21]. Our network synthesis algorithm replicates this structure through preferential attachment, following the methodology of Nagaraja [18]. We create m_0 initial nodes connected in a clique, and then for each new node v, we create m new edges to existing nodes, such that the probability of choosing any given node is proportional to the degree of that node; i.e.: $\Pr[(v,i)] = \frac{d_i}{\sum_j d_j}$, where d_i is the degree of node i. In our simulations, we use $m = 5$, giving an average node degree of 10.

In such a scale free topology of 1000 nodes, we consider a fraction $f = 10\%$ of the nodes to be compromised by a single adversary. The compromised nodes are distributed uniformly at random in the topology. Compromised nodes introduce ψ additional Sybil nodes and establish a scale free topology amongst themselves. We configure SybilInfer to use 20 samples for computing the marginal probabilities, and label as honest the set of nodes whose marginal probability of being honest is greater than 0.5. The experiment is repeated 100 times with different scale free topologies.

Figure 3(a) illustrates the false positives and false negatives classifications returned by SybilInfer, for varying value of ψ, the number of additional Sybil nodes introduced. We observe that when $\psi < 100$, $\alpha = 0.7$, then all the malicious identities are classified as honest by SybilInfer. However, there is a threshold at $\psi = 100$, beyond which all of the Sybil identities, including the initially compromised entities are flagged as attackers. This is because beyond this point, the E_{XX} for the Sybil region exceeds the natural threshold leading to full detection, validating Theorem C. The value $\psi = 100$ is clearly the optimal attack strategy, in which the attacker can introduce the maximal number of Sybils without being detected. We also note that even in the worst case, the false positives are less than 5%. The false positive nodes have been misclassified because these nodes are closer to the Sybil region; SybilInfer is thus incentive compatible in the sense that nodes which have mostly honest friends are likely not to be misclassified.

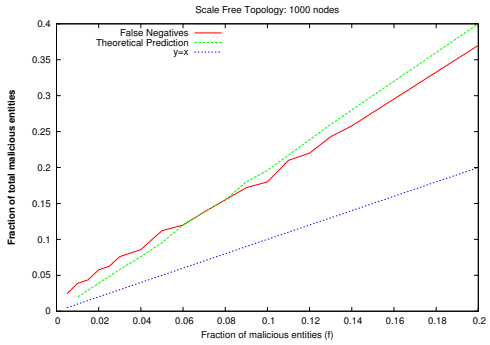


Figure 4. Scale Free Topology: fraction of total malicious and Sybil identities as a function of real malicious entities.

We can also see the effect of varying the threshold E_{XX}. As α is increased from 0.65 to 0.7, the ψ for the optimal attacker strategy increases from 70 to 100. This is because an increase in the threshold E_{XX} allows the adversary to insert more Sybils undetected. The advantage of increasing α lies in reducing the worst case false positives. We can see that by increasing α from 0.7 to 0.75, the worst case false positives can be reduced from 5% to 2%.

Note that for the remainder of the paper, we shall use $\alpha = 0.7$.

We also wish to show that the security of our scheme depends primarily on the number of colluding malicious nodes and not on the number of attack edges. To this end, we chose the compromised nodes to have the lowest number of attack edges (instead of choosing them uniformly at random), and repeat the experiment. Figure 3(b) illustrates that the false positives and false negatives classifications returned by SybilInfer, where the average number of attack edges are 500. Note that these results are very similar to the previous case illustrated in Figure 3(a), where the number of attack edges is around 800. This analysis indicates that the security provided by SybilInfer primarily depends on the number of colluding entities. The implication is that the compromise of high degree nodes does not yield any significant advantage to the adversary. As we shall see, this is in contrast to SybilGuard and SybilLimit, which are extremely vulnerable when high degree nodes are compromised.

Our next experiment establishes the number of Sybil nodes that can be inserted into a network given different fractions of compromised nodes. We vary the fraction of compromised colluding nodes f, and for each value of f, we compute the optimal number of additional Sybil identities that the attackers can insert, as in the previous experiment.

Figure 4 presents a plot of the maximum Sybil identities as a function of the compromised fraction of nodes f. Note that our theoretical prediction (which is strategy-independent) matches closely with the attacker strategy of connecting Sybil nodes in a scale free topology. The adversary is able to introduce roughly about 1 additional Sybil identity per real entity. For instance, at $f = 0.2$, the total number of Sybil identities is 0.37. As we observe from the figure the ability of the adversary to just include about one additional Sybil identity per compromised node embedded in the social network remains constant, no matter the fraction f of compromised nodes in the network.

4.4 Experimental evaluation using real-world data

Next we validate the security guarantees provided by SybilInfer using a sampled LiveJournal topology. A variant of snowball [9] sampling was used to collect the full data set data, comprising over 100,000 nodes.

To perform our experiments we chose a random node and collect all nodes in its three hop neighbourhood. The resulting social network has about 50,000 nodes. We then perform some pre-processing step on the sub-graph:

- Nodes with degree less than 3 are removed, to filter out nodes that are too new to the social network, or inactive.
- If there is an edge between $A \rightarrow B$, but no edge between $B \rightarrow A$, then $A \rightarrow B$ is removed (to only keep the symmetric friendship relationships.)

We note that despite this pre-processing nodes all degrees can be found in the final dataset, since nodes with initial degree over 3 will have some edges removed reducing their degree to less than 3.

After pre-processing, the social sub-graph consists of about 33,000 nodes. First, we ran SybilInfer on this topology without introducing any artificial attack. We found a bottleneck cut diving off about $2,000$ Sybil nodes. It is impossible to establish whether these nodes are false positives (a rate of 6%) or a real-world Sybil attack present in the LiveJournal network. Since there is no way to establish ground truth, we do not label these nodes as either honest/dishonest.

Next, we consider a fraction f of the nodes to be compromised and compute the optimal attacker strategy, as in our experiments with synthetic data. Figure 5 shows the fraction of malicious identities accepted by SybilInfer as a function of fraction of malicious entitites in the system. The trend is similar to our observations on synthetic scale free topologies. At $f = 0.2$, the fraction of Sybil identities accepted by SybilInfer is approximately 0.32.

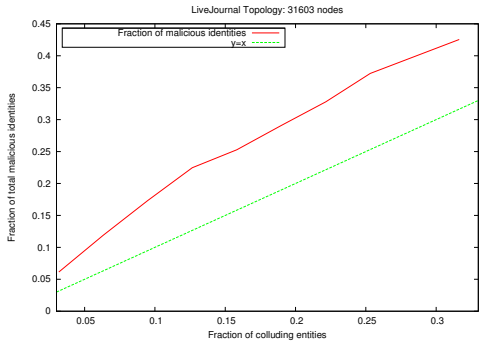


Figure 5. LiveJournal Topology: fraction of total malicious identities as a function of real malicious entities

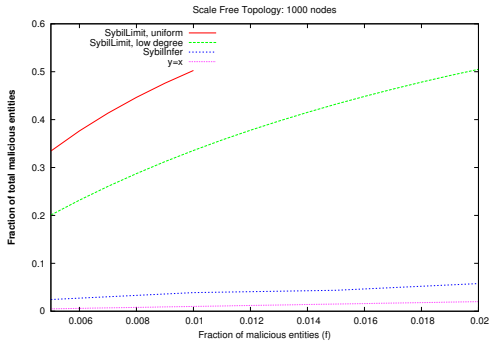


Figure 6. Comparison with related work

4.5 Comparison with SybilLimit and SybilGuard

SybilGuard [27] and SybilLimit [26] are state of the art decentralized protocols that defend against Sybil attacks. Similar to SybilInfer, both protocols exploit the fact that a Sybil attack disrupts the fast mixing property of the social network topology, albeit in a heuristic fashion. A brief overview of the two systems can be found in the appendix, and their full descriptions is given in [27, 26].

Figure 6 compares the performance of SybilInfer with the performance of SybilLimit. First it is worth noting that the fraction of compromised nodes that SybilLimit tolerates is only a small fraction of the range within which SybilInfer provide its guarantees. SybilLimit tolerates up to $f = 0.02$ compromised nodes when the degree of attackers is low (about degree 5 – green line), while we have already shown the performance of SybilLimit for compromised fractions up to $f = 0.35$ in figure 4. Within the interval SybilLimit is applicable, our system systematically outperforms: when very few compromised nodes are present in the system ($f = 0.01$) our system only allows them to control less than 5% of the entities in the system, versus SybilLimit that allows them to control over 30% of entities (rendering insecure byzantine fault tolerance mechanisms that requires at least 2/3 honest nodes) At the limit of SybilLimit's applicability range when $f = 0.02$, our approach caps the number of dishonest entities in the system to fewer than 8%, while SybilLimit allows about 50% dishonest entities. (This large fraction renders leader election or other voting systems ineffective.)

An important difference between SybilInfer and SybilLimit is that the former is not sensitive to the degree of the attacker nodes. SybilLimit provides very weak guarantees when high degree (e.g. degree 10 – red line) nodes are compromised, and can protect the system only for $f < 0.01$. In this case SybilInfer allows for 5% total malicious entities, while SybilLimit allows for over 50%.

This is an illustration that SybilInfer performs an order of magnitude better than the state of the art both in terms of range of applicability and performance within that range (SybilGuard's performance is strictly worse than SybilLimit's performance, and is not illustrated.) An obvious question is: "why does SybilInfer perform so much better than SybilGuard and SybilLimit?" It is particularly pertinent since all three systems are making use of the same assumptions, and a similar intuition, that there should be a "gap" between the honest and Sybil regions of a social network. The reason SybilLimit and SybilGuard provide weaker guarantees is that they interpret these assumptions in a very sensitive way: they assume that *an overwhelming majority* of random walks staring in the honest region will stay in the honest region, and then bound the number of walks originating from the Sybil region via the number of corrupt edges. As a result they are very sensitive to the length of those walks, and can only provide strong guarantees for a very small number of corrupt edges. Furthermore the validation procedure relies on collisions between honest nodes, via the birthday paradox, which adds a further layer of inefficiency to estimating good from bad regions.

SybilInfer, on the other hand, interprets the disruption in fast-mixing between the honest and dishonest region simply as a faint bias in the last node of a short random walk (as illustrated in figures 2(a) and 2(b).) In our experiments, as well as in theory, we observe a very large fraction of the T walks crossing between the honest and dishonest regions. Yet the faint difference in the probability of landing on nodes in the honest and dishonest regions is present, and the sampler makes use of it to get good cuts between the honest and dishonest nodes.

4.6 Computational and time complexity

Two implementations of the SybilInfer sampler were build in Python and C++, of about 1KLOC each. The Python implementation can handle 10K node networks, while the C++ implementation has handled up to 30K node networks, returning results in seconds.

The implementation strategy for both samplers has favoured a low time complexity over storage costs. The critical loop performs $O(|V| \cdot \log |V|)$ Metropolis Hastings iterations per sample returned, each only requiring about $O(\log |V|)$ operations. Two copies of the full state are stored, as well as associated data that allows for fast updating of the state, which requires $O(|V|)$ storage. The transcript of evidence traces T is also stored, as well as an index over it, which dominates the storage required and makes it order $O(|V| \cdot \log |V|)$.

There is a serious time complexity penalty associated with implementing non-naive sampling strategies. Our Python implementation biases the candidate moves towards nodes that are more or less likely to be part of the honest set. Yet exactly sampling nodes from this known probability distribution, naively may raise the cost of each iteration to be $O(|V|)$. Depending on the differential between the highest and lowest probabilities, faster sampling techniques like rejection sampling [15] can be used to bring the cost down. The Python implementation uses a variant of Metropolis-Hastings to implement selection of candidate nodes for the next move, at a computation cost of $O(\log |V|)$. The C++ implementation uses naive sampling from the honest or dishonest sets, and has a very low cost per iteration of order $O(1)$.

The Markov chain sampling techniques used consider sequences of states that are very close to each other, differing at most by a single node. This enables a key optimization, where the counts $N_{XX}, N_{X\bar{X}}, N_{\bar{X}X}$ and $N_{\bar{X}\bar{X}}$ are stored for each state and updated when the state changes. This simple variant of self-adjusting computation [1], allows for very fast computations of the probabilities associated with each state. Updating the counts, and associated information is an order $O(\log |V|)$ operation. The alternative, of recounting these quantities from T would cost $O(|V| \log |V|)$ for every iteration, leading to a total computational complexity for our algorithm of $O((|V| \log |V|)^2)$. Hence implementing it is vital to getting results fast.

Finally our implementations use a zero-copy strategy for the state. Two states and all associated information are maintained at any time, the current state and the candidate state. Operations on the candidate state can be done and undone in $O(\log |V|)$ per operation. Accepted moves can be committed to the current state at the same cost. These operations can be used to maintain the two states synchronised for use by the Metropolis-Hastings sampler. The naive strategy of re-writing the full state would cost $O(|V|)$ per iteration, making the overall complexity of the scheme $O(|V|^2 \log |V|)$.

5 Deployment Strategies

So far we presented an overview of the SybilInfer algorithm, as well as a theoretical and empirical evaluation of its performance when it comes to detecting Sybil nodes. The core of the algorithm outperforms SybilGuard and SybilLimit, and is applicable in settings beyond which the two systems provide no security guarantees whatsoever. Yet a key difference between the previous systems and SybilInfer is the latter's reliance on the full friendship graph to perform the random walks that drive the inference engine. In this section we discuss how this constraint still allows SybilInfer to be used for important classes of applications, as well as how it can be relaxed to accommodate peer-to-peer systems with limited resources per client.

5.1 Full social graph knowledge

The most straightforward way of applying SybilInfer is using the full graph of a social network to infer which nodes are honest and which nodes are Sybils, given a known honest seed node. This is applicable to centralised on-line services, like free email hosting services, blogging sites, and discussion forums that want to deter spammers. Today those systems use a mixture of CAPTCHA [25] and network based intrusion detection to eliminate mass attacks. SybilInfer could be used to either complement those mechanisms and provide additional information as to which identities are suspicious, or replace those systems when they are expensive and error prone. One of the first social network based Sybil defence systems, Advogato [14], worked in such a centralized fashion.

The need to know the social graph does not preclude the use of SybilInfer in distributed or even peer-to-peer systems. Social networks, once mature, are generally stable and do not change much over time. Their rate of change is by no means comparable to the churn of nodes in the network, and as a result the structure of the social network could be stored and used to perform inference on multiple nodes in a network, along with a mechanisms to share occasional updates. The storage overhead for storing large social networks is surprisingly low: a large social network with 10 billion nodes (roughly the population of planet earth) with each node having about 1000 friends, can be stored in about 187Gb of disk space uncompressed. In such settings it is likely that SybilInfer computation will be the bottleneck, rather than storage of the graph, for the foreseeable future.

A key application of Sybil defences is to ensure that volunteer relays in anonymous communication networks belong to independent entities, and are not controlled

by a single adversary. Practical systems like Mixmaster [17], Mixminion [5] and Tor [7] operate such a volunteer based anonymity infrastructure, that are very susceptible to Sybil attacks. Extending such an infrastructure to use SybilInfer is an easy task: each relay in the system would have to indicate to the central directory services which other nodes it considers honest and non-colluding. The graph of nodes and mutual trust relations can be used to run SybilInfer centrally by the directory service, or by each individual node that wishes to use the anonymizing service. Currently, the largest of those services, the Tor network has about 2000 nodes, which is well within the computation capabilities of our implementations.

5.2 Partial social graph knowledge

SybilInfer can be used to detect and prevent Sybil attacks, using only a partial view of the social graph. In the context of a distributed or peer-to-peer system each user discovers only a fixed diameter sub-graph around them. For example a user may choose to retrieve and store all other users two or three hops away in the social network graph, or discover a certain threshold of nodes in a breadth first manner. SybilInfer is then applied on the extracted sub-graph to detect potential Sybil regions. This allows the user to prune its social neighbourhood from any Sybil attacks, and is sufficient for selecting a set of honest nodes when sampling from the full network is not required. Distributed backup and storage, and all friend and friend-of-friend based sharing protocols can benefit from such protection. The storage and communication cost of this scheme is constant and relative to the number of nodes in the chosen neighbourhood.

In cases where nodes can afford to know a larger fraction of the social graph, they could choose to discover $O(c \cdot \sqrt{|V|})$ nodes in their neighbourhood, for some small integer c. This increases the chances two arbitrary nodes have to know a common node, that can perform the SybilInfer protocol and act as an introduction point for the nodes. In this protocol Alice and Bob want to ensure the other party is not a Sybil. They find a node C that is in the $c \cdot \sqrt{|V}$ neighbourhood of both of them, and each make sure that with high probability it is honest. They then contact node C that attests to both of them, given its local run of the SybilInfer engine, that they are not Sybil nodes (with C as the honest seed.) This protocol introduces a single layer of transitive trust, and therefore it is necessary for Alice and Bob to be quite certain that C is indeed honest. Its storage and communication cost is $O(\sqrt{|V|})$, which is the same order of magnitude as SybilLimit and SybilGuard. Modifying this simple minded protocol into a fully fledged one-hop distributed hash table [13] is an interesting challenge for future work.

SybilInfer can also be applied to specific on-line communities. In such cases a set of nodes belonging to a certain community of interest (a social club, a committee, a town, etc.) can be extracted to form a sub-graph. SybilInfer can then be applied on this partial view of the graph, to detect nodes that are less well integrated than others in the group. There is an important distinction between using SybilInfer in this mode or using it with the full graph: while the results using the full graph output an "absolute" probability for each node being a Sybil, applying SybilInfer to a partial view of the network only provides a "relative" probability the node is honest *in that context.* It is likely that nodes are tagged as Sybils, because they do not have many contacts within the select group, which given the full graph would be classified as honest. Before applying SybilInfer in this mode it is important to assess, at least, whether the subgroup is fast-mixing or not.

5.3 Using SybilInfer output optimally

Unlike previous systems the output of the SybilInfer algorithm is a probabilistic statement, or even more generally, a set of samples that allows probabilistic statements to be estimated. So far in the work we discussed how to make inferences about the marginal probability specific nodes are honest of dishonest by using the returned samples to compute $\Pr[i \text{ is honest}]$ for all nodes i. In our experiments we applied a 0.5 threshold to the probability to classify nodes as honest or dishonest. This is a rather limited use of the richer output that SybilInfer provides.

Distributed system applications can, instead of using marginal probabilities of individual nodes, estimate the probability that the particular security guarantees they require hold. High latency anonymous communication systems, for example, require a set of different nodes such that with high probability at least one of them is honest. Path selection is also subject to other constraints (like latency.) In this case the samples returned by SybilInfer can be used to calculate exactly the sought probability, i.e. the probability a single node in the chosen path is honest. Onion routing based system, on the other hand are secure as long as the first and last hop of the relayed communication is honest. As before, the samples returned by SybilInfer can be used to choose a path that has a high probability to exhibit this characteristic.

Other distributed applications, like peer-to-peer storage and retrieval have similar needs, but also tunable parameters that depend on the probability of a node being dishonest. Storage systems like OceanStore, use Rabin's information dispersion algorithm to divide files into chunks stored and retrieved to reconstruct a file. The degree of redundancy required crucially depends on the probability nodes are compromised. Such algorithms can use SybilInfer to foil Sybil attacks, and calculate the probability the set of nodes to be used to store particular files contains certain fractions of honest nodes. This probability can in turn inform the choice of parameters to maximise the sur-

vivability of the files.

Finally a note of warning should accompany any Sybil prevention scheme: it is not the goal of SybilInfer (or any other such scheme) to ensure that all adversary nodes are filtered out of the network. The job of SybilInfer is to ensure that a certain fraction of existing adversary nodes cannot significantly increase its control of the system by introducing 'fake' Sybil identities. As it is illustrated by the examples on anonymous communications and storage, system specific mechanisms are still crucial to ensure that a minority of adversary entities cannot compromise any security properties. SybilInfer can only ensure that this minority remains a minority and cannot artificially increase its share of the network.

Sybil defence schemes are also bound to contain false-positives, namely honest nodes labeled as Sybils. For this reason other mechanisms need to be in place to ensure that those users can seek a remedy to the automatic classification they suffered from the system, potentially by making some additional effort. Proofs-of-work, social introduction services, or even payment targeting those users could be a way of ensuring SybilInfer is not turned into an automated social exclusion mechanism.

6 Conclusion

We presented SybilInfer, an algorithm aimed at detecting Sybil attacks against peer-to-peer networks or open services, and label which nodes are honest and which are dishonest. Its applicability and performance in this task is an order of magnitude better than previous systems making similar assumptions, like SybilGuard and SybilLimit, even though it requires nodes to know a substantial part of the social structure within which honest nodes are embedded. SybilInfer illustrates how robust Sybil defences can be bootstrapped from distributed trust judgements, instead of a centralised identity scheme. This is a key enabler for secure peer-to-peer architectures as well as collaborative web 2.0 applications.

SybilInfer is also significant due to the use of machine learning techniques and their careful application to a security problem. Cross disciplinary designs are a challenge, and applying probabilistic techniques to system defence should not be at the expense of strength of protection, and strategy-proof designs. Our ability to demonstrate that the underlying mechanisms behind SybilInfer is not susceptible to gaming by an adversary arranging its Sybil nodes in a particular topology is, in this aspect, a very import part of the SybilInfer security design.

Yet machine learning techniques that take explicitly into account noise and incomplete information, as the one contained in the social graphs, are key to building security systems that degrade well when theoretical guarantees are not exactly matching a messy reality. As security increasingly becomes a "people" problem, it is likely that approaches that treat user statements beyond just black and white and make explicit use of probabilistic reasoning and statements as their outputs will become increasingly important in building safe systems.

Acknowledgements. This work was performed while Prateek Mittal was an intern at Microsoft Research, Cambridge, UK. The authors would like to thank Carmela Troncoso, Emilia Käsper, Chris Lesniewski-Laas, Nikita Borisov and Steven Murdoch for their helpful comments on the research and draft manuscript. Our shepherd, Tadayoshi Kohno, and ISOC NDSS 2009 reviewers provided valuable feedback to improve this work. Barry Lawson was very helpful with the technical preparation of the final manuscript.

References

[1] U. A. Acar. *Self-adjusting computation*. PhD thesis, Pittsburgh, PA, USA, 2005. Co-Chair-Guy Blelloch and Co-Chair-Robert Harper.

[2] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In *STOC*, pages 230–240. ACM, 1987.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 299–314, New York, NY, USA, 2002. ACM.

[4] F. Dabek. A cooperative file system. Master's thesis, MIT, September 2001.

[5] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, May 2003.

[6] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant dht routing. In *ESORICS 2005: Proceedings of the European Symp. Research in Computer Security*, pages 305–318, 2005.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[8] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[9] L. Goodman. Snowball sampling. *Annals of Mathematical Statistics*, 32:148–170.

[10] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[11] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[12] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[13] C. Lesniewski-Laas. A sybil-proof one-hop dht. In *Proceedings of the Workshop on Social Network Systems*, Glasgow, Scotland, April 2008.
[14] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium*, pages 18–18, Berkeley, CA, USA, 1998. USENIX Association.
[15] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.
[16] S. Milgram. The small world problem. 2:60–67, 1967.
[17] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol — Version 2. IETF Internet Draft, July 2003.
[18] S. Nagaraja. Anonymity in the wild: Mixes on unstructured networks. In N. Borisov and P. Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. Springer.
[19] D. J. Phillips. Defending the boundaries: Identifying and countering threats in a usenet newsgroup. *Inf. Soc.*, 12(1), 1996.
[20] D. Randall. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engineering*, 8(2):30–41, 2006.
[21] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing Journal*, 6(1), Aug. 2002.
[22] Secondlife: secondlife.com.
[23] Sophos. Sophos facebook id probe shows 41% of users happy to reveal all to potential identity thieves, August 14 2007.
[24] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.
[25] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *In Proceedings of EUROCRYPT 03, Lecture Notes in Computer Science*, 2003.
[26] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.
[27] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, 2006.
[28] del.icio.us: delicious.com.
[29] Facebook: www.facebook.com.
[30] Orkut: www.orkut.com.
[31] Wikipedia: www.wikipedia.org.

A An overview of SybilGuard and SybilLimit

A.1 SybilGuard

In SybilGuard, each node first obtains $\sqrt{n}$ independent samples from the set of honest nodes of size n. Since, for a given honest network size n, the mixing time of the social network is $O(\log n)$, it suffices to use a long random walk of length $w = \sqrt{n} \cdot \log n$ to gather those samples. To prevent active attacks biasing the samples, SybilGuard performs the random walk over constrained *random route*. Random routes require each node to use a pre-computed random permutation as a one to one mapping from incoming edges to outgoing edges giving them the following important properties:

- Convergence: Two random routes entering an honest node along the same edge will always exit along the same edge.
- Back-traceability: The outgoing edge of a random route uniquely determines the incoming edge at an honest node.

Since the size of the set of honest nodes, n is unknown, SybilGuard requires an estimation technique to figure the needed length of the random route.

The validation criterion for accepting a node as honest in SybilGuard is that there should be an intersection between the random route of the verifier node and the suspect node. It can be shown using the birthday paradox that if two honest nodes are able to obtain $\sqrt{n}$ samples from the honest region, then their samples will have an non empty intersection with high probability, and will thus be able to validate each other.

There are cases when the random route of an honest node ends up within the Sybil region, leading to a "bad" sample, and the possibility of accepting Sybil nodes as honest. Thus, SybilGuard is only able to provide bounds on the number of Sybil identities if such an event is rare, which translates into an assumption that the maximum number of attack edges is $g = o(\frac{\sqrt{n}}{\log n})$. To further reduce the effects of random routes entering the Sybil region (bad samples), nodes in SybilGuard can perform random routes along all their edges and validate a node only if a majority of these random routes have an intersection with the random routes of the suspect.

SybilGuard's security really depends on the number of attack edges in the system, connecting honest and dishonest users. To intersect a verifier's random route, a sybil node's random route must traverse an attack edge (say A). Due to the convergence property, the random routes of all sybil nodes traversing A will intersect the verifier's random route at the same node and along the same edge. All such nodes form an equivalence group from the verifier's perspective. Thus, the number of sybil groups is bounded by the number of attack edges, i.e. g. Moreover, due to the back-traceability property, there can be at most w distinct routes that intersect the verifiers route at a particular node and a particular edge. Thus, there is a bound on the size of the equivalence groups. To sum up, SybilGuard divides the accepted nodes into equivalence groups, with the guarantee that there are at most g sybil groups whose

maximum size is w.

Unlike SybilInfer, nodes in SybilGuard do not require knowledge of the complete network topology. On the other hand, the bounds provided by SybilGuard are quite weak: in a million node topology, SybilGuard accepts about 2000 Sybil identities per attack edge! (Attack edges are trust relations between an honest and a dishonest node.) Since the bounds provided by SybilGuard depend on the number of attack edges, high degree nodes would be attractive targets for the attackers. To use the same example, in a million node topology, the compromise of about 3 high degree nodes with about 100 attack edges each enables the adversary to control more than $1/3$ of all identities in the system, and thus prevent honest nodes from reaching byzantine consensus. In contrast, the bounds provided by SybilInfer depend on the number of colluding entities in the social network, and not on the number of attack edges. Lastly, SybilGuard is only able to provide bounds on Sybil identities when $g = o(\frac{\sqrt{n}}{\log n})$, while SybilInfer is not bound by any such threshold on the number of colluding entities.

A.2 SybilLimit

In contrast to SybilGuard's methodology of using a single instance of a very long random route, SybilLimit employs multiple instances ($\sqrt{m}$) of short random walks ($O(\log n)$) to sample nodes from the honest set, where m denotes the number of edges amongst the honest nodes. It can be shown that as long as $g = o(\frac{n}{\log n})$, then with high probability, a short random walk is likely to stay within the set of honest nodes, i.e., the probability of a sample being "bad" is small. The validation criterion for accepting a node as honest in SybilLimit is that there should be an intersection amongst the last edge (i.e. the *tail*) of the random routes of the verifier and the suspect. Similar to SybilGuard, it can be shown using the birthday paradox that two honest nodes will validate each other with high probability. Note that if a random route enters the sybil region, then the malicious tail of that route could advertise intersections with many sybil identities. To counter this attack, a further "balance" condition is imposed that ensures that a particular intersection cannot validate arbitrary number of identities. Sybil identities can induce a total of $g \cdot w \cdot m$ intersections at honest tails. Thus, for every attack edge, SybilLimit accepts at most $w = O(\log n)$ sybil identities.

A key shortcoming of SybilLimit is its reliance on the value of $w = O(\log n)$, the length of the short random walk, without any mechanisms to estimate it. Yet the correct estimation of this parameter is essential to calculate the security bounds of the scheme. Underestimating it is leads to an increase of false positives, while overestimating it will result in the random walks ending in the Sybil region, allowing the adversary to fake intersections with sybil identities.

Assuming there was a way to estimate all parameter required by SybilLimit, our proposal, SybilInfer, provides an order of magnitude better guarantees. Furthermore these guarantees relate to the number of (real) dishonest entities in the system, unlike SybilLimit that depends on number of attack edges. As noted, in comparison with SybilGuard, SybilInfer does not assume any threshold on the number of colluding entities, while SybilLimit can bound the number of sybil identities only when $g = o(\frac{n}{\log n})$.