Aiding the Detection of Fake Accounts in Large Scale Social Online Services

Qiang Cao [†]
Duke University
qiangcao@cs.duke.edu

Michael Sirivianos [‡]
Telefonica Research
msirivi@tid.es

Xiaowei Yang
Duke University
xwy@cs.duke.edu

Tiago Pregueiro
Tuenti, Telefonica Digital
tiago@tuenti.com

Abstract

Users increasingly rely on the trustworthiness of the information exposed on Online Social Networks (OSNs). In addition, OSN providers base their business models on the marketability of this information. However, OSNs suffer from abuse in the form of the creation of fake accounts, which do not correspond to real humans. Fakes can introduce spam, manipulate online rating, or exploit knowledge extracted from the network. OSN operators currently expend significant resources to detect, manually verify, and shut down fake accounts. Tuenti, the largest OSN in Spain, dedicates 14 full-time employees in that task alone, incurring a significant monetary cost. Such a task has yet to be successfully automated because of the difficulty in reliably capturing the diverse behavior of fake and real OSN profiles.

We introduce a new tool in the hands of OSN operators, which we call *SybilRank* . It relies on social graph properties to rank users according to their perceived likelihood of being fake (Sybils). SybilRank is computationally efficient and can scale to graphs with hundreds of millions of nodes, as demonstrated by our Hadoop prototype. We deployed SybilRank in Tuenti's operation center. We found that ∼90% of the 200K accounts that SybilRank designated as most likely to be fake, actually warranted suspension. On the other hand, with Tuenti's current user-report-based approach only ∼5% of the inspected accounts are indeed fake.

1 Introduction

The surge in popularity of online social networking (OSN) services such as Facebook, Twitter, Digg, LinkedIn, Google+, and Tuenti has been accompanied by an increased interest in attacking and manipulating them. Due to their open nature, they are particularly vulnerable to the Sybil attack [26], under which a malicious user can create multiple fake OSN accounts.

The problem. It has been reported that 1.5 million fake or compromised Facebook accounts were on sale during February 2010 [7]. Fake (Sybil) OSN accounts can be used for various purposes [6, 7, 9]. For instance, they enable spammers to abuse an OSN's messaging system to post spam [28, 51], or waste an OSN advertising customer's resources by making him pay for online ad clicks or impressions from or to fake profiles. Fake accounts can also be used to acquire users' private contact lists [17]. Sybils can use the "+1" button to manipulate Google search results [11] or to pollute location crowdsourcing results [8]. Furthermore, fake accounts can be used to access personal user information [27] and perform large-scale crawls over social graphs [42].

The challenge. Due to the multitude of the reasons behind their creation (§2.1), real OSN Sybils manifest numerous and diverse profile features and activity patterns. Thus, automated Sybil detection (e.g., Machine-Learning-based) does not yield the desirable accuracy (§2.2). As a result, adversaries can cheaply create fake accounts that are able to evade detection [19]. At the same time, although the research community has extensively discussed social-graph-based Sybil defenses [23, 52–54, 57, 58], there is little evidence of wide industrial adoption due to their shortcomings in terms of effectiveness and efficiency (§2.4). Instead, OSNs employ time-consuming manual account verification, driven by user reports on abusive accounts. However, only a small fraction of the inspected accounts are indeed fake, signifying inefficient use of human labor (§2.3).

If an OSN provider can detect Sybil nodes in its system effectively, it can improve the experience of its users and their perception of the service by stemming annoying spam messages and invitations. It can also increase the marketability of its user base and its social graph. In addition, it can enable other online services or distributed systems to treat a user's online social network identity as an authentic digital identity, a vision foreseen by recent efforts such as Facebook Connect [2].

We therefore aim to answer the question: "can we design a social-graph-based mechanism that enables a multi-million-user OSN to pick up accounts that are very likely to be Sybils?" The answer can help an OSN stem malicious activities in its network. For example, it can enable human verifiers to focus on likely-to-be-fake accounts. It can also guide the OSN in sending CAPTCHA [14] challenges to suspicious accounts, while running a lower risk to annoy legitimate users.

Our solution. We present the design (§4), implementation (§5), and evaluation (§6,§7,§5) of SybilRank. SybilRank is a Sybil inference scheme customized for OSNs whose social relationships are bidirectional.

[†] Part of Q. Cao's work was conducted while interning with Telefonica Research.
[‡] M. Sirivianos is currently with the Cyprus University of Technology.

Our design is based on the same assumption as in previous work on social-graph-based defenses [23, 52–54, 57, 58]: that OSN Sybils have a disproportionately small number of connections to non-Sybil users. Differently, our system achieves a significantly higher detection accuracy at a substantially lower computational cost. It can also be implemented in a parallel computing framework, e.g., MapReduce [24], enabling the inference of Sybils in OSNs with hundreds of millions of users.

Social-graph-based solutions uncover Sybils from the perspective of already known non-Sybil nodes (*trust seeds*). Unlike [52] and [53], SybilRank's computational cost does not increase with the number of trust seeds it uses (§4.5). This facilitates the use of multiple seeds to increase the system's robustness to seed selection errors, such as designating a Sybil as a seed. It also allows the OSN to cope with the multi-community structure of online social graphs [54] (§4.2.2).

We show that SybilRank outperforms existing approaches [23,33,38,53,57]. In our experiments, it detects Sybils with at least 20% lower false positive and negative rates (§6) than the second-best contender in most of the attack scenarios. We also deployed SybilRank on Tuenti, the leading OSN in Spain (§7). Almost 100% and 90% of the 50K and 200K accounts, respectively, that SybilRank designated as most suspicious, were indeed fake. This compares very favorably to the ∼5% hit rate of Tuenti's current abuse-report-based approach.

Our contributions. In summary, this work makes the following contributions:

• We re-formulate the problem of Sybil detection in OSNs. We observe that due to the high false positives of binary Sybil/non-Sybil classifiers, manual inspection needs to be part of the decision process for suspending an account. Consequently, our aim is to efficiently derive a quality ranking, in which a substantial portion of Sybils ranks low. This enables the OSN to focus its manual inspection efforts towards the end of the list, where it is more likely to encounter Sybils. Moreover, our ranking can inform the OSN on whether to challenge suspicious users with CAPTCHAs.

• The design of SybilRank, an effective Sybil-likelihood ranking scheme for OSN users based on the landing probability of short random walks. We efficiently compute this landing probability using early terminated power iteration. In addition, SybilRank copes with the multi-community structure of social graphs [54] using multiple seeds at no additional computational cost. We thoroughly compare SybilRank with existing Sybil detection schemes.

• We deployed SybilRank on Tuenti, an OSN with 11M users. Our system has allowed Tuenti operations to detect in the same time period 18 times more fake accounts than their current process.

2 Background and Related Work

We have conducted a survey with Tuenti [4] to understand the activities of fake accounts in the real world, the importance of the problem for them, and what countermeasures they currently employ. In the rest, we discuss our findings and prior OSN Sybil defense proposals.

2.1 Fake accounts in the real world

Fake accounts are created for profitable malicious activities, such as spamming, click-fraud, malware distribution, and identity fraud [7,12]. Some fakes are created to increase the visibility of niche content, forum posts, and fan pages by manipulating votes or view counts [6, 9].

People also create fake profiles for social reasons. These include friendly pranks, stalking, cyberbullying, and concealing a real identity to bypass real-life constraints [9]. Many fake accounts are also created for social online games [9].

2.2 Feature-based approaches

The large number of distinct reasons for the creation of Sybil accounts [6, 7, 9] results in numerous and diverse malicious behaviors manifested as profile features and activity patterns. Automated feature-based Sybil detection [55] (e.g., Machine-Learning-based) suffers from high false negative and positive rates due to the large variety and unpredictability of legitimate and malicious OSN users' behaviors. This is reminiscent of how ML has not been very effective in intrusion detection [49].

High false positives in particular pose a major obstacle in the deployment of automated methods, as real users respond very negatively to erroneous suspension of their accounts [10]. To make matters worse, attackers can always adapt their behavior to evade detection by automated classifiers. Boshmaf et al. [19] recently showed that the automated Facebook Immune System [50] was able to detect only 20% of the fakes they deployed, and almost all the detected accounts were flagged by users.

2.3 Human-in-the-loop counter-measures

In response to the inapplicability of automated account suspension, OSNs employ CAPTCHA [5] and photo-based social authentication [13] to rate-limit suspected users, or manually inspect the features of accounts reported as abusive (flagged) by other users [12, 51, 55].

The manual inspection involves matching profile photos to the age or address, understanding natural language in posts, examining the friends of a user, etc [12]. The inspectors also use simple tools to parse activity and IP statistics of suspicious accounts. However, these tasks require human intelligence and intuition, rendering them hard to automate and scale. In addition, flagged accounts have by definition already caused annoyance to some users. Therefore, the need arises for a method that en-

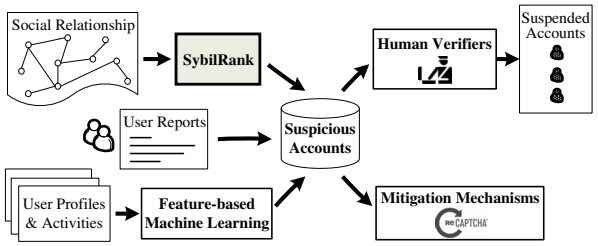


Figure 1: SybilRank as a part of an OSN's Sybil defense toolchain.

ables human verifiers to focus on accounts that are very likely to be fake and to disable them in a timely manner.

Tuenti receives on average 12,000 reports regarding abusive accounts and 4,000 reports for violating photos per day. An employee can review an average of 250 to 350 reports in an hour. Among those reported suspicious accounts, only $\sim$5% are indeed fake [12]. On average, Tuenti's manual inspection team, which consists of 14 employees, deletes 800 to 1500 fake accounts per day.

Our solution can be a component of the overall framework employed by OSN providers to ensure the health of their user base (Figure 1). It is a proactive method that can uncover fakes even before they interact with real users. It complements ML- and user-report-based methods. The users that these methods designate as suspicious are typically challenged in the form of CAPTCHAs or are manually inspected.

2.4 Social-graph-based approaches

Sybil detection has been the focus of the research community for a while now [23, 52–54, 57, 58]. However, *none of the existing proposals can play the role of SybilRank in Figure 1*. The reason is that prior schemes either do not exhibit equivalent accuracy or they incur a prohibitive computational cost. Their key features are documented in a recent survey [56]. We next compare them to our proposal.

The decentralized protocols SybilGuard [58] and SybilLimit [57] infer Sybils based on a large volume of random walk traces, which leads to an $O(\sqrt{m}n \log n)$ (m is the number of edges, n is the number of nodes) computational cost in a centralized setting. SybilInfer [23] is also built on random walk traces with $O(n(\log n)^2)$ cost per honest seed, but does not specify any upper-bound guarantee on false rates [56]. Mohaisen et al. [40] propose to use weighted random walks in SybilLimit. However, they follow the random walk sampling paradigm instead of our power-iteration method (§4.2), incurring a cost as high as SybilLimit.

The flow-based scheme GateKeeper [53] improves over SumUp [52]. It relies on a strong assumption that requires balanced graphs [56] and costs $O(sn \log n)$ (s is the number of seeds, referred to as ticket sources). Viswanath et al. [54] propose community detection algorithms such as Mislove's algorithm [38] to detect Sybils. However, community detection (CD) algorithms rarely provide provable guarantees, and Mislove's CD algorithm costs $O(n^2)$. Compared to the above schemes, SybilRank achieves equivalent or higher accuracy (analytically), but it is computationally more efficient, i.e., it has $O(n \log n)$ cost irrespective of the number of seeds.

Moreover, SybilRank addresses important limitations of existing social-graph-based defenses [54]. First, SybilRank leverages its efficient support for multiple trust seeds to reduce the false positives resulting from the existence of multiple non-Sybil communities. Second, it enables very flexible seed selection (any non-Sybil node can be a seed), which makes it harder for attackers to target the seeds. In addition, SybilRank's effectiveness decreases only moderately as the Sybil's distance from the trust seeds decreases (§6.5).

Zhao et al.'s [61] BotGraph detects spam webmail user-bots by leveraging the fact that they are highly correlated in terms of the IP address of their controllers. The same mechanism can be used in social graphs to uncover Sybils that form tight-knit communities. Tangentially, Yang et al. [55] recently analyzed a sample of 660K Sybils in the RenRen OSN, and found that they do not form tight-knit communities. Regardless, SybilRank and the schemes in [23, 53, 54, 58] rely on the assumption that the connectivity between Sybils and non-Sybil users is limited and are not very sensitive to the number of Sybils or the inter-Sybil connectivity.

Sybil-resilient systems [27, 36, 37, 42, 44, 46–48, 59] leverage application-specific knowledge to effectively mitigate Sybils, e.g., to limit the number of votes collected from Sybil users [52]. However, a Sybil-resilient design optimized for an application may not be applicable to other systems, while a social-graph-based Sybil inference mechanism can be applied to any application that binds its users to OSN accounts.

3 Models, Assumptions, and Goals

We now introduce our system and threat model, and our design assumptions and goals.

3.1 System and threat model

We consider bilateral social relationships and model an OSN as an *undirected* graph $G = (V, E)$, where each node in V corresponds to a user in the network, and each edge in E corresponds to a bilateral social relationship. In the social graph G, there are $n = |V|$ nodes, $m = |E|$ undirected edges, and a node v has a degree of $deg(v)$. We assume that the OSN provider, e.g., Tuenti, has access to the entire social graph.

In our threat model, attackers can launch Sybil attacks [21, 26] by creating many fake identities. Like ex-

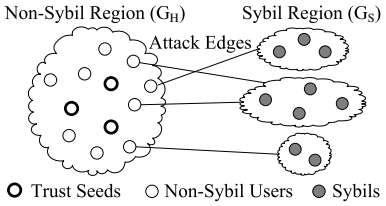


Figure 2: Non-Sybil region, Sybil region, and attack edges in an OSN under a Sybil attack. All Sybils created by malicious users are placed into the Sybil region. The Sybil collective may not be well connected.

isting work [23,57,58], we divide the node set V into two disjoint sets H and S, representing non-Sybil and Sybil users respectively, as shown in Figure 2. We denote the *non-Sybil region* G_H as the subgraph induced by the non-Sybil user set H, which includes all non-Sybil users and the links among them. Similarly, the *Sybil region* G_S is the subgraph induced by S. The non-Sybil and the Sybil regions are connected by g *attack edges* between Sybils and non-Sybil users.

3.2 Assumptions

We make the following assumptions.

Social graph. The social graph is undirected. The non-Sybil region G_H is well connected and non-bipartite. In this case, random walks on the graph can be modeled as an irreducible and aperiodic Markov chain [16]. This Markov chain is guaranteed to converge to a stationary distribution in which the landing probability on each node after sufficient steps is proportional to its degree.

Limited attack edges. We assume that Sybils establish a limited number of attack edges due to the difficulty of soliciting and maintaining reciprocal social relationships. Although previous studies [17, 19] suggest that fake identities can befriend others, a recent study shows that most of their connections are established with other fakes [43]. SybilRank is designed for large scale attacks, where fake accounts are crafted and maintained at a low cost, and are thus unable to befriend many real users. Furthermore, SybilRank can be deployed over a social graph that includes only strong-relationship edges. For instance, Google+ may consider only social connections between users that appear in mutually close circles.

The limited attack edges result in a sparse cut between the non-Sybil and the Sybil region. Since the well-connected non-Sybil region is unlikely to have such a sparse cut [58], there should be a significant difference between the mixing time of the non-Sybil region G_H and the entire graph G [16]. A graph's *mixing time* is the maximum number of steps that a random walk needs to make so that the probability of landing at each node reaches the stationary distribution [16]. As in previous work [23, 57, 58], we start with the assumption that the non-Sybil region G_H is fast mixing, i.e., its mixing time is $O(\log n)$, and discuss the scenarios where the non-Sybil region mixes slower in §4.2.3.

Attack edge distribution. To make our analysis tractable, we assume that Sybils randomly attach attack edges to non-Sybils. A more effective attack strategy against a social-graph-based Sybil defense is to establish attack edges close to the trust seeds. We refer to this attack as a *targeted attack*. We experimentally study how SybilRank performs under the targeted attack in § 6.5.

3.3 Goals

SybilRank aims to aid OSNs in identifying highly suspicious accounts by ranking users. In principle, SybilRank can be used to defend against malicious accounts that are created on a large scale and at a low per-account cost for purposes, such as rating manipulation (e.g., with Facebook "likes"), spam, and crawls. Our design has the following main goals:

Effectiveness. The system should mostly rank nodes that are Sybils lower than non-Sybils (low false positives), while limiting the number of non-Sybils ranked below Sybils (low false negatives). It should be robust under various attack strategies. A very high portion of the nodes at the bottom of the ranked list should be fake. The portion of fakes can decrease as we go up the list.

Efficiency. The system should have a low computation cost. It should be able to handle large social networks with commodity machines, so that OSN providers can deploy it on their existing clusters.

4 Design

We now describe the design of SybilRank. We first provide an overview of the approach and proceed with a detailed description of each component.

4.1 Overview

SybilRank relies on the observation that an *early-terminated* random walk [56] starting from a non-Sybil node in a social network has a higher degree-normalized (divided by the degree) landing probability to land at a non-Sybil node than a Sybil node. Intuitively, this observation holds because the limited number of attack edges forms a narrow passage from the non-Sybil region to the Sybil region in a social network. When a random walk is sufficiently long, it has a uniform degree-normalized probability of landing at any node, a property referred to as the convergence of a random walk [16]. However, a shortened random walk originating from a non-Sybil node tends to stay within the non-Sybil region of the network, as the walk is unlikely to traverse one of the relatively few attack edges.

Our key insight is to rank nodes in a social graph according to the degree-normalized probability of a short random walk that starts from a non-Sybil node to land on them. We screen out low-ranked nodes as potential

fake users. A further novelty of our approach is that we use power iteration [34], a standard technique to efficiently calculate the landing probability of random walks in large graphs. This is in contrast to prior work that used a large number of random walk traces [23, 40, 56–58] obtained at a high computational cost. For ease of exposition, we refer to the probability of the random walk to land on a node as the node's *trust*.

As shown in Figure 3, SybilRank unveils users that are suspected to be Sybils after three stages. In Stage I, through $w = O(\log n)$ power iterations (§4.2), trust flows from known non-Sybil nodes (trust seeds) and spreads over the entire network with a bias towards the non-Sybil region. In Stage II, SybilRank ranks nodes based on their degree-normalized trust. In the final stage, SybilRank assigns portions of fake nodes in the intervals of the ranked list. This enables OSNs to focus their manual inspection efforts or to regulate the frequency with which they send CAPTCHAs to suspected users.

We next describe each stage in detail.

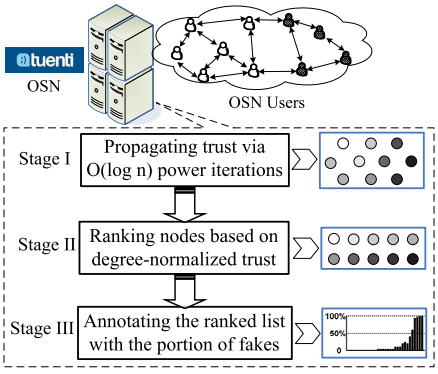


Figure 3: SybilRank detects Sybils in three stages. Black users are Sybils. Darker nodes obtain less trust after trust propagation.

4.2 Propagating trust

Power iteration involves successive matrix multiplications where each element of the matrix represents the random walk transition probability from one node to a neighbor node. Each iteration computes the landing probability distribution over all nodes as the random walk proceeds by one step.

4.2.1 Early-terminated random walks

We terminate the power iterations after $O(\log n)$ steps. The number of iterations needed to reach the stationary distribution is equal to the graph's mixing time. In an undirected graph, if a random walk's transition probability to a neighbor node is uniformly distributed, the landing probability on each node remains proportional to its degree after reaching the stationary distribution. SybilRank exploits the mixing time difference between the non-Sybil region G_H and the entire graph G to distinguish Sybils from non-Sybils. The intuition is that if we seed all trust in the non-Sybil region, then trust can flow into the Sybil region only via the limited number of attack edges. If we terminate the power iteration early before it converges globally, non-Sybil users will obtain higher trust than that in the stationary distribution, whereas the reverse holds for Sybils.

Trust propagation via power iteration. We define $T^{(i)}(v)$ as the trust value on node v after i iterations. Initially, the total trust, denoted as T_G ($T_G > 0$), is evenly distributed on K ($K > 0$) trust seeds $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_K$:

$$T^{(0)}(v) = \begin{cases} \frac{T_G}{K} & \text{if node } v \text{ is one of the } K \text{ trust seeds} \\ 0 & \text{else} \end{cases}$$

Seeding trust on multiple nodes makes SybilRank robust to seed selection errors, as incorrectly designating a node that is Sybil or close to Sybils as a seed causes only a small fraction of the total trust to be initialized in the Sybil region.

During each power iteration, a node first evenly distributes its trust to its neighbors. It then collects trust distributed by its neighbors and updates its own trust accordingly. The process is shown below. Note that the total amount of trust T_G remains unchanged.

$$T^{(i)}(v) = \sum_{(u,v)\in E} \frac{T^{(i-1)}(u)}{deg(u)}$$

Early termination. With sufficiently many power iterations, the trust vector converges to the stationary distribution: $\lim_{i\to\infty} T^{(i)}(v) = \frac{deg(v)}{2m} \times T_G$ [16]. However, SybilRank terminates the power iteration after $w = O(\log n)$ steps, thus before convergence. This number of iterations is sufficient to reach an approximately uniform distribution of degree-normalized trust over the fast-mixing non-Sybil region, but limits the trust escaping to the Sybil region.

Alternative use of power iteration. We note that power iteration is also used by the trust inference mechanisms PageRank, EigenTrust, and TrustRank [22, 30, 33, 45], where it is executed until convergence to the stationary distribution of the complete graph [34]. At each step and with a constant probability, PageRank's random walks jump to random users, and EigenTrust/TrustRank's walks jump to trust seeds. EigenTrust/TrustRank is personalized PageRank with a customized reset distribution over a particular set of seeds.

SybilRank's random walks do not jump to random users, because this would allow Sybils to receive trust in each iteration. Besides, SybilRank's random walks do not jump to the trust seeds, because this would assign high trust to Sybils that are close to the trust seeds (§6.3).

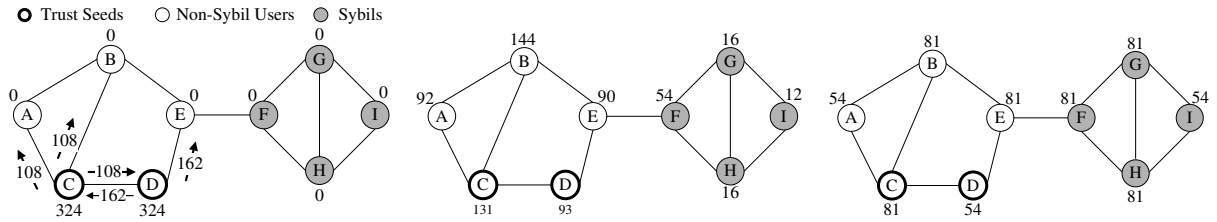


Figure 4: Trust propagation through power iterations.

In addition, we do not seek to improve the Sybil resilience of power-iteration-based trust inference mechanisms by enforcing early termination. This is because with their implicit directed connections to random nodes (PageRank) and trust seeds (EigenTrust and TrustRank), the mixing time of their modified graphs decreases significantly. A formal analysis on the convergence of random walks on these graphs is documented in [34]. Empirical reports show that EigenTrust only takes 10 iterations to converge on a 1000-node graph [33]. Therefore, early termination after $O(\log n)$ steps cannot improve those schemes.

Example. Figure 4 illustrates the process of our trust propagation. In this example, we initially seed $T_G = 648$ on the non-Sybil nodes C and D. We do not normalize T_G to 1 for ease of exposition. After four power iterations, all non-Sybil nodes $\{A, B, C, D, E\}$ obtain higher degree-normalized trust than any of the Sybils $\{F, G, H, I\}$. However, as shown in Figure 4(c), if we let the power iteration converge (more than 50 iterations), the Sybils have similar trust as the non-Sybil nodes, and each node's trust is only proportional to its degree.

4.2.2 Coping with the multi-community structure

Existing social-graph-based defenses are sensitive to the multi-community structure in the non-Sybil region [54]. Due to the limited connectivity between communities, they may misclassify non-Sybils that do not belong to the communities of the trust seeds as Sybils.

Distributing seeds among communities. SybilRank intends to reach a uniform distribution of degree-normalized trust among the non-Sybil region, which is independent of the location of the non-Sybil seeds. Thus, any set of non-Sybil users are eligible for the trust seeds. Seeding trust on Sybils would degrade SybilRank's effectiveness as the initial trust on the Sybil seeds is not bounded by the limited attack edges. OSN providers can easily identify non-Sybil users to seed trust by manually inspecting a few users. SybilRank works with an arbitrary number of non-Sybil seeds regardless of their locations. In contrast, the seed selection is complicated for previous trust inference schemes (§4.2.1) and Sybil defenses such as SumUp [52] because they have to guarantee that the seed(s) are "far" from Sybils.

We leverage SybilRank's support for multiple seeds to improve its performance in OSNs that have a multi-community structure. The key idea is to distribute seeds among the communities. Thus, at initialization we distribute trust in such a manner that the non-Sybil users are not starved of trust even if the inter-community connectivity is weak. We validate this design choice with simulations in §6.4, where SybilRank maintains a high detection accuracy in synthetic graphs with high-level community structure.

Inspecting random users for trust seeds to cover most of communities in the non-Sybil region would require a prohibitive manual inspection workload, as indicated by the Coupon Collector's Problem [39]. Instead, we apply an efficient community detection but not Sybil-resilient algorithm to obtain an estimation of the community structure [18], and then seed trust on non-Sybil users in each major community.

Estimating the multi-community structure. We use the Louvain method [18], which can efficiently detect communities. This method iteratively groups closely connected communities together to improve the partition modularity. In each iteration, every node represents a community, and well-connected neighbor nodes are combined into the same community. The graph is reconstructed at the end of each iteration by converting the resulting communities to nodes and adding links that are weighted by the inter-community connectivity. Each iteration has a computational cost linear to the number of edges in the corresponding graph and a small number of iterations are typically required. In addition, this method can be parallelized on commodity machines [60].

As illustrated in Figure 5, in each identified community we inspect a small set of random nodes ($\sim$100 in total for the 11-million-node Tuenti social network) and only seed trust at the nodes that pass the verification.

Community detection algorithms have been proposed to directly detect Sybils [54]. They seek a partition of a graph that has dense intra-community connectivity and weak inter-community connectivity. For instance, the Louvain method searches for a partition with high modularity [18]. Thus, Sybils that are not well connected among each other may be classified as non-Sybils belonging to nearby non-Sybil communities. In contrast,

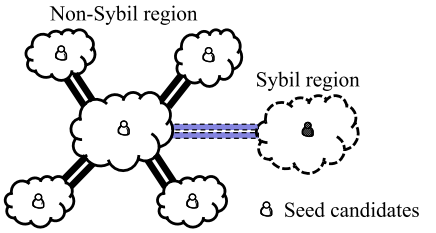


Figure 5: Distributing seeds into multiple communities. The Sybil seed candidates (black) cannot pass the manual inspection.

by placing seeds in communities SybilRank is able to uncover subsets of Sybils within Louvain-detected communities, as shown in our Tuenti deployment (§7.2).

4.2.3 Sensitivity to the mixing time

Although for analytical tractability, we assume that the non-Sybil region is fast mixing as in previous work [23, 53, 57], SybilRank's effectiveness does not rely on the absolute value of the non-Sybil region's mixing time. Instead, it only requires that the graph G containing both Sybils and non-Sybils has a longer mixing time than the non-Sybil region G_H. Under this condition, the early-terminated power iteration yields a gap between the degree-normalized trust of non-Sybils and Sybils.

Ideally, the number of iterations that SybilRank performs is set equal to the mixing time of the non-Sybil region. Previous Sybil defenses [23, 57] assume that the mixing time of social networks is $O(\log n)$. However, measurement studies [25, 41] show that the mixing time of some social networks is longer than expected. It is unclear whether this increase derives from a large coefficient in front of the term $\log n$, or the possibility that the mixing time of social networks is not $O(\log n)$.

In SybilRank we simply use $O(\log n)$ power iterations. If the mixing time of the non-Sybil region is larger than this value, the trust that escapes to the Sybil region is further limited. However, we also run the risk of starving the non-Sybil users that are not well-connected to seeds. This risk is mitigated by placing seeds in many communities and by dispersing multiple seeds in each community (§4.2.2), thereby ensuring that the trust is initiated somewhere close to those non-Sybil users.

4.3 Ranking by degree-normalized trust

After $w = O(\log n)$ power iterations, SybilRank ranks nodes by their degree-normalized trust. A node v's degree-normalized trust is defined as: $\hat{T}_v = \frac{T^{(w)}(v)}{deg(v)}$. We rank nodes by degree-normalized trust for the following two reasons.

Eliminating the node degree bias. This design gives each non-Sybil node almost identical degree-normalized trust. It reduces false positives from low-degree non-Sybil nodes and false negatives from high-degree Sybils. This is because after $O(\log n)$ power iterations, the trust distribution in the non-Sybil region approximates the stationary distribution in that region, i.e., the amount of trust on each non-Sybil node is proportional to its degree.

The removal of the degree bias simplifies the Sybil/non-Sybil classification, i.e., all non-Sybil users are supposed to belong to the same class with an almost identical degree-normalized trust. This is in contrast to prior trust inference schemes [30, 33, 45] that attempt to differentiate between Sybils and non-Sybils with highly variable trust scores.

Bounding highly-ranked Sybils. The degree normalization step ensures that the number of Sybils ranked higher than the non-Sybil nodes is $O(\log n)$ per attack edge, as we explain in §4.6.

4.4 Annotating the ranked list

SybilRank relies on the limited attack edges to distinguish Sybils. It may give high rankings to the Sybils that obtain many social links to non-Sybil users, and consider them as non-Sybil users. This is true for all social-graph-based defenses. It may result in even the top and bottom intervals of the ranked list to comprise a non-negligible portion of fake and real accounts, respectively. Thus, an OSN cannot simply identify a pivot in the ranked list below which all nodes are fake, and it still has to rely on manual inspection.

At the same time, OSNs have limited resources for manual inspection. To aid OSNs to adjust their inspection focus, we annotate intervals in the ranked list with fake portions. In particular, we sample random users in each interval of a particular size and report a portion of fakes after manual inspection. With these annotations, OSNs can decide where on the ranked list to assign their limited human verifiers. The annotations can also be used to regulate the frequency of CAPTCHAs and other challenges sent to the suspected users. Moreover, the annotations can help OSNs to decide on accounts that do not exhibit sufficient evidence on whether they are fake.

4.5 Computational cost

SybilRank's computational cost is $O(n \log n)$. This is because each power iteration costs $O(n)$, and we iterate $O(\log n)$ times. The cost for ranking the nodes according to their degree-normalized trust is also $O(n \log n)$. The cost for estimating communities is $O(m)$, because each iteration in Louvain method has a computational cost linear to the number of edges and the graph shrinks rapidly with only a few iterations. Since the node degree in OSNs is always limited, the community estimation costs $O(n)$. Thus the overall computational cost is $O(n \log n)$, irrespective of the number of trust seeds.

4.6 Security Guarantee

We now discuss SybilRank's security guarantee under the assumption that the attack edges are randomly es-

tablished between non-Sybils and Sybils. Although our system does not depend on the absolute mixing time of the non-Sybil region (§4.2.3), our analysis assumes that the non-Sybil region is fast-mixing. Due to space limitations we only present the conclusion and its high-level intuition, and refer the reader to our technical report [20] for the complete proof. We use the notation in §3.1.

Theorem 1. *When an attacker randomly establishes g attack edges in a fast mixing social network, the total number of Sybils that rank higher than non-Sybils is $O(g \log n)$.*

After early termination, the trust distribution in the entire graph G has not reached its stationary distribution. Since trust propagation starts from the non-Sybil region, the Sybil region (on the aggregate) gets only a fraction $f < 1$ of the trust it should obtain in the stationary distribution. On the other hand, as the total trust is conserved in the entire graph, the non-Sybil region obtains an aggregate amount of trust that is c ($c > 1$) times higher than in the stationary distribution. Further, since the non-Sybil region is well connected, each non-Sybil node obtains approximately identical degree-normalized trust, i.e., $c \times \frac{T_G}{2m}$, where $\frac{T_G}{2m}$ is a node's degree-normalized trust in the stationary distribution (§4.2).

The amount of degree-normalized trust obtained by each Sybil depends on how Sybils connect to each other. However, since the aggregate amount of trust of the Sybil region is bounded, on average, each Sybil obtains $f \times \frac{T_G}{2m}$ degree-normalized trust, which is less than that of a non-Sybil node. We are able to show that at most $O(\log n)$ Sybils per attack edge obtain higher degree-normalized trust than non-Sybil nodes [20]. It is worth noting that SybilRank is able to provide this guarantee even when it uses an arbitrary number of trust seeds.

5 MapReduce Implementation

We now briefly describe how we implement SybilRank using the Hadoop [1] MapReduce [24] parallel computing framework. This implementation enables an OSN provider to process social network graphs with hundreds of millions of users on a cluster of commodity machines.

We divide the entire graph into multiple partitions so that each of them fits into the hardware of a commodity machine. The complexity of SybilRank is dominated by the first two stages (Figure 3): trust propagation and node ranking. Together they have $O(n \log n)$ complexity. For the trust propagation stage, we observe that trust splitting and trust aggregation at each iteration are inherently parallelizable. Therefore, we treat each iteration as a MapReduce job, and create multiple map tasks to split trust and multiple reduce tasks to aggregate trust simultaneously. For the node ranking stage, we use Hadoop's built-in sorting feature.

Efficiency. We evaluate the efficiency of our prototype on an Amazon EC2 cluster to process very large-scale synthetic graphs with hundreds of millions of nodes. The cluster consists of 11 m1.large instances, one of which serves as the master and the other 10 as slaves.

We generate very large synthetic graphs based on the scale-free model as in §6.1. The synthetic graphs are generated with exponentially increasing sizes from 10M to 160M nodes. SybilRank performs successfully on each graph with $\log n$ power iterations. The total execution time includes two parts: a) the time to seed trust and partition the graph during initialization; and b) the time to execute power iterations to propagate trust and to rank the final results. The latter dominates the total execution time, which increases almost linearly with the size of the social graphs (see [20]). For the largest graph (160M nodes), our prototype finishes in less than 33 hours. This result suggests that SybilRank can process very large social graphs using a few commodity machines.

6 Sybil Ranking Evaluation

In this section, we perform a comparative evaluation of SybilRank's ability to provide a meaningful ranking of nodes to uncover Sybils. We first compare SybilRank against other approaches in terms of its effectiveness in assigning low ranking to Sybils. We then examine the SybilRank's component that copes with the multi-community structure, and we study the resilience of our approach to attacks that target the seeds. For a fair comparison, we do not use SybilRank's component for coping with the multi-community structure except for §6.4.

6.1 Simulation setup

Compared approaches. We compare SybilRank (SR) against the state-of-the-art social-graph-based Sybil defenses, i.e., SybilLimit (SL) [57], SybilInfer (SI) [23], Mislove's community detection [38] (CD), and GateKeeper (GK) [53]. Importantly, we also compare to EigenTrust (ET) [33], which uses power iteration to assign trust.

Datasets. The non-Sybil regions of the simulated social graphs, which comprise exclusively non-Sybils, are samples of several popular social networks (Table 1). The Facebook graph [29] is a connected component sampled via the "forest fire" sampling method [35]. The synthetic graph is generated using Barabasi's scale-free model [15]. The rest of the graphs [3] have been widely used to study social graph properties and to evaluate recent Sybil defense mechanisms [41, 54].

Attack strategies. We create a $5K$-node Sybil region that connects to a non-Sybil region through a varying number of random attack edges. We choose this large

number of Sybils to stress-test each scheme. To investigate the schemes' robustness to the formation of the Sybil collective, we include two representative Sybil region structures: regular random graphs and scale-free graphs [15]. We call the first attack *regular attack*: each Sybil establishes connections to d random Sybils. We refer to the second attack as a *scale-free attack*: each Sybil preferentially connects to d Sybils upon its arrival, with the probability of connecting to a Sybil proportional to the Sybil's degree. We set $d = 4$ in both attacks.

Social Network	Nodes	Edges	Clustering Coefficient	Diameter
Facebook	10,000	40,013	0.2332	17
ca-AstroPh	18,772	198,080	0.3158	14
ca-HepTh	9,877	25,985	0.2734	18
Synthetic	10,000	39,399	0.0018	7
wiki-Vote	7,115	100,736	0.1250	7
soc-Epinions	10,000	222,077	0.0946	6
soc-Slashdot	10,000	153,404	0.0582	4
email-Enron	10,000	105,343	0.1159	6

Table 1: Social graphs used in our experiments. The last three graphs are 10K-node BFS samples.

Performance metrics. Our evaluation is based on a framework [54] that reduces defense schemes to a general model: producing a trust-based node ranking. The conversion for SybilLimit, SybilInfer, and CD is documented in [54]. For GateKeeper, we rank nodes by the number of tickets that each node obtains. For EigenTrust, we rank nodes according to their trust scores.

We use three metrics to compare the node ranking: the area under the Receiver Operating Characteristic (ROC) curve [31], the false positive rates, and the false negative rates. The ROC curve exhibits the change of the true positive rate with the false positive rate as a pivot point moves along the ranked list: a node below the pivot point in the ranked list is determined to be a Sybil; if the node is actually a non-Sybil, we have a false positive. The area under the ROC curve measures the overall quality of the ranking, i.e., the probability that a random non-Sybil node is ranked higher than a random Sybil. It ranges from 0 to 1, with 0.5 indicating a random ranking. An effective Sybil detection scheme should achieve a value > 0.5. Given a node ranked list, sliding the pivot point regulates the trade-off between the two false rates. We set the pivot point based on a fixed value for one false rate and compute the other false rate. We set the fixed false rate equal to 20%. In the real world, OSNs do not need a pivot point because none of the defenses so far can yield a binary Sybil/non-Sybil classifier with an acceptable false positive rate.

Trust seed selection. For a fair comparison, we strive to use the same trust seeds for all schemes in each simulation on each social network. For schemes that use a single seed, we randomly pick a node from the top-10 non-Sybil nodes that have the highest degree. For schemes supporting multiple seeds at one run, i.e., SybilRank, EigenTrust, and GateKeeper, we use 50 trust seeds. One seed is the same top-10 degree node as the one used in the single-seed schemes, and the other 49 seeds are randomly chosen from the non-Sybil nodes.

Other simulation settings. We perform $\log n$ power iterations for SybilRank, where n is the size of the social graph. We run EigenTrust until convergence with a reset probability 0.15, as in [33]. For each attack scenario, we average the results over 100 runs.

6.2 Ranking Quality Comparison

To compare the Sybil defense schemes, we start with a few attack edges and increase the number to a sufficiently large value such that the detection accuracy of each scheme degrades significantly. We show representative simulation results in Figure 6 and refer the reader to our technical report for the complete results [20]. As can be seen, when the number of attack edges is small, most of the schemes perform well and Sybils can be distinguished from non-Sybils by connectivity.

SybilRank. SybilRank outperforms all other schemes. It achieves the highest value of the area under the ROC curve and the lowest false positive and false negative rates. For the Facebook graph under the regular attack, even if the $5K$-node Sybil cluster obtains 1500 attack edges, a non-Sybil node has a probability of 70% to rank higher than a random Sybil as indicated by the value of the area under the ROC curve.

SybilLimit. SybilLimit outperforms most other schemes, but it performs worse than SybilRank. We believe that this is due to the different use of random walks, i.e., SybilLimit uses random walk traces, while SybilRank uses the power-iteration-computed landing probability. SybilLimit's security guarantee only limits the Sybils accepted by the verifier's (trust seed's) random walks that never cross any attack edge to the Sybil region [56]. However, the accepted Sybils cannot be bounded if a verifier's random walk enters into the Sybil region. In contrast, SybilRank allows trust to escape to the Sybil region, but does not accept a Sybil unless it gets higher degree-normalized trust than non-Sybil users.

GateKeeper. It performs worse than both SybilRank and SybilLimit, although it bounds the accepted Sybils to $O(\log g)$ per attack edge, where g is the total number of attack edges. This is because this bound comes from a strong assumption that does not always hold in real social networks: with high probability, a breadth-first search starting from a non-Sybil user and visiting at most $n/2$ nodes covers a large fraction of non-Sybil users [56].

SybilInfer. We observe a steep fall in the area under the ROC curve for SybilInfer when the number of at-

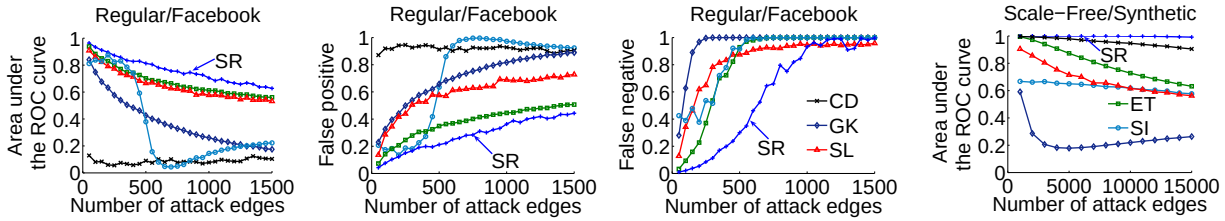


Figure 6: We depict the area under the ROC curve, the false positive rate, and the false negative rate with respect to the number of attack edges under the regular attack and the scale-free attack for various Sybil detection schemes in the Facebook and the scale-free synthetic graphs. In the ROC curve figures (a and d), a higher curve indicates a more effective scheme. In the false rate figures (b and c), a lower curve indicates a more effective scheme. SR stands for SybilRank; CD for the community detection algorithm, GK for GateKeeper; SI for SybilInfer; ET for EigenTrust; and SL for SybilLimit.

tack edges is close to 500 in Facebook under the regular attack. We suspect that this sharp performance degradation is due to the fact that SybilInfer uses the Metropolis-Hastings (MH) algorithm to sample the non-Sybil node set [23]. However, it remains unclear when the sampling converges, although Danezis et al. provide an empirical estimation and terminate the sampling after $O(n \log n)$ steps. If the Sybils obtain many attack edges and become hard to be detected, $O(n \log n)$ steps may not suffice to reach the convergence of the MH sampling and therefore, the detection accuracy is likely to degrade.

Mislove's CD. It underperforms under the regular attack, with <0.2 under the ROC curve area (Figure 6(a)). It interestingly becomes effective under the scale-free attack in the synthetic graph (Figure 6(d)). This significant performance difference is due to the greedy search for the local community, which is sensitive to the graph topology and cannot provide a false rate bound [56].

Although Mislove's CD algorithm was intended to be used with one seed, it can support multiple seeds at the same cost: by initializing the local community search with those seeds. In §6.4 we show that this extension can improve its performance to some extent.

EigenTrust. EigenTrust improves over PageRank [45], and uses the same basic mechanism as TrustRank [30]. We can see that in Figure 6(a) EigenTrust mostly outperforms previously proposed Sybil defenses. However, it has at least 20% higher false positive and negative rates than SybilRank in most of the attack scenarios.

6.3 Comparison with EigenTrust

EigenTrust is more related to SybilRank because it also uses power iteration. By further investigating EigenTrust we reveal the importance of SybilRank's two main differentiating characteristics: a) removal of degree bias (§4.3); and b) early termination and not jumping back to trust seeds (§4.2.1).

Impact of the connectivity of the Sybil region. We examine how the connection density within the Sybil region impacts the node ranking generated by EigenTrust and SybilRank. To do so, we vary the number of edges

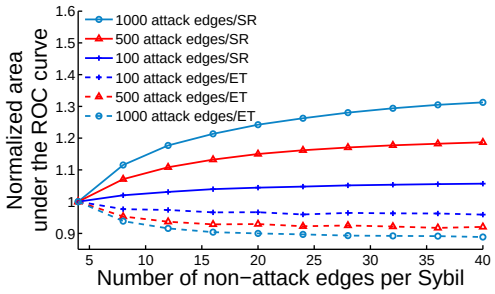


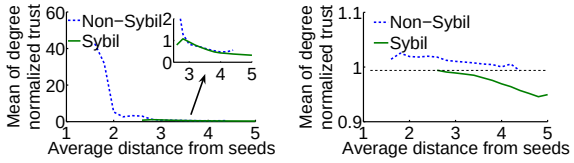
Figure 7: Normalized area under the ROC curve as a function of the number of edges connecting to other Sybils.

each Sybil has with other Sybils from 4 to 40 under a regular attack on the Facebook graph. We refer to such edges as *non-attack*.

Figure 7 shows the *normalized area under the ROC curve*, which is computed by dividing the area under the ROC curve for each attack scenario with the baseline case where each Sybil has only 4 non-attack edges. As can be seen in Figure 7, with EigenTrust, the value of the area under the ROC curve decreases when the connections within the Sybil region become dense (the normalized area under the ROC curve is always less than 1). This result is because in EigenTrust a node that has many incoming links or is pointed to by other highly-ranked nodes is typically ranked high [33, 45]. A malicious user can simply create dense connections within the Sybil region to boost the ranks of selected Sybils. Therefore, denser Sybil connections lower EigenTrust's values of the area under the ROC curve, indicating that more Sybils are ranked higher than non-Sybils.

On the contrary with SybilRank, due to the removal of degree bias, high-degree Sybils do not benefit. Instead, SybilRank's performance improves as the connections among Sybils become denser. This is because it exploits the sparse cut between the Sybil and the non-Sybil region, which the addition of Sybil connections makes even sparser (it lowers the conductance).

Impact of the distance from trust seeds. EigenTrust's trust distribution is sensitive to the locations of the seeds because its random walks jump back to them.



(a) EigenTrust (b) SybilRank

Figure 8: Trust distribution with respect to the average distance to seeds in the synthetic graph under the regular attack with 10000 attack edges.

Figure 8 compares the distribution of *degree-normalized* trust generated by EigenTrust and SybilRank with respect to a node's average shortest hop distance from the trust seeds. We simulate a regular attack in the synthetic graph with $10K$ attack edges, and select 5 seeds for both EigenTrust and SybilRank. The total trust is set to $2m$. As shown in Figure 8(a), EigenTrust tends to allocate high degree-normalized trust to nodes close to the seeds. Nodes relatively farther from the seeds get substantially lower trust. In fact, the degree-normalized trust distribution in EigenTrust has a "heavy" tail. Among the $10K$ non-Sybil nodes, more than $9.4K$ have degree-normalized trust lower than 2. As we zoom into the tail, this large set of non-Sybil users and Sybils have indistinguishable degree-normalized trust .

Figure 8(b) shows that unlike EigenTrust, SybilRank assigns roughly the same degree-normalized trust to each non-Sybil node, while keeping a distinguishable gap between non-Sybils and Sybils. This empirically validates our observation that the degree-normalized trust is approximately uniformly distributed in the non-Sybil region after $O(\log n)$ power iterations (§4.3).

6.4 Coping with multiple communities

As discussed in §4.2.2, we distribute seeds among communities to cope with the multi-community structure in the non-Sybil region. We illustrate this with simulations on synthetic graphs. The simulations also include Mislove's CD and EigenTrust, which can be initialized with multiple seeds at no additional computational cost. We do not include GateKeeper because it uses random walks to seek seeds (ticket sources), thus we do not fully control the placement of seeds among the communities.

Similar to the simulation scenarios in [54], we set up a non-Sybil region consisting of 5 scale-free synthetic communities, each of which has 2000 nodes with an average degree of 10. We designate one community as the core of the social graph. The other 4 communities do not have any connections to each other, but connect to the graph core via only 500 edges. This process builds a non-Sybil region where multiple communities connect to the core community of the graph via limited links.

We contrast the following two seed selection strategies: a) all 50 seeds are confined to the core community; and b) the seeds are distributed among all the non-Sybil communities, each of which has 10 seeds. As shown in Figure 9, seed selection strategy (b) improves the detection accuracy for all schemes. Wide seed coverage in the non-Sybil region offsets the impact of its multi-community structure: the resulting ranking is mostly determined by the sparse cut between the non-Sybil region and the Sybil region. In Figure 9, we see that SybilRank maintains the highest accuracy for each of the seed placement strategies due to the disadvantages of EigenTrust and Mislove's CD mentioned in §6.3 and §6.2.

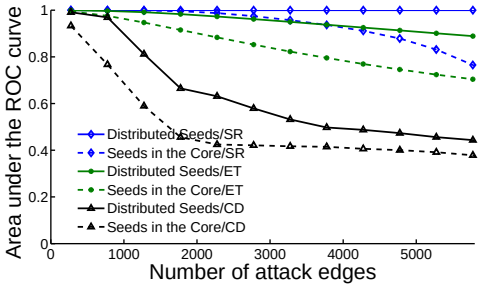


Figure 9: Comparing SybilRank, EigenTrust and Mislove's CD in a multi-community graph.

6.5 Resilience to seed-targeting attacks

Last, we study how various schemes perform under targeted attacks. In these attacks, sophisticated attackers may obtain partial or full knowledge about the OSN and discover the locations of the trust seeds. They can then establish attack edges to nodes close to the seeds.

We simulate the targeted attack in the Facebook graph. The 5K-node Sybil region forms a regular attack structure, and has 200 attack edges connecting to the non-Sybil region. Instead of being completely randomly distributed, those 200 attack edges are established by randomly connecting to the k non-Sybil nodes with the shortest distance from the trust seed. For schemes supporting multiple seeds, we target the attack edges to the highest-degree trust seed. We vary k from 1K to 10K. A smaller k signifies a shorter average distance between Sybils and seeds.

As shown in Figure 10, when attack edges are attached close to the seed, all schemes' performance degrades, while SybilRank keeps the most stable performance across a wide range of k values. This is because SybilRank does not assign excessive trust to nodes that are close to the seeds (§6.3). However, SybilRank's performance still degrades when k is small. This is because these closely targeted attacks force SybilRank to "leak" a fraction of trust in the Sybil region during early power iterations. Thus, its detection accuracy reduces as Sybils may gain higher trust than non-Sybils.

7 Real-world Deployment

We now discuss the deployment of our system on a snapshot of Tuenti's complete social friendship graph, which

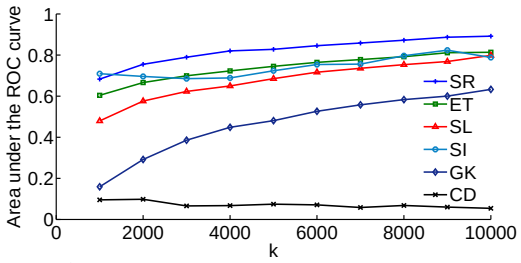


Figure 10: Detection effectiveness under the attacks targeting the k non-Sybils with the shortest distance to the seed.

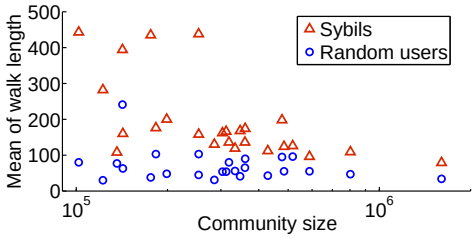


Figure 11: Contrasting the mean length of random walks from Sybils and from random users in each community.

was obtained in August 2011. Due to the sheer volume of users, it would be infeasible for us to manually inspect whether each user is a Sybil. Thus, we are unable to evaluate SybilRank with the same metrics as in the simulations (§6), such as the area under the ROC curve, the false negative rate and the false positive rate. Instead, we attempt to determine the portion of fake users at varying segments of the ranked list. We do so by manually inspecting a user sample in each particular interval in the ranked list (§4.4). Due to the practical constraints and the limited availability of human verifiers, we do not deploy the other Sybil defenses on Tuenti.

Pre-processing. We observe that in Tuenti some fake Tuenti accounts are well-maintained and have extremely high degree. They may introduce many attack edges if they connect to real users. At the same time, brand new real users always have weak connectivity to others due to the limited time they have been in the OSN, resulting in false positives. To reduce the impact of these two factors, we perform pre-processing before applying SybilRank: we prune the edges on extremely-high-degree nodes and defer the consideration of very recent users (see [20]).

Communities in Tuenti. The complete Tuenti social graph has 1,421,367,504 edges and 11,291,486 nodes, among which 11,216,357 nodes form a Giant Connected Component (GCC). Our analysis focuses on the GCC. With the Louvain method, we found 595 communities, among which 25 large communities contain more than 100K nodes. We inspected 4 nodes in each community and designated as SybilRank trust seeds the nodes that pass the manual verification.

7.1 Validating the mixing time assumptions

As discussed in §3.2, SybilRank relies on the mixing time gap between the non-Sybil region and the entire graph. Since we seed trust in each large community, the trust propagation is mainly determined by those communities. To this end, we investigate the 25 large communities identified by the Louvain method. As shown in §7.2, fake accounts are embedded in each community. We then measure the mixing time gap between each community and its non-Sybil part.

We measure the mixing time using its definition, i.e., the maximum necessary walk length to achieve a given variation distance [39] from the stationary distribution. Due to Tuenti's large population, it is difficult to remove all Sybils in order to measure the mixing time of the non-Sybil part in each community. Therefore, our measurement approximates the mixing time gap. We do so by contrasting the mean length of random walks from random users and from the Sybils that are captured by SybilRank in each community.

We hypothesize that due to the majority of users in Tuenti being non-Sybil, if the Sybils are weakly connected to the majority, given a total variation distance, the random walks from Sybils need to be longer than that from average users. In such a case, the Sybils' random walk length can approximate the mixing time of the entire community, and we designate the length of random walks from average users as the mixing time of the non-Sybil part. Since the average users may include hidden Sybils, using their walk length only overestimates the mixing time of the non-Sybil part.

We select 1000 random users and 100 confirmed Sybils in each community. The total variation distance is set equal to 0.01. As shown in Figure 11, the mean length of random walks from the random users in each community is small (mostly < 100), while the mean Sybil walk length is much longer. This indicates that the Sybils connect to the majority users with a limited number of edges, which makes their needed walk length longer. This fact demonstrates that social-graph-based defenses can be effective for our real OSN graph. In addition, recall that in SybilRank we have placed multiple random seeds in each community, which facilitates the convergence of the trust propagation. The power iterations that SybilRank needs are even less than the random walk length in Figure 11.

7.2 Detecting Fakes in Tuenti

Manually inspecting the ranked list. We run SybilRank on the complete Tuenti social graph. We inspected 2K users at the bottom of the resulting ranked list, and all of them were fake (Sybils). We further examined the ranked list by inspecting the first lowest-ranked one million users. We randomly selected 100 users out of each 50K-user interval for inspection. As shown in Figure 12, the 100% fake portion was maintained at the first 50K-user interval, but the fake portion gradually decreased as we went up the list. Up to the first 200K lowest-ranked

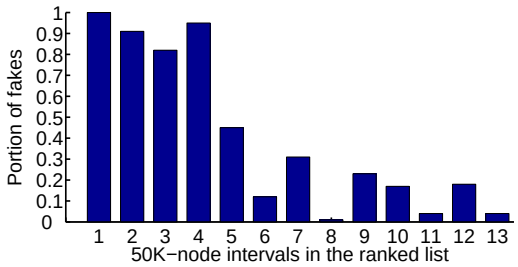


Figure 12: Sybil distribution over the lowest 650K users on the ranked list (intervals are numbered from the bottom).

users, around 90% are fake, as opposed to the ∼5% hit rate of Tuenti's current abuse-report-based method. This suggests an 18-fold increase in the efficiency with which Tuenti can process suspected accounts.

We also observe that above the 200K lowest-ranked users, the portion of fakes decreases abruptly from ∼90% to ∼50%, and then ∼10%. This is because fake accounts that have established many social links to real users can be ranked high. This reveals that SybilRank's limitation lies in the open nature of OSNs, i.e., the ease at which some fake accounts can befriend real users.

Although we have sampled users for inspection until the lowest 1 million users, due to confidentiality reasons we report the exact portions of fake users until the lowest 650K users. Above the lowest 650K, we obtain even lower portions of fakes, which subsequently stabilize. One can sample above the point in which the portion of fakes stabilizes to infer the portion of fakes in the complete Tuenti network. We have also obtained a more accurate estimate of the portion of fakes in the network by uniformly sampling over the complete network. This allows us to determine how many fake accounts are not captured by SybilRank. Again, we cannot reveal this statistic, as we are bound by confidentiality.

The portion of fakes we report is directly related to *precision* [32], which is a performance metric used in Collaborative Filtering. It is the ratio of relevant items over the top θ highest ranked items in terms of relevance. The results in Figure 12 reflect a precision as high as 90% among the first 200K lowest-ranked users (see [20]).

Formation of the Sybil collective. We showed above that SybilRank achieves an almost 100% portion of fakes in the first $50K$ lowest-ranked users. We now study the social connections among those $50K$ users. We found that fakes in Tuenti are rarely isolated from each other and that real world Sybils exhibit various formations (see [20]). We observed three large connected components that manifest a simple tree-like connection pattern between nodes of similar degree. This indicates that those accounts may be automatically crafted for spam, rating manipulation, and other attacks on a large scale.

In addition, those $50K$ users do not form a single connected component. Instead they form many separate connected clusters. Presumably this is due to the fact that the attackers behind those fake accounts are not centrally coordinated. We also investigate the degree of those $50K$ users. 80% of these users have no more than 10 friends, while there are hundreds among these Sybils that have more than 100 friends. This indicates that the node degree is not a reliable metric to detect fake accounts due to its large variance among fake users.

Sybils in large communities. SybilRank leverages the community structure to properly seed trust. It is much more accurate than just determining if an entire community is fake. Among the $50K$ lowest-ranked users, we found that part of them are embedded in the 25 large communities (see [20]). For example, each top-10 largest community has hundreds of Sybils. This indicates that SybilRank detects fake accounts even in large communities that mostly consist of non-Sybil users.

7.3 Discussion

With SybilRank, Tuenti needs ∼570 man hours to go over the 200K lowest ranked users and discover ∼180K fake accounts. With its current abuse-report-based method, which has only ∼5% hit rate, and assuming all these fakes are reported, Tuenti would need ∼10,300 hours. By executing SybilRank periodically, e.g., every month, we expect Tuenti to remain able to efficiently identify a substantial number of fake accounts.

Our study pin-pointed 200K suspicious accounts after a total of 20 hours of SybilRank and community estimation processing. Those accounts can be verified by one full-time (8h) employee in ∼70 days. This compares favorably to the feature-based detection mechanism used by Yang et al. [55], which unveiled in real time 100K fakes in the 120M-user RenRen, over 6 months.

Yang et al. [55] reported that 70% of their detected fake nodes do not have any connections to other fakes. However, unlike RenRen, Tuenti is invitation-only. Thus, a fake account is at least connected to its inviter when created. As a result, the number of isolated accounts is small (<70K) compared to the number of the fake accounts detectable by SybilRank. In addition, we found that Sybils in Tuenti do form dense connections among themselves, which as we show in §6.3 further enables SybilRank to uncover Sybils. Last, we note that most detected fake accounts were spammers [12].

8 Conclusion

Large scale social online services place immense attention to the experience of their user base, and the marketability of their user profiles and the social graph. In this context, they face a significant challenge by the existence and continuous creation of fake user accounts, which dilutes the advertising value of their network and annoys legitimate users. To this end, we have proposed

SybilRank, an effective and efficient fake account inference scheme, which allows OSNs to rank accounts according to their perceived likelihood of being fake. Therefore, this work represents a significant step towards practical Sybil defense: it enables an OSN to focus its expensive manual inspection efforts, as well as to correctly target existing countermeasures, such as CAPTCHAs.

9 Acknowledgments

We are grateful to the anonymous reviewers, our shepherd John Byers, Ang Li, and Vijay Erramilli for their valuable feedback. We are particularly thankful to Konstantina Papagiannaki for her extensive feedback on our paper's presentation. We also thank Nguyen Tran, Jinyang Li, Alan Mislove, and George Danezis for making their source code available for our evaluation. This work was partly funded by NSF Awards CNS-0845858 and CNS-1017858.

References

[1] Apache Hadoop. `http://hadoop.apache.org/`.
[2] Facebook connect. `developers.facebook.com/connect.php`.
[3] Stanford Large Network Dataset Collection. `http://snap.stanford.edu/data/index.html`.
[4] Tuenti: A Private, Invitation-only Social Platform Used by Millions of People to Communicate and Share Every Day. `http://www.tuenti.com`.
[5] An Explanation of CAPTCHAs. `http://www.facebook.com/note.php?note_id=36280205765`, 2008.
[6] Fake Accounts in Facebook - How to Counter it. `http://tinyurl.com/5w6un9u`, 2010.
[7] Stolen Facebook Accounts for Sale. `http://tinyurl.com/25cngas`, 2010.
[8] Tuenti, Spain's Leading Social Network, Switches on Local for a Location-based Future. `http://tinyurl.com/yeygmw5`, 2010.
[9] Why the Number of People Creating Fake Accounts and Using Second Identity on Facebook are Increasing. `http://tinyurl.com/3uwq75x`, 2010.
[10] Google+ Account Suspensions Over ToS Drawing Fire. `http://tinyurl.com/5vrt524`, 2011.
[11] Google Explores +1 Button To Influence Search Results. `http://tinyurl.com/7g927oy`, 2011.
[12] Personal communication with the Manager of User Support and the Product Manager of the Core and Community Management teams in Tuenti, 2011.
[13] The Facebook Blog: A Continous Commitment to Security. `https://www.facebook.com/blog/blog.php?post=486790652130`, 2011.
[14] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Eurocrypt*, 2003.
[15] A.-L. Bárabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286:509–512, 1999.
[16] E. Behrends. Introduction to Markov chains: with Special Emphasis on Rapid Mixing. In *Advanced Lectures in Mathematics*, 2000.
[17] 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*, 2009.
[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast Unfolding of Communities in Large Networks. In *Journal of Statistical Mechanics: Theory and Experiment*, 2008.
[19] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The Socialbot Network: When Bots Socialize for Fame and Money. In *ACSAC*, 2011.
[20] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the Detection of Fake Accounts in Large Scale Social Online Services. Technical Report, `http://www.cs.duke.edu/~qiangcao/publications/sybilrank_tr.pdf`, 2011.
[21] A. Cheng and E. Friedman. Sybilproof Reputation Mechanisms. In *P2PEcon*, 2005.
[22] P.-A. Chirita, J. Diederich, and W. Nejdl. MailRank: Using Ranking for Spam Detection. In *CIKM*, 2005.
[23] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes Using Social Networks. In *NDSS*, 2009.
[24] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, 2004.
[25] M. Dell amico and Y. Roudier. A Measurement of Mixing Time in Social Networks. In *5th International Workshop on Security and Trust Management*, 2009.
[26] J. R. Douceur. The Sybil Attack. In *IPTPS*, 2002.
[27] P. W. L. Fong. Preventing Sybil Attacks by Privilege Attenuation: A Design Principle for Social Network Systems. In *IEEE S&P*, 2011.
[28] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao. Detecting and Characterizing Social Spam Campaigns. In *IMC*, 2010.
[29] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A Walk in Facebook: Uniform Sampling of Users in Online Social Networks. In *IEEE INFOCOM*, 2010.
[30] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web Spam with TrustRank. In *VLDB*, 2004.
[31] J. A. Hanley and B. J. McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143, 1982.
[32] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22, 2004.
[33] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW*, 2003.
[34] A. N. Langville and C. D. Meyer. Deeper Inside Pagerank. *Internet Mathematics*, 1:2004, 2004.
[35] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *ACM SIGKDD*, 2006.
[36] C. Lesniewski-Laas and M. F. Kaashoek. Whanau: A Sybil-proof Distributed Hash Table. In *NSDI*, 2010.
[37] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: Leveraging Social Networks to Thwart Unwanted Traffic. In *NSDI*, 2008.
[38] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are Who You Know: Inferring User Profiles in Online Social Networks. In *ACM WSDM*, 2010.
[39] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
[40] A. Mohaisen, N. Hopper, and Y. Kim. Keep Your Friends Close: Incorporating Trust into Social-Network-based Sybil Defenses. In *INFOCOM*, 2011.
[41] A. Mohaisen, A. Yun, and Y. Kim. Measuring the Mixing Time of Social Graphs. In *ACM IMC*, 2010.
[42] M. Mondal, B. Viswanath, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Limiting Large-scale Crawls of Social Networking Sites. In *SIGCOMM Poster Session*, 2011.
[43] M. Motoyama, D. McCoy, K. Levchenko, G. M. Voelker, and S. Savage. Dirty Jobs: The Role of Freelance Labor in Web Service Abuse. In *Proceedings of the USENIX Security Symposium*, 2011.
[44] A. Nazir, S. Raza, C.-N. Chuah, and B. Schipper. Ghostbusting Facebook: Detecting and Characterizing Phantom Profiles in Online Social Gaming Applications. In *WOSN*, 2010.
[45] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford, 1999.
[46] A. Post, V. Shah, and A. Mislove. Bazaar: Strengthening User Reputations in Online Marketplaces. In *USENIX NSDI*, 2011.
[47] D. Quercia and S. Hailes. Sybil Attacks Against Mobile Users: Friends and Foes to the Rescue. In *INFOCOM*, 2010.
[48] P. Resnick and R. Sami. Sybilproof Transitive Trust Protocols. In *ACM Electronic Commerce Conference*, 2009.
[49] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE S&P*, 2010.
[50] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, SNS, 2011.
[51] K. Thomas, C. Grier, V. Paxson, and D. Song. Suspended Accounts in Retrospect: An Analysis of Twitter Spam. In *IMC*, 2011.
[52] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-Resilient Online Content Rating. In *NSDI*, 2009.
[53] N. Tran, J. Li, L. Subramanian, and S. S. Chow. Optimal Sybil-resilient Node Admission Control. In *INFOCOM*, 2011.
[54] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An Analysis of Social Network-based Sybil Defenses. In *ACM SIGCOMM*, 2010.
[55] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering Social Network Sybils in the Wild. In *IMC*, 2011.
[56] H. Yu. Sybil Defenses via Social Networks: A Tutorial and Survey. In *ACM SIGACT News, Distributed Computing Column*, 2011.
[57] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense Against Sybil Attacks. In *IEEE S&P*, 2008.
[58] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *SIGCOMM*, 2006.
[59] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal Sybil-Resistance for Recommendation Systems. In *IEEE S&P*, 2009.
[60] Y. Zhang, J. Wang, Y. Wang, and L. Zhou. Parallel Community Detection on Large Networks with Propinquity Dynamics. KDD, 2009.
[61] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. BotGraph: Large Scale Spamming Botnet Detection. In *NSDI*, 2009.