SybilSCAR: Sybil Detection in Online Social Networks via Local Rule based Propagation

Binghui Wang, Le Zhang, Neil Zhenqiang Gong
ECE Department, Iowa State University
{binghuiw, lezhang, neilgong}@iastate.edu

***Abstract*—Detecting Sybils in online social networks (OSNs) is a fundamental security research problem as adversaries can leverage Sybils to perform various malicious activities. Structure-based methods have been shown to be promising at detecting Sybils. Existing structure-based methods can be classified into two categories: Random Walk (RW)-based methods and Loop Belief Propagation (LBP)-based methods. RW-based methods cannot leverage labeled Sybils and labeled benign users simultaneously, which limits their detection accuracy, and they are not robust to noisy labels. LBP-based methods are not scalable, and they cannot guarantee convergence.**

In this work, we propose SybilSCAR, a new structure-based method to perform Sybil detection in OSNs. SybilSCAR maintains the advantages of existing methods while overcoming their limitations. Specifically, SybilSCAR is Scalable, Convergent, Accurate, and Robust to label noises. We first propose a framework to unify RW-based and LBP-based methods. Under our framework, these methods can be viewed as iteratively applying a (different) *local rule* to every user, which propagates label information among a social graph. Second, we design a new local rule, which SybilSCAR iteratively applies to every user to detect Sybils. We compare SybilSCAR with a state-of-the-art RW-based method and a state-of-the-art LBP-based method, using both synthetic Sybils and large-scale social network datasets with real Sybils. Our results demonstrate that SybilSCAR is more accurate and more robust to label noise than the compared state-of-the-art RW-based method, and that SybilSCAR is orders of magnitude more scalable than the state-of-the-art LBP-based method and is guaranteed to converge. To facilitate research on Sybil detection, we have made our implementation of SybilSCAR publicly available on our webpages.

I. Introduction

Online social networks (OSNs) are becoming more and more important and essential platforms for people to interact with each other, process information, and diffuse social influence, etc. For example, Facebook was reported to have 1.65 billion monthly active users as of April 2016 [1], and it has become the third most visited website worldwide, just next to Google.com and YouTube.com, according to Alexa [2]. However, it is well known that OSNs are vulnerable to *Sybil attacks*, in which attackers maintain a large number of Sybils, e.g., spammers, fake users, and compromised normal users. For instance, 10% of Twitter users were fake [3]. Adversaries can leverage such Sybils to perform various malicious activities such as disrupting democratic election [4], influencing financial market [5], distributing spams and phishing attacks [6], as well as harvesting private user data [7]. Therefore, Sybil detection in OSNs is an urgent research problem.

Indeed, this research problem has attracted increasing attention from multiple research communities including networking, security, and data mining. Among various methods, structure based methods have demonstrated promising results, e.g., SybilRank [8] was deployed to detect a large amount of Sybils in Tuenti, the largest OSN in Spain. Most structure-based methods [9], [10], [11], [12], [8], [13], [14], [15] can be grouped into two categories: Random Walk (RW)-based methods and Loop Belief Propagation (LBP)-based methods. Given a training dataset, these methods iteratively propagate label information among the social graph to predict labels for users. RW-based methods implement the propagation using random walks, while LBP-based methods implement the propagation using Loopy Belief Propagation [16]. RW-based methods [9], [10], [11], [12], [8], [13], [14] suffer from two major limitations: 1) they cannot leverage the labeled Sybils and labeled benign users in the training dataset simultaneously, and 2) they are not robust to label noise in the training dataset. The label of a user is noisy if the label is incorrect. Label noise often exists in practice due to human mistakes when manually labeling users [6], [17]. LBP-based methods also suffer from two limitations: 1) they are not scalable, and 2) they cannot guarantee convergence on real-world OSNs. The second limitation makes LBP-based methods sensitive to the number of iterations that the methods run.

Our work: We propose a new structure-based method, called SybilSCAR, to perform Sybil detection in OSNs. SybilSCAR combines the advantages of RW-based methods and LBP-based methods, while overcoming their limitations. First, we propose a general framework to unify state-of-the-art RW-based and LBP-based methods. Under our framework, each structure-based method can be viewed as iteratively applying a *local rule* to every user, which propagates label information from the training dataset to other users in the OSN. A local rule updates a user's label information via combining the user's neighbors' label information and the prior knowledge that we know about the user. Although RW-based methods and LBP-based methods use very different mathematical foundations (i.e., random walks vs. loopy belief propagation), they can be viewed as applying different local rules under our framework. Our framework makes it possible to analyze and compare different methods in a unified way. Moreover, our framework provides new insights on how to design better structure-based methods. Specifically, designing better structure-based methods reduces to designing better local rules.

Second, we design a novel local rule that integrates the advantages of both RW-based methods and LBP-based methods, while overcoming their limitations. SybilSCAR iteratively applies our local rule to every user. Our local rule, like RW-based methods and LBP-based methods, leverages the *homophily property* of OSNs. In an OSN that satisfies the homophily property, two linked users share the same label with a high probability. In our local rule, we associate a weight with each edge, which represents the probability that the two corresponding users have the same label. For a neighbor v of u, our local rule models v's influence (we call it *neighbor influence*) to u's label as the probability that u is a Sybil, given v's information alone. Our local rule combines neighbor influences and prior knowledge about a user in a multiplicative way to update knowledge about the user's label. Moreover, we linearize the multiplicative local rule in order to make SybilSCAR convergent.

Third, we theoretically analyze the convergence conditions of SybilSCAR. We also empirically compare SybilSCAR with SybilRank [8], a state-of-the-art RW-based method, and SybilBelief [15], a state-of-the-art LBP-based method, using three datasets: 1) a Facebook social graph with synthesized Sybils, 2) a small Twitter dataset (8,167 users and 68,282 edges) with real Sybils, and 3) a large Twitter dataset (21M users and 265M edges) with real Sybils. Our results demonstrate that 1) SybilSCAR achieves better detection accuracies than SybilRank and SybilBelief, 2) SybilSCAR is robust to larger label noises than SybilRank, and is as robust as SybilBelief; 3) SybilSCAR is as space and time efficient as SybilRank, but is several times more space efficient and orders of magnitude more time efficient than SybilBelief; 4) SybilSCAR and SybilRank are convergent, but SybilBelief is not.

In summary, our key contributions are as follows:

- We propose SybilSCAR, a structure-based method, to detect Sybils in OSNs. SybilSCAR is convergent, scalable, robust to label noise, and more accurate than existing methods.
- We propose a local rule based framework to unify state-of-the-art RW-based methods and LBP-based methods. Under our framework, we design a novel local rule that is the key component of SybilSCAR.
- We evaluate SybilSCAR both theoretically and empirically, and we compare it with a state-of-the-art RW-based method and a state-of-the-art LBP-based method. Our empirical results on multiple social network datasets demonstrate that SybilSCAR significantly outperforms the state-of-the-art RW-based method in terms of accuracy and robustness to label noise, and that SybilSCAR outperforms the state-of-the-art LBP-based method in terms of scalability and convergence.

II. Related Work

A. Structure-based Methods

We classify structure-based methods into Random Walk (RW)-based methods and Loopy Belief Propagation (LBP)-based methods.

RW-based Methods: Representative RW-based methods include [9], [10], [11], [12], [13], [8]. Two major limitations of RW-based methods are: 1) they cannot leverage labeled Sybils and labeled benign users simultaneously, which limits their detection accuracy, and 2) they are not robust to label noise. For instance, SybilGuard and SybilLimit assume that random walks starting from a benign user can fast reach other benign users, while starting from a Sybil is hard to enter the benign region. SybilInfer aims to leverage Bayesian inference and Monte-Carlo sampling to directly detect the bottleneck cut between benign users and Sybil users. However, all of them achieve limited performance because *they can only use one labeled benign user*. For the same reason, they are also not robust to noise in the label. Other limitations of these methods include being unscalable and assuming that the benign region of the social graph is fast mixing, which was shown to be invalid in real-world social graphs [18]. SybilRank was shown to outperform a variety of Sybil detection methods [8], and we treat it as a state-of-the-art RW-based method. SybilRank uses short random walks to propagate reputation among the social graph from a set of labeled benign users. SybilRank can only leverage labeled benign users, and it is not robust to label noise (see our experimental results in Figure 2).

LBP-based Methods: SybilBelief [15] is an existing LBP-based method. SybilBelief models a social network as a pairwise Markov Random Field (pMRF). Given some labeled Sybils and labeled benign users, SybilBelief first assigns prior probabilities to them and then uses LBP [16] to iteratively estimate the posterior probability of being a Sybil for each remaining user. The posterior probability of being a Sybil is used to predict a user's label. SybilBelief can leverage both labeled Sybils and labeled benign users simultaneously, and it is robust to label noise [15]. However, SybilBelief suffers from two limitations: 1) SybilBelief is not scalable because LBP requires maintaining messages on each edge, and 2) SybilBelief is not guaranteed to converge because LBP might oscillate on graphs with loops [16]. The second limitation means that SybilBelief's performance heavily relies on the number of iterations that LBP runs, but the best number of iterations might be different for different social networks.

B. Other methods

Some methods detect Sybils via binary machine learning classifiers. In particular, most methods in this direction represent each user using a set of features, which can be extracted from users' local subgraph structure (e.g., ego-network) [19], [20], [21] and side information (e.g., IP address, behaviors, and content) [22], [23], [24], [25], [26], [27]. Then, given a training dataset consisting of labeled benign users and labeled Sybils, they learn a binary classifier, e.g., Support Vector Machine [28]. Finally, the classifier is used to predict labels for the remaining users. A fundamental limitation of these methods is that attackers can mimic benign users by manipulating their profiles, so as to bypass the detection. However, these methods can still be used to filter the basic Sybils. These feature-based methods can be further combined with structure-based methods. For instance, Íntegro [14] first uses a feature-based method to predict a probability that each user is a *victim* (a victim is a user connecting to at least one Sybil), and then adapts SybilRank based on these probabilities. VoteTrust [29] leverages interactions between users. These methods are orthogonal to structure-based methods.

III. Problem Definition

We formally define our structure-based Sybil detection problem, introduce our design goals, and describe the threat model we consider in the paper.

A. Structure-based Sybil Detection

Suppose we are given an undirected social network $G = (V, E)$, where a node $v \in V$ represents a user and an edge $(u, v) \in E$ indicates a mutual relationship between u and v. For instance, on Facebook, an edge (u, v) could mean that u is in v's friend list and vice versa. On Twitter, an edge (u, v) could mean that u and v follow each other. Our structure-based Sybil detection is defined as follows:

Definition 1 (Structure-based Sybil Detection)**.** *Suppose we are given a social network and a training dataset consisting of some labeled Sybils and labeled benign nodes. Structure-based Sybil detection is to predict the label of each remaining node by leveraging the global structure of the social network.*

B. Design Goals

We target a method that satisfies the following goals:

1) Leveraging both labeled benign users and Sybils: Social network service providers often have a set of labeled benign users and labeled Sybils. For instance, verified users on Twitter or Facebook can be treated as labeled benign users; users spreading spam or malware can be treated as labeled Sybils, which can be obtained through manual inspection [8] or crowdsourcing [30]. Our method should be able to leverage both labeled benign users and labeled Sybils to enhance detection accuracy.

2) Robust to label noise: A given label of a user is noisy if it does not match the user's true label. Labeled users may have noisy labels. For instance, an adversary could compromise a labeled benign user or make a Sybil whitelisted as a benign user. In addition, labels obtained through manual inspection, especially crowdsourcing, often contain noises due to human mistakes [30]. We target a method that is robust when a minority fraction of given labels are incorrect.

3) Scalable: Real-world OSNs often have hundreds of millions of users and edges. Therefore, our method should be scalable and easily parallelizable.

4) Convergent: Existing methods and our method are iterative methods. Convergence makes it easy to determine when to stop an iterative method. It is hard to set the best number of iterations for an iterative method that is not convergent.

C. Threat Model

We call the subgraph containing all benign nodes and edges between them the *benign region*, and call the subgraph containing all Sybil nodes and edges between them the *Sybil region*. Edges between the two regions are called *attack edges*.

Homophily: One basic assumption under structure-based Sybil detection methods is that the benign region and the Sybil region are sparsely connected (i.e., the number of attack edges is relatively small), compared with the edges among the two regions. In other words, most benign users would not establish trust relationships with Sybils. We note that this assumption is equivalent to requiring that the OSNs follow *homophily*, i.e., two linked nodes share the same label with a high probability. For an extreme example, if the benign region and the Sybil region are separated from each other, then the OSN has a perfect homophily, i.e., every two linked nodes have the same label. Note that, it is of great importance to obtain OSNs that satisfy this assumption, otherwise the detection accuracies of structure-based methods are limited. For instance, Yang et al. [31] showed that RenRen *friendship* social network does not satisfy this assumption, and thus the performance of structure-based methods are unsatisfactory. However, Cao et al. [8] found that Tuenti, the largest OSN in Spain, satisfies the homophily assumption, and SybilRank can detect a large amount of Sybils in Tuenti.

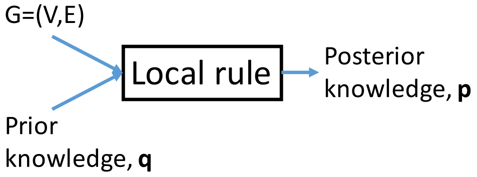


Fig. 1: Our proposed framework to unify state-of-the-art RW-based and LBP-based Sybil detection methods.

Generally speaking, there are two ways for service providers to construct a social network that satisfies homophily. One way is to approximately obtain trust relationships between users by looking into user interactions [32], predicting tie strength [33], asking users to rate their social contacts [34], etc. The other way is to preprocess the network structure so that structure-based methods are suitable to be applied. Specifically, analysts could detect and remove compromised benign nodes (e.g., front peers) [35], or employ feature-based classifier to filter Sybils, so as to decrease the number of attack edges and enhance the homophily. For instance, Alvisi et al. [36] showed that if the attack edges are established randomly, simple feature-based classifiers are sufficient to enforce Sybils to be suitable for structure-based Sybil detection. We note that the reason why the RenRen friendship social network did not satisfy homophily in the study of Yang et al. is that RenRen even didn't deploy simple feature-based classifiers at that time [31].

IV. Design of SybilSCAR

First, we propose a local rule based framework to unify state-of-the-art RW-based and LBP-based Sybil detection methods. Second, we design a new local rule. Third, we present our SybilSCAR algorithm, which is based on our local rule.

A. Our Local Rule based Framework

We first present a local rule based framework to unify state-of-the-art RW-based methods [12], [8], [13], [14] and LBP-based methods [15]. Specifically, we observe that these methods can be viewed as iteratively applying a *local rule* to every node in a *weighted* social network. Different methods use different local rules, resulting in different performances.

These methods first assign the *prior knowledge* of all nodes using a training dataset, and then propagate the prior knowledge among the social network to obtain the *posterior knowledge* via iteratively applying their local rules to every node. Specifically, *a local rule updates the posterior knowledge of a node by combining the influences from its neighbors with its prior knowledge*. We call the influence from a neighbor *neighbor influence*. Figure 1 shows our unified framework.

Notations: We denote by w_{uv} the weight of the edge (u,v), Γ_u the set of neighbors of node u, and d_u the total weights of edges linked to u, i.e., $d_u = \sum_{v\in\Gamma_u} w_{uv}$. In RW-based methods [12], [13], [14], edge weights model the relative importance (e.g., level of trust) of edges. In LBP-based method [15], an edge weight w_{uv} models the tendency that u and v share the same label. We denote by q_u and p_u the prior knowledge and posterior knowledge of the node u, respectively. In RW-based methods, q_u and p_u are the prior and posterior reputation scores of u, respectively, and they represent relative benignness of nodes. In LBP-based method, q_u and p_u are the prior and posterior probabilities that node u is a Sybil, respectively.

Additive local rule for RW-based methods: State-of-the-art RW-based methods [12], [8], [13], [14] first assign prior reputation scores for every node using a training dataset. Then they iteratively apply the following local rule to every node:

Random walk based additive local rule:

$$p_u = (1-\alpha)\sum_{v\in\Gamma_u} \underbrace{p_v \frac{w_{uv}}{d_v}}_{\text{neighbor influence}} + \alpha \underbrace{q_u}_{\text{prior knowledge}}, \quad (1)$$

where $\alpha \in [0,1]$ is called a restart probability of the random walk. We note that SybilRank uses a restart probability of 0 and normalizes the final reputation scores by node degrees.

We have two observations for the additive local rule. First, the neighbor influence from a neighbor v to u is a fraction of v's current reputation score p_v, and the fraction is proportional to the edge weight w_{uv}. Second, this local rule combines the prior knowledge and the neighbor influences *linearly* to update the posterior knowledge about a node.

Multiplicative local rule for LBP-based methods: SybilBelief [15], a LBP-based method, associates a binary random variable x_u with each node u, where $x_u = 1$ indicates that u is Sybil while $x_u = -1$ indicates that u is benign. Then, q_u and p_u are the prior and posterior probabilities that $x_u = 1$, respectively. SybilBelief first assigns the prior probabilities for nodes using a set of labeled benign nodes and/or a set of labeled Sybils, and then it iteratively applies the following local rule [15]:

LBP based multiplicative local rule:

$$m_{vu}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v,x_u) \prod_{z\in\Gamma_v/u} m_{zv}(x_v) \quad (2)$$

$$p_u = \frac{q_u \prod_{v\in\Gamma_u} m_{vu}(1)}{q_u \prod_{v\in\Gamma_u} m_{vu}(1) + (1-q_u)\prod_{v\in\Gamma_u} m_{vu}(-1)}, \quad (3)$$

where node potential $\phi_v(x_v)$ and edge potential $\varphi_{vu}(x_v,x_u)$ are defined as:

$$\phi_v(x_v) := \begin{cases} q_v & \text{if } x_v = 1 \\ 1-q_v & \text{if } x_v = -1 \end{cases}$$

$$\varphi_{vu}(x_v,x_u) := \begin{cases} w_{vu} & \text{if } x_u x_v = 1 \\ 1-w_{vu} & \text{if } x_u x_v = -1, \end{cases}$$

We also have two observations for the multiplicative local rule. First, *this local rule explicitly models neighbor influences*. Specifically, the neighbor influence from a neighbor v to u (i.e., $m_{vu}(x_u)$) is defined in Equation 2. To compute the neighbor influence $m_{vu}(x_u)$, u's neighbor v needs to multiply the neighbor influences from all its neighbors except u. Second, according to Equation 3, this local rule combines the neighbor influences with the prior probability *nonlinearly*.

Comparing RW-based additive local rule with LBP-based multiplicative local rule: LBP-based multiplicative local rule can tolerate a relatively larger fraction of label noise because of its nonlinearity [15], and it can leverage both labeled benign nodes and labeled Sybils. However, LBP-based multiplicative local rule is space and time inefficient because it requires a large amount of space and time to maintain the neighbor influences associated with every edge, and methods using this local rule are not guaranteed to converge. In contrast, RW-based additive local rule is space and time efficient, and methods using this local rule are guaranteed to converge. However, this local rule is sensitive to label noise, and it cannot leverage labeled benign nodes and labeled Sybils simultaneously.

B. Our New Local Rule

We aim to design a local rule that integrates the advantages of both RW-based and LBP-based local rules, while overcoming their limitations. Roughly speaking, our idea is to leverage the *multiplicativeness* like LBP-based local rule to be robust to label noise, while avoiding maintaining neighbor influences to be as space and time efficient as RW-based local rule. Next, we first discuss modeling of neighbor influence, and then present how we combine neighbor influences with prior knowledge.

Neighbor influence: We denote w_{vu}, which ranges from 0 to 1, as the *homophily strength* of the edge (u,v). $w_{uv} > 0.5$ means that u and v are in a *homogeneous relationship*, i.e., they tend to share the same label; $w_{uv} < 0.5$ means that u and v are in a *heterogeneous relationship*, i.e., they tend to have the opposite labels; and $w_{uv} = 0.5$ means that u and v are not correlated. We associate a binary random variable x_u with a node u, where $x_u = 1$ and $x_u = -1$ mean that u is a Sybil and benign node, respectively. We denote by f_{vu} the neighbor influence from v to u, where v is a neighbor of u. We model f_{vu} as the probability that u is a Sybil (i.e., $x_u = 1$), given the neighbor v's posterior probability p_v and the homophily strength w_{vu}. Formally, we have:

$$f_{vu} = Pr(x_u = 1 | p_v, w_{vu}). \quad (4)$$

Based on the homophily property, the neighbor influence f_{vu} should meet the following constraints:

$$p_v = 0.5 \text{ or } w_{vu} = 0.5 \Longrightarrow f_{vu} = 0.5 \quad (5)$$

$$p_v > 0.5 \text{ and } w_{vu} > 0.5 \Longrightarrow f_{vu} > 0.5 \quad (6)$$

$$p_v < 0.5 \text{ and } w_{vu} > 0.5 \Longrightarrow f_{vu} < 0.5 \quad (7)$$

$$p_v > 0.5 \text{ and } w_{vu} < 0.5 \Longrightarrow f_{vu} < 0.5 \quad (8)$$

$$p_v < 0.5 \text{ and } w_{vu} < 0.5 \Longrightarrow f_{vu} > 0.5, \quad (9)$$

where Equation 5 means that we cannot learn anything about u's label if v's label is undecidable (i.e., $p_v = 0.5$) or u and v are uncorrelated (i.e., $w_{vu} = 0.5$); Equations 6 and 7 mean that u and v tend to share the same label if they are in a homogeneous relationship; and Equations 8 and 9 mean that u and v tend to have opposite labels if they are in a heterogeneous relationship.

To satisfy all the above constraints in Equations 5-9, we model f_{vu} as follows:

$$f_{vu} = p_v w_{vu} + (1-p_v)(1-w_{vu}). \quad (10)$$

We can verify that f_{vu} in Equation 10 indeed satisfies the above constraints. Another way to interpret our model of f_{vu} is that w_{vu} is the probability that u and v have the same label and f_{vu} is the probability that u is a Sybil conditioned on w_{vu} and p_v. In our model, it is straightforward to compute a neighbor influence f_{vu}.

Combining neighbor influences with prior: In our local rule, a node's posterior probability of being Sybil is updated by combining its neighbor influences with its prior probability of being Sybil. In order to tolerate label noise, we leverage the multiplicative local rule in LBP-based methods. Specifically, we have:

$$p_u = \frac{q_u \prod_{v\in\Gamma_u} f_{vu}}{q_u \prod_{v\in\Gamma_u} f_{vu} + (1-q_u)\prod_{v\in\Gamma_u}(1-f_{vu})}. \tag{11}$$

However, methods that iteratively apply the above multiplicative local rule to every user are not guaranteed to converge. Therefore we further *linearize* Equation 11. For convenience, we set $w_{uv} = w$ for all edges in this paper. However, we believe that learning the edge weights w_{uv} for different edges would be a valuable future work. We first define concepts *residual variable and vector*.

Definition 2 (Residual Variable and Vector)**.** *We define the residual of a variable y as $\hat{y} = y - 0.5$; and define the residual vector $\hat{\mathbf{y}}$ of $\mathbf{y}$ as $\hat{\mathbf{y}} = [y_1 - 0.5, y_2 - 0.5, \cdots]$.*

With above definition, we denote $\hat{w}$ as the residual homophily strength. Moreover, by substituting variables in Equation 10 with their corresponding residuals, we have the residual neighbor influence $\hat{f}_{vu}$ as follows:

$$\hat{f}_{vu} = 2\hat{p}_v\hat{w}. \tag{12}$$

Based on the approximations $\ln(1+x) \approx x$ and $\ln(1-x) \approx -x$ when x is small, we have the following theorem, which linearizes Equation 11.

Theorem 1. *The residual posterior probability of being a Sybil for a node u can be linearized as:*

$$\hat{p}_u = \hat{q}_u + \sum_{v\in\Gamma(u)} \hat{f}_{vu}. \tag{13}$$

Proof: See Appendix. ■

By combining Equation 12 and Equation 13, we obtain our new local rule as follows:

Our local rule:

$$\hat{p}_u = \hat{q}_u + 2\sum_{v\in\Gamma(u)} \hat{p}_v\hat{w}. \tag{14}$$

C. SybilSCAR Algorithm

Our SybilSCAR iteratively applies our local rule to every node to compute the posterior probabilities. Suppose we are given a set of labeled Sybils which we denote as L_s and a set of labeled benign nodes which we denote as L_b. SybilSCAR first utilizes L_s and L_b to assign a prior probability of being a Sybil for all nodes. Specifically,

$$q_u = \begin{cases} \theta & \text{if } u \in L_s \\ 1-\theta & \text{if } u \in L_b \\ 0.5 & \text{otherwise,} \end{cases} \tag{15}$$

where $\theta > 0.5$ indicates that we assign a higher prior probability of being a Sybil to labeled Sybils. Considering that the labels might have noise, we will set θ to be smaller than 1. In practice, these prior probabilities can also be obtained from feature-based methods. Specifically, for each user we can leverage a binary classifier, trained using user's local features, to produce the probability of being a Sybil, which can then be treated as the user's prior probability. With such prior probabilities, SybilSCAR iteratively applies our local rule in Equation 14 to update residual posterior probabilities of all nodes.

Representing SybilSCAR as a matrix form: For convenience, we denote by a vector $\mathbf{q}$ the prior probability of being a Sybil for all nodes, i.e., $\mathbf{q} = [q_1; q_2; \cdots; q_{|V|}]$. Similarly, we denote by a vector $\mathbf{p}$ the posterior probability of all nodes, i.e., $\mathbf{p} = [p_1; p_2; \cdots; p_{|V|}]$. Moreover, we denote $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}$ as the residual prior probability vector and residual posterior probability vector of all nodes, respectively. We denote $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ as the adjacency matrix of the social graph, where the uth row represents the neighbors of u. Formally, if there exists an edge (u,v) between nodes u and v, then the entry $A_{uv} = A_{vu} = 1$, otherwise $A_{uv} = A_{vu} = 0$. With these notations, we can represent our SybilSCAR as iteratively apply the following equation:

$$\hat{\mathbf{p}}^{(t)} = \hat{\mathbf{q}} + 2\hat{w}\mathbf{A}\hat{\mathbf{p}}^{(t-1)}, \tag{16}$$

where $\hat{\mathbf{p}}^{(t)}$ is the residual posterior probability vector in the tth iteration. Initially, we set $\hat{\mathbf{p}}^{(0)} = \hat{\mathbf{q}}$.

Algorithm 1 SybilSCAR

```
Input: $G = (V,E)$, $L_s$, $L_b$, $\theta$, $w$, $\varepsilon$, and $T$.
Output: $p_u, \forall u \in V$.
  Initialize $\hat{\mathbf{p}}^{(0)} = \hat{\mathbf{q}}$.
  Initialize $t = 1$.
  while $\frac{\|\hat{\mathbf{p}}^{(t)} - \hat{\mathbf{p}}^{(t-1)}\|_1}{\|\hat{\mathbf{p}}^{(t)}\|_1} \geq \varepsilon$ and $t \leq T$ do
    Update residual posterior vector $\hat{\mathbf{p}}^{(t)}$ using Equation 16;
    $t = t + 1$.
  end while
  return $\hat{\mathbf{p}}^{(t)} + 0.5$.
```

Algorithm 1 summarizes the pseudocode of SybilSCAR. We stop running SybilSCAR when the relative errors of residual posterior probabilities between two consecutive iterations is smaller than some threshold or it reaches the predefined number of maximum iterations. After SybilSCAR halts, we predict u to be a Sybil if $p_u > 0.5$, otherwise we predict u to be benign.

V. Theoretical Analysis

A. Convergence Analysis

We analyze the condition when SybilSCAR converges.

Lemma 1 (Sufficient and Necessary Convergence Condition for a Linear System [37])**.** *Suppose we are given an iterative linear process: $\mathbf{y}^{(t)} \leftarrow \mathbf{c} + \mathbf{M}\mathbf{y}^{(t-1)}$. The linear process converges with any initial choice $\mathbf{y}^{(0)}$ if and only if the spectral radius[1] of $\mathbf{M}$ is smaller than 1, i.e., $\rho(\mathbf{M}) < 1$.*

[1] The spectral radius of a square matrix is the maximum of the absolute values of its eigenvalues.

Proof: See [37]. ■

Based on Equation 16 and Lemma 1, we are able to analyze the convergence condition of SybilSCAR.

Theorem 2 (Sufficient and Necessary Convergence Condition of SybilSCAR)**.** *The sufficient and necessary condition that makes SybilSCAR converge is equivalent to bounding the residual homophily strength $\hat{w}$ as:*

$$\hat{w} < \frac{1}{2\rho(\mathbf{A})}. \tag{17}$$

Proof: Let $\mathbf{M} = 2\hat{w}\cdot\mathbf{A}$, then $\rho(\mathbf{M}) = 2\hat{w}\cdot\rho(\mathbf{A})$. Using Lemma 1, $\rho(\mathbf{M}) < 1$ holds if and only if $\hat{w} < \frac{1}{2\rho(\mathbf{A})}$. ■

Theorem 2 provides a strong sufficient and necessary convergence condition. However, in practice setting $\hat{w}$ using Theorem 2 is computationally expensive, as it involves computing the largest eigenvalue with respect to spectral radius. Hence, we instead derive a *sufficient condition* for SybilSCAR's convergence, which enables us to set $\hat{w}$ with cheap computation. Specifically, our sufficient condition is based on the fact that any norm is an upper bound of the spectral radius [38], i.e., $\rho(\mathbf{M}) \leq \|\mathbf{M}\|$, where $\|\cdot\|$ indicates some matrix norm. In particular, we use the induced l_1 matrix norm $\|\cdot\|_1$ [2]. In this way, our sufficient condition for convergence is as follows:

Theorem 3 (Sufficient Convergence Condition of SybilSCAR)**.** *The sufficient condition that makes SybilSCAR converge is*

$$\hat{w} < \frac{1}{2\|\mathbf{A}\|_1} = \frac{1}{2\max_{u\in V} D_u}, \tag{18}$$

where $D_u = |\Gamma_u|$ is the degree of u.

Proof: As $\rho(\mathbf{M}) \leq \|\mathbf{M}\|_1$, we achieve the sufficient condition by enforcing $\|\mathbf{M}\|_1 < 1$, where $\mathbf{M} = 2\hat{w}\cdot\mathbf{A}$. Specifically, we have

$$2\hat{w}\cdot\rho(\mathbf{A}) < 1 \tag{19}$$

$$\Longleftarrow 2\hat{w}\|\mathbf{A}\|_1 < 1 \tag{20}$$

$$\Longleftarrow \hat{w} < \frac{1}{2\|\mathbf{A}\|_1} = \frac{1}{2\max_{u\in V} D_u}. \tag{21}$$

■

Theorem 3 provides a guideline for us to set $\hat{w}$, i.e., once $\hat{w}$ is smaller than the inverse of 2 times of the maximum node degree, SybilSCAR is guaranteed to converge. In practice, however, some nodes (e.g., celebrities) could have orders of magnitude bigger degrees than the others (e.g., ordinary people), and such nodes make $\hat{w}$ very small. In our experiments, we note that SybilSCAR can still converge when replacing the maximum node degree with the average node degree.

B. Complexity Analysis

SybilSCAR, state-of-the-art RW-based methods [12], [8], [13], and LBP-based method [15] have the same space complexity, i.e., $O(|E|)$, and their time complexity is $O(t|E|)$, where t is the number of iterations. Although SybilSCAR and SybilBelief (a LBP-based method) have the same asymptotic space and time complexity, SybilSCAR is several times more space efficient and orders of magnitude more time efficient than SybilBelief in practice, as we demonstrate in our experiments. This is because SybilBelief needs to store neighbor influences (i.e., $m_{vu}(x_u)$) in both directions of every edge and update them in every iteration.

TABLE I: Dataset statistics.

Dataset	Facebook	Small Twitter	Large Twitter
#Nodes	54,941	8,167	21,297,772
#Edges	237,324	68,282	265,025,545
Ave. degree	8.64	16.72	24.89
Ave. #attack edge	0.05	49.46	126.84

Parallel implementation: SybilSCAR, state-of-the-art RW-based methods [12], [8], [13], and LBP-based method can be easily implemented in parallel. Specifically, we can divide nodes into groups, and a thread or computer applies the corresponding local rule to a group of nodes iteratively.

VI. Evaluations

We compare SybilSCAR with SybilRank [8], a state-of-the-art RW-based method, and SybilBelief [15], a state-of-the-art LBP-based method, in terms of accuracy, robustness to label noise, scalability, and convergence.

A. Experimental Setups

Dataset description: We describe the datasets used in our experiments. Table I shows some statistics about these datasets, where the last row shows the average number of attack edges per Sybil in each dataset.

1) Facebook with synthesized Sybils: Following previous works [10], [11], [8], [15], we use a real-world social network as the benign region, while synthesizing the Sybil region using the *Preferential Attachment (PA)* model [39], which is a widely used method to generate networks; and we add attack edges between the benign region and Sybil region uniformly at random. We use a Facebook social network with 43,953 nodes and 182,384 undirected edges as the benign region. The Facebook dataset is an interaction graph from the New Orleans regional network [40]. In this graph, nodes are Facebook users and a link is added between two users if they comment on each other's wall posts at least once. We synthesized the Sybil region such that the fraction of Sybils in the social network is 20%; the number of attack edges is 500. Note that this Facebook dataset with synthesized Sybils satisfies the homophily property very well, so we expect all compared structure-based methods to achieve high accuracies.

2) Small Twitter with real Sybils: We obtained a publicly available Twitter dataset with 1,000 Sybils and 10,000 benign nodes from Yang et al. [13]. This published dataset is a subset of the authors' original one. These Sybils were labeled spammers. Since a Sybil can follow a large number of benign nodes without being followed back, we transform this network to be an undirected one via keeping an edge between two nodes only if they follow each other. Then we extracted the largest connected component of the undirected network since all compared methods work on connected networks. After preprocessing, we have 8,167 nodes in total, with 7,358 benign nodes and 809 Sybils; 68,282 edges and 40,010 attack edges.

[2] $\|\mathbf{M}\|_1 = \max_j \sum_i |\mathbf{M}_{ij}|$, the maximum absolute column sum of the matrix.

In other words, 9.9% of nodes are Sybils, the average degree is 16.72, and the average attack edge per Sybil is 49.46.

3) Large Twitter with real Sybils: We obtained a snapshot of a large-scale Twitter follower-followee network crawled by Kwak et al. [41]. We also transformed the follower-followee network into an undirected one via keeping an edge between two users only if they follow each other, and then we extracted the largest connected component of the undirected network. Finally, we obtained a Twitter network with 21,297,772 nodes and 265,025,545 edges, with an average degree of 24.89. To perform evaluation, we need groundtruth labels of the users. Since the Twitter network includes users' Twitter IDs, we can write a crawler to visit each user's profile using Twitter's API, which tells us the status (i.e., active, suspended, or deleted) of each user. In our groundtruth (a part of them were obtained from Hao Fu who collected them in 2014), 145,183 nodes were suspended, 2,566,944 nodes were deleted, and the remaining 18,585,645 users are active. In our experiments, we take suspended users as Sybils and the active users as benign nodes. The average number of attack edges per Sybil is 126.84. Note that our groundtruth labels might be noisy because some active users might be Sybils, but they evaded Twitter's detection, and Twitter might have deleted some Sybils.

Compared methods: We compare SybilSCAR with SybilRank [8], a state-of-the-art RW-based method, and SybilBelief [15], a state-of-the-art LBP-based method. In addition, we use random guessing as a baseline.

Training and testing sets: We randomly select 20, 100, 100,000 benign users and Sybils on Facebook, small Twitter, and large Twitter to construct training datasets, respectively. The remaining benign and Sybil nodes are used as testing data.

Parameter setting: For SybilSCAR, we set $\theta = 0.9$ to consider possible label noises, i.e., we assign a prior probability 0.9, 0.1, and 0.5 to labeled Sybils, labeled benign nodes, and unlabeled nodes, respectively; we set $\varepsilon = 10^{-3}$ and $T = 20$; considering different average degrees of Facebook, small Twitter, and large Twitter, we set $\hat{w} = 0.1, 0.05$, and 0.04, respectively. We set the parameters of SybilRank and SybilBelief according the papers that introduced them. For instance, for SybilBelief, the edge weight is set to be 0.9 for all edges; SybilRank requires early termination, and we set the number of iterations as $\lceil \log(|V|) \rceil$.

We implemented SybilRank and SybilSCAR in C++, and we obtained implementation of SybilBelief (also in C++) from its authors. We performed all our experiments on a Linux machine with 16GB memory and 8 cores.

B. Experimental Results

Overall ranking accuracy: Viswanath et al. [42] demonstrated that Sybil detection methods can be treated as ranking mechanisms, and they can be evaluated using Area Under the Receiver Operating Characteristic Curve (AUC). Therefore, we adopt AUC to evaluate ranking accuracy. Suppose we rank nodes with respect to their posterior reputation/probability of being a Sybil in a descending order. AUC is the probability that a randomly selected Sybil ranks higher than a randomly selected benign node.

TABLE II: AUCs of compared methods.

Methods	Facebook	Small Twitter	Large Twitter
SybilRank	0.99	0.86	0.69
SybilBelief	1.00	0.98	0.78
SybilSCAR	1.00	0.99	0.82

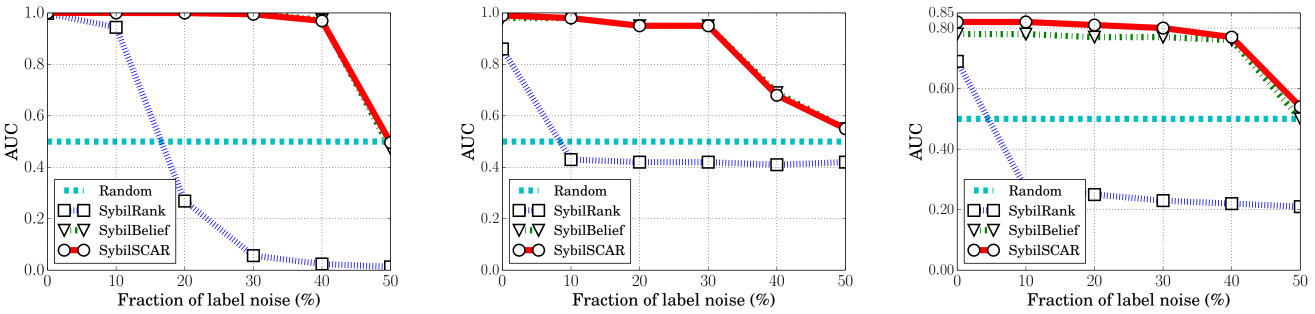
Table II shows the AUCs of all compared methods on the three datasets. We have several observations. First, when the social network satisfies the homophily property very well, all the compared methods achieve very accurate results. Specifically, on the Facebook dataset, all the three methods achieve AUCs that are close to 1. Second, on the two Twitter datasets which less satisfy the homophily property, SybilSCAR and SybilBelief are substantially more accurate than SybilRank. The reason includes that SybilSCAR and SybilBelief can leverage both labeled benign nodes and Sybils, and they are based on a multiplicative local rule. Third, SybilSCAR is slightly more accurate than SybilBelief on the two Twitter datasets. Recall that SybilSCAR uses a local rule that is approximated from the multiplicative local rule. This implies that our approximation not only makes SybilSCAR convergent, but also improves its accuracy. Fourth, SybilSCAR's accuracy decreases on the large Twitter dataset. Possible reasons include that the large Twitter dataset less satisfies the homophily property and that the groudtruth labels have noises.

Robustness to label noise: In practice, a training dataset might have noises, i.e., some labeled benign nodes are actually Sybils and some labeled Sybils are actually benign. Such noises could be introduced by human mistakes [17]. Thus, one natural question is how label noise impacts the accuracy.

For a given level of noise $\tau\%$, we randomly choose $\tau\%$ of labeled Sybils in the training dataset and mislabel them as benign users; and we also sample $\tau\%$ of labeled benign users in the training dataset and mislabel them as Sybils. We vary $\tau\%$ from 10% to 50% with a step size of 10%. Note that we didn't perform experiments for $\tau\% > 50\%$ as all these methods cannot detect Sybils when a majority of labels are incorrect. Figure 2 shows the AUCs of SybilRank, SybilBelief, and SybilSCAR on the three datasets against different levels of label noises. We observe that 1) SybilSCAR and SybilBelief almost have the same robustness against label noise, except that SybilSCAR is slightly more robust to label noise on the large Twitter dataset; 2) SybilSCAR and SybilBelief are much more robust to label noise than SybilRank. Specifically, SybilSCAR and SybilBelief can tolerate label noise up to 40% on Facebook and large Twitter, and 30% on small Twitter. However, the performance of SybilRank is worse than random guessing when the level of label noise is higher than 20%. The possible explanation is that both SybilBelief and SybilSCAR use multiplicative local rules, and they incorporate both labeled benign users and Sybils in the training dataset.

Scalability: We evaluate scalability in terms of the peak memory and time used by each method. Because evaluating scalability requires social networks with varying number of edges, we evaluate scalability on synthesized graphs with different number of edges.

Figure 3 exhibits the peak memory and time used by SybilRank, SybilBelief, and SybilSCAR for different number

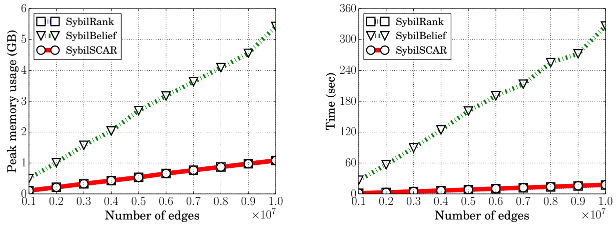


(a) Facebook

(b) Small Twitter

(c) Large Twitter

Fig. 2: AUCs of SybilRank, SybilBelief, and SybilSCAR vs. level of label noise. SybilSCAR and SybilBelief have very close robustness to label noise, while they are much more robust to label noise than SybilRank.



(a) Peak memory usage

(b) Time

Fig. 3: Space and time efficiency of SybilRank, SybilBelief, and SybilSCAR, vs. number of edges. SybilSCAR and SybilRank have almost the same space and time efficiency, while SybilSCAR is several times more space efficient and orders of magnitude more time efficient than SybilBelief.

of edges. Note that all these methods' time efficiencies depend on their number of iterations. To avoid bias introduced by the number of iterations, we run these methods for the same number of iterations, i.e., 20 in our experiments. We observe that: 1) all methods have linear space and time complexity, which is consistent with our theoretical analysis in Section V-B; 2) SybilRank and SybilSCAR use almost the same space and time; 3) SybilSCAR requires a few times less memory than SybilBelief and is orders of magnitude faster than SybilBelief. The reason is that SybilBelief needs a large amount of resources to store and maintain the messages on every edge in the graph.

Convergence: We define a relative error of residual posterior probability vectors of SybilSCAR as $\frac{\|\hat{\mathbf{p}}^{(t)}-\hat{\mathbf{p}}^{(t-1)}\|_1}{\|\hat{\mathbf{p}}^{(t)}\|_1}$, where $\hat{\mathbf{p}}^{(t)}$ is the residual vector of posterior probability produced by SybilSCAR in the tth iteration. Similarly, we can define relative errors for SybilRank and SybilBelief using their vectors of posterior reputation/probability. Figure 4 shows the relative errors vs. the number of iterations on small Twitter (Due to space limitation, we do not show the similar results on the other two datasets). We observe that 1) SybilSCAR and SybilRank converge after several iterations; 2) SybilSCAR converges faster than SybilRank; and 3) the relative errors of SybilBelief oscillate. SybilBelief does not converge because there exists many loops in real-world social networks and LBP may oscillate on graphs with loops, as pointed out by the author of LBP [16].

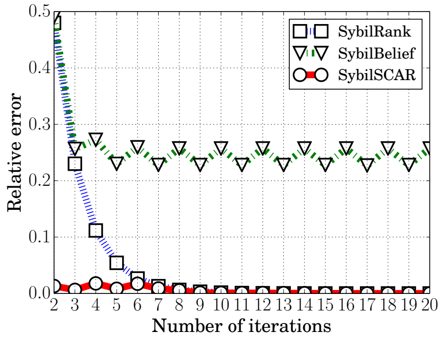


Fig. 4: Relative errors of SybilRank, SybilBelief, and SybilSCAR. vs. the number of iterations. SybilSCAR and SybilRank can converge, but SybilBelief cannot.

C. Summary

We summarize our key observations as follows:

- Compared to SybilRank, SybilSCAR is substantially more accurate and more robust to label noise.
- Compared to SybilBelief, SybilSCAR is orders of magnitude more scalable and guaranteed to converge.

VII. Conclusion and Future Work

In this work, we first propose a local rule based framework to unify state-of-the-art Random Walk (RW)-based Sybil detection methods and Loopy Belief Propagation (LBP)-based Sybil detection methods. Our framework makes it possible to analyze and compare different Sybil detection methods in a unified way. Second, we design a new local rule. Our local rule integrates advantages of RW-based methods and LBP-based methods, while overcoming their limitations. Our evaluation results on both synthesized Sybils and real-world Sybils demonstrate that SybilSCAR is more accurate and more robust to label noise than RW-based methods, and that SybilSCAR is orders of magnitude more scalable than SybilBelief and guaranteed to converge.

Future research directions include 1) automatically learning the homophily strength for each edge, 2) theoretically analyzing different local rules with respect to accuracy and robustness to label noise, and 3) applying SybilSCAR to detect other types of Sybils such as web spams, fake reviews, and fake likes.

REFERENCES

[1] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in *NDSS*, 2012.

[2] Facebook Popularity. (2015, October). [Online]. Available: http://www.alexa.com/topsites

[3] Sybils in Twitter, "http://www.nbcnews.com/technology/1-10-twitter-accounts-fake-say-researchers-2d11655362."

[4] Hacking Election. (2016, May). [Online]. Available: http://goo.gl/G8o9x0

[5] Hacking Financial Market. (2016, May). [Online]. Available: http://goo.gl/4AkWyt

[6] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *IEEE S & P*, 2011.

[7] 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.

[8] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *NSDI*, 2012.

[9] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman., "SybilGuard: Defending against Sybil attacks via social networks," in *SIGCOMM*, 2006.

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

[11] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil nodes using social networks," in *NDSS*, 2009.

[12] A. Mohaisen, N. Hopper, and Y. Kim, "Keep your friends close: Incorporating trust into social network-based sybil defenses," in *INFOCOM*, 2011.

[13] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammer's social networks for fun and profit," in *WWW*, 2012.

[14] Y. Boshmaf, D. Logothetis, G. Siganos, J. Leria, J. Lorenzo, M. Ripeanu, and K. Beznosov, "Integro: Leveraging victim prediction for robust fake account detection in osns," in *NDSS*, 2014.

[15] N. Z. Gong, M. Frank, and P. Mittal, "Sybilbelief: A semi-supervised learning approach for structure-based sybil detection," *IEEE TIFS*, vol. 9, no. 6, pp. 976–987, 2014.

[16] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, 1988.

[17] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, "Social turing tests: Crowdsourcing sybil detection," *NDSS*, 2013.

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

[19] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *PAKDD*, 2010.

[20] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network Sybils in the wild," in *IMC*, 2011.

[21] A. H. Wang, "Don't follow me - spam detection in twitter," in *SECRYPT 2010*, 2010.

[22] G. S. S. Yardi, D. Romero and D. Boyd, "Detecting spam in a Twitter network," *First Monday*, vol. 15(1), 2010.

[23] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *SIGIR*, 2010.

[24] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[25] J. Song, S. Lee, and J. Kim, "Spam filtering in Twitter using sender-receiver relationship," in *RAID*, 2011.

[26] G. Wang, T. Konolige, C. Wilson, and X. Wang, "You are how you click: Clickstream analysis for sybil detection," in *Usenix Security*, 2013.

[27] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *CCS*, 2014.

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, 1995.

[29] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, and Y. Dai, "Votetrust: Leveraging friend invitation graph to defend against social network sybils," in *INFOCOM*, 2013.

[30] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, "Social turing tests: Crowdsourcing Sybil detection," in *NDSS*, 2013.

[31] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM TKDD*, vol. 8, no. 1, p. 2, 2014.

[32] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Eurosys*, 2009.

[33] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in *CHI*, 2009.

[34] W. Wei, F. Xu, C. Tan, and Q. Li, "SybilDefender: Defend against Sybil attacks in large social networks," in *INFOCOM*, 2012.

[35] Y. Wang and A. Nakao, "Poisonedwater: An improved approach for accurate reputation ranking in p2p networks," *FGCS*, vol. 26, no. 8, pp. 1317–1326, 2010.

[36] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, "Sok: The evolution of sybil defense via social networks," in *IEEE S & P*, 2013.

[37] Y. Saad, *Iterative methods for sparse linear systems*. Siam, 2003.

[38] N. Derzko and A. Pfeffer, "Bounds for the spectral radius of a matrix," *Mathematics of Computation*, vol. 19, no. 89, 1965.

[39] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, 1999.

[40] B. Viswanath, A. Misolve, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in Facebook," in *WOSN*, 2009.

[41] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 591–600.

[42] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based Sybil defenses," in *SIGCOMM*, 2010.

APPENDIX A
PROOF OF THEOREM 1

We denote $\mathscr{Z}_u = q_u \prod_{v\in\Gamma(u)} f_{vu} + (1-q_u)\prod_{v\in\Gamma(u)}(1-f_{vu})$. Rewriting $p_u = \frac{1}{\mathscr{Z}_u} q_u \prod_{v\in\Gamma(u)} f_{vu}$ with the corresponding residual variables yields

$$\begin{aligned}
&0.5+\hat{p}_u = \frac{1}{\mathscr{Z}_u}\big(0.5+\hat{q}_u\big)\prod_{v\in\Gamma(u)}\big(0.5+\hat{f}_{vu}\big) \\
&\Longrightarrow \ln(1+2\hat{p}_u) = -\ln\mathscr{Z}_u + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\big(0.5+\hat{f}_{vu}\big) \\
&= -\ln\mathscr{Z}_u + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\big(0.5\big) + \sum_{v\in\Gamma(u)} \ln(1+2\hat{f}_{vu})
\end{aligned}$$

Using approximation $\ln(1+x)\approx x$ when x is small, we have:

$$2\hat{p}_u = -\ln\mathscr{Z}_u + 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) + \sum_{v\in\Gamma(u)} 2\hat{f}_{vu}. \tag{22}$$

Similarly, via rewriting $1-p_u = \frac{1}{\mathscr{Z}_u}(1-q_u)\prod_{v\in\Gamma(u)}(1-f_{vu})$ with the corresponding residual variables and using approximation $\ln(1-x)\approx -x$ when x is small, we have:

$$-2\hat{p}_u = -\ln\mathscr{Z}_u - 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) - \sum_{v\in\Gamma(u)} 2\hat{f}_{vu}. \tag{23}$$

Adding Equation 22 with Equation 23 yields $\ln\mathscr{Z}_u = |\Gamma(u)|\cdot\ln(0.5)$. Via substituting this relation into Equation 22 or Equation 23, we have:

$$\hat{p}_u = \hat{q}_u + \sum_{v\in\Gamma(u)} \hat{f}_{vu}. \tag{24}$$