2018 IEEE 26th International Conference on Network Protocols

# Sybil Detection in Social-Activity Networks: Modeling, Algorithms and Evaluations

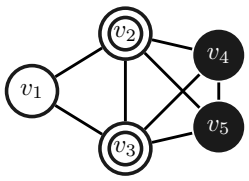
Xiaoying Zhang, Hong Xie, John C. S. Lui
*CSE Dept., The Chinese University of Hong Kong*
{xyzhang,hxie,cslui}@cse.cuhk.edu.hk

***Abstract*—Detecting fake accounts (sybils) in online social networks (OSNs) is vital to protect OSN operators and their users from various malicious activities. Typical graph-based sybil detection (a mainstream methodology) assumes that sybils can make friends with only a limited (or small) number of honest users. However, recent evidences showed that this assumption does not hold in real-world OSNs, leading to low detection accuracy. To address this challenge, we explore users' activities to assist sybil detection. The intuition is that honest users are much more selective in choosing who to interact with than to befriend with. We first develop the *social and activity network* (SAN), a two-layer hyper-graph that unifies users' friendships and their activities, to fully utilize users' activities. We also propose a more practical sybil attack model, where sybils can launch both friendship attacks and activity attacks. We then design Sybil_SAN to detect sybils via coupling three random walk-based algorithms on the SAN, and prove the convergence of Sybil_SAN. We develop an efficient iterative algorithm to compute the detection metric for Sybil_SAN, and derive the number of rounds needed to guarantee the convergence. We use *"matrix perturbation theory"* to bound the detection error when sybils launch many friendship attacks and activity attacks. Extensive experiments on both synthetic and real-world datasets show that Sybil_SAN is highly robust against sybil attacks, and can detect sybils accurately under practical scenarios, where current state-of-art sybil defenses have low accuracy.**

## I. Introduction

Online social network (OSNs), such as Twitter, Facebook, LinkedIn, Google+, are becoming increasingly popular. They serve as essential platforms for people to make new friends, share their experiences, and diffuse social influence, etc. However, due to the innate openness, i.e., allowing users to create new identities readily, OSNs are particularly vulnerable to sybil attacks, where an attacker can create multiple pseudonymous identities (we call *sybils* here), to subvert the system. For example, a sybil may distribute spam or phishing attacks [1], harvest personal user information [2], gain disproportionate influence/voting [3], [4], etc. Twitter reported that $10\%$ of Twitter users are fake [5]. Similarly, Facebook estimated that about 83 millions of its users are fake [6]. Thus, it is important to detect sybils in OSNs.

Among various methods, the graph-based sybil detection is the mainstream one, due to its computational efficiency and generality to detect sybils with different activity behavior. Typically, the graph-based sybil detection can be described as: (1) Model an OSN as a graph, in which nodes represent user accounts and edges represent users' friendships. (2) The objective is to exploit the graph structure to identify those sybil users given a small set of users with known labels (honest or sybil). The underlying assumption is that honest users seldom make friends with sybils, i.e., there are a limited number of friendship links between honest users and sybil users (a.k.a. the *limited-attack-edges assumption*), where an attack edge means a link between an honest user and a sybil. Under this assumption, a large number of algorithms have been proposed, e.g., [7]–[16], just to name a few. However, recent works [17]–[19] found that sybils are able to create a larger number of attack edges, i.e., the limited-attack-edges assumption does not hold in real-world OSNs. Breaking down this assumption does lead to a very low detection accuracy [20], [21]. This motivates us to explore practical sybil attack models and design effective sybil detection algorithms.




(a) Friendship graph

**Among honest users**
- user $v_1$ mentions user $v_2$
- user $v_3$ retweets a tweet of user $v_1$
- user $v_3$ mentions user $v_1$

**Among sybils users**
- user $v_5$ replies to user $v_4$

**From sybils to honest users**
- user $v_5$ retweets a tweet of user $v_3$

(b) Users' activities

**Fig. 1: A motivating example**

Our idea is to explore user activities to refine the attack model and design new detection algorithms. The intuition is that luring an honest user to conduct some daily activities, e.g., replying a tweet, is far more difficult than luring an honest user to establish friendship links. This intuition is supported by some experimental studies in Twitter [22]. Let us use a simple example to illustrate benefits of exploring activities.

***Example* 1 (Benefits of exploring activities).** *Consider the friendship graph in Figure 1a, where $v_1, v_2, v_3$ are honest nodes and $v_4, v_5$ are sybil nodes. Each undirected link, e.g., $(v_1, v_2)$, indicates a friendship relationship. Given that $v_2, v_3$ are honest nodes, it is difficult to detect the sybils out, i.e., $v_4, v_5$, because their connectivity to $v_2$ and $v_3$ are as good as the honest node $v_1$. Figure 1b shows the historical activities of the users in Figure 1a. Given $v_2, v_3$ are honest, we can easily see that $v_4, v_5$ are more suspicious to be sybils than $v_1$, because the honest nodes $v_2, v_3$ did not initiate any activity to $v_4$ and $v_5$, while $v_3$ actively interacted with $v_1$ twice.*

Example 1 uses a simple case to highlight that exploring users' activities can help us to detect sybils out, which are

978-1-5386-6043-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICNP.2018.00015



IEEE computer society

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

difficult to be detected from the friendship graph (i.e., the limited-attack-edges assumption does not hold). The challenge is that in practice, sybils may also lure some honest users to conduct some daily activities with them. Furthermore, many users in real-world OSNs are not very active in interacting with others, e.g., by analyzing a subnetwork of Twitter, we found that $113,214$ out of $409,694$ users (around $28\%$ of users) interact with others at most once only. This increases the risk of classifying such inactive users as sybils since they also seldom interact with other honest users, similar with sybils. This paper aims to explore such general settings and answer:

- *How to fully utilize users' friendships and their activities to detect sybils and do away with the limited-attack-edges assumption?*
- *How to practically model sybils' attacking behavior on both friendships and activities, and design effective sybil detection algorithms with theoretical guarantees?*

Our contributions are:

- We develop a two-layer hyper-graph model to fully utilize users' friendships and their activities in an OSN. We propose a new sybil attack model in which sybils can launch both friendship attacks and activity attacks. Our attack model relies on empirical findings on real-world user and sybil behavior. (Section III)
- We design the Sybil_SAN to detect sybils, which propagates the trust (distrust) from given honest (sybil) nodes to other user nodes via coupling three random walks on SAN, with convergence guarantee. Computing the converged trust (distrust) score is expensive, we also design an iterative algorithm to calculate it. (Section IV)
- We apply *"Markov chain mixing time"* to derive the number of rounds needed to guarantee that the iterative algorithm terminates. We also apply *"matrix perturbation theory"* to bound the error in the detection metric (i.e., normalized trust scores), when sybils launch more friendship attacks and activity attacks. (Section V)
- Experiments on both synthetic and real-world sybil datasets show that under practical scenarios with large attacks in friendships and activities, Sybil_SAN can still detect sybils accurately, while the compared algorithms have very low accuracy. Experimental results further verify that our Sybil_SAN is highly robust (in terms of the detection metric) against sybil attacks on both friendships and activities. (Section VI and VII)

## II. Background and Intuition

In this section, we first introduce the current state-of-the-art approaches on sybil detection, i.e., *graph-based* sybil detection, for online social networks, as well as state the fundamental limitations of such approaches. Then we highlight our intuition to develop a practical sybil attack model, which enables us to design effective detection algorithms to address these fundamental limitations.

### A. Graph-based Sybil Detection and Limitations

Graph-based sybil detection (or defense) in online social networks (OSNs) has been an active area of research. The canonical formulation is that users in an OSN are classified either into *honest nodes* and *sybil nodes*, and the objective is to identify these sybil nodes by simply relying on the friendship graph. The mainstream methodologies, e.g., [7]–[16], assume an attack model that sybils can establish only a limited number of links (or friendships) with honest nodes (in this work, we call such links as "*attack edges*"). We refer to this assumption as the "*limited-attack-edges assumption*", which leads to the following fundamental limitation: *to guarantee an accurate detection of sybil nodes, each sybil node can launch at most $O(1/\log(|\mathcal{V}|))$ attack links on average, where $\mathcal{V}$ is the set of all nodes in the network* [7].

However, recent studies revealed that the limited-attack-edges assumption does not hold in real-world OSNs. In particular, Yang et al. [17] found that in RenRen, a popular online social network in China, each sybil node could launch many friend requests to honest users. More importantly, around $26\%$ of such requests were accepted. In other words, the number of attack edges is much higher than previously assumed. Sridharan et al [18] also found that in Twitter, a large number of spam accounts could attract honest nodes to be their followers, and these spam accounts (or nodes) become deeply embedded in Twitter. Moreover, attack edges can be established automatically. For example in Facebook, socialbots managed to get an average request acceptance of up to $80\%$ [19]. Furthermore, as shown by results in recent studies [20], [21], if one relaxes the limited-attack-edges assumption, it will lead to low detection accuracy. All the above evidences point to the fact that sybil attack model based on the limited-attack-edges assumption is not practical, and purely exploiting the friendship graph to detect sybils is quite limited in real-world OSNs. This motivates us to investigate *practical sybil attack models* and design *effective sybil detection algorithms*.

### B. Main Intuition

Our intuition is that the social activities (e.g., tweets or retweets) among users contain rich information, which can enable us to differentiate the sybil nodes from honest nodes. For instance, in real life one may exchange business cards with strangers but people will also be more cautious in selecting whom to further interact with. This behavior is in line with users in OSNs, i.e., honest users may be willing to establish links with sybils, however, they seldom interact with sybils. In fact, this user behavior in OSNs has been justified by an analysis of a dataset containing thousands of sybils in Twitter by Zhang et al. [22], which showed that non-sybil users tend to be more selective in retweeting/replying to, and mentioning other users. These observations enable us to develop a practical sybil attack model, or the "*social-and-activity-based sybil attack model*", which will be presented in Section III.

One naive approach to detect sybils in our social-and-activity-based sybil attack model is composed of two steps: (1) First, one can address the limited-attack-edges assumption by



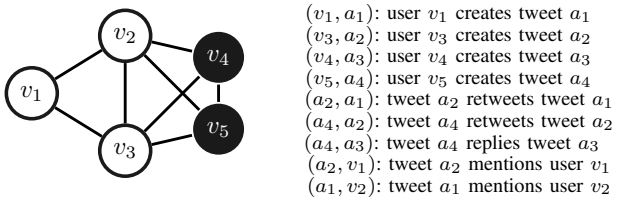
Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

using the social activities among users to adjust the weights or even delete some links on the friendship graph. For example, one can delete the friendship link between two users when they interact less than a given number (usually small) of times. (2) Then, apply the existing graph-based sybil detection algorithm to detect sybils. However, the drawback of this naive approach is that in real-world OSNs, there are many users who are not very active in interacting with others. In particular, by analyzing a sub-network of Twitter, we found that 113,214 out of 409,694 users (or around 28% of users) interact with others at most only one time. These users may be misclassified as sybils, since they also seldom interact with other honest users, leading to low accuracy of the detection algorithms [23] (we will further justify this in our experiments). The reason is that compressing social activities to friendship graph cannot fully utilize the activity data. This motivates us to explore an interesting and fundamental question: *How to fully utilize the advantages of both users' friendship graph and their activities to detect sybils?* We aim to address this question, and refer to our approach as the *social-and-activity-based sybil detection*.

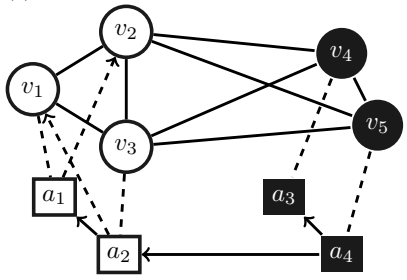
## III. Sybil Attack Model

In this section, we first formulate a SAN to characterize the friendships and historical social activities in an online social network. Then, based on the SAN, we present our sybil attack model. Finally, we introduce our main objective.

### A. The Social and Activity Network Model




**(a) Social network and users' activities**




**(b) Constructed SAN**

**Fig. 2: An example in Twitter for constructing a SAN.**

We formulate a two-layer *hyper-graph* to unify users' friendships and historical activities. These two layers are:
**Layer 1: Friendship graph.** We use an undirected[1] graph

$$\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E})$$

to characterize the friendship between users. The set

$$\mathcal{V} \triangleq \{v_1, \ldots, v_{|\mathcal{V}|}\}$$

denotes all users (or nodes) in a social network. The node set can be partitioned into a subset of honest nodes, which is denoted by $\mathcal{V}_h$, and a subset of sybil nodes $\mathcal{V}_s$, where $\mathcal{V}_h \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V} = \mathcal{V}_h \cup \mathcal{V}_s$. For example, Figure 2a depicts a social network of 5 nodes, i.e., $\mathcal{V} = \{v_1, \ldots, v_5\}$, three honest nodes $\mathcal{V}_h = \{v_1, v_2, v_3\}$ and two sybil nodes $\mathcal{V}_s = \{v_4, v_5\}$. The set

$$\mathcal{E} \subseteq \{(v_i, v_j) | v_i, v_j \in \mathcal{V}, v_i < v_j\}$$

denotes all the undirected edges in a social network, where $(v_i, v_j) \in \mathcal{E}$ represents friendship between node $v_i$ and $v_j$. For each edge $(v_i, v_j)$, we assume $v_i < v_j$ for the purpose of eliminating the redundancy that $(v_i, v_j)$ and $(v_j, v_i)$ represent the same undirected edge. For example, Figure 2a shows a social network with eight edges. The edge $(v_1, v_2)$ represents a friendship between honest nodes $v_1$ and $v_2$, and $(v_2, v_5)$ shows a friendship between an honest node $v_2$ and a sybil $v_5$.
**Layer 2: Activity graph.** We use a mixed graph (containing both directed and undirected edges)

$$\widetilde{\mathcal{G}} \triangleq (\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \mathcal{F})$$

to characterize the historical activities. The set

$$\mathcal{A} \triangleq \{a_i | i = 1, \ldots, |\mathcal{A}|\}$$

denotes a set of all activity nodes. An activity node can be interpreted as a tweet (or retweet, etc.) in Twitter, or a post (or a comment, etc.) in Facebook. Figure 2b shows 4 activity nodes $\mathcal{A} = \{a_1, \ldots, a_4\}$. Each activity node is associated with *only one* creator, i.e., a node in the friendship graph. We use an undirected edge $(v, a)$, where $v \in \mathcal{V}$ and $a \in \mathcal{A}$, to represent that the user $v$ creates the activity $a$. The set

$$\mathcal{C} \triangleq \{(v, a) | v \in \mathcal{V}, a \in \mathcal{A}\}$$

denotes a set of edges which reflect the creator-activity relationships. For example, in Figure 2b, the edge $(v_1, a_1)$ means that user $v_1$ creates the activity $a_1$. We use a directed edge from an activity to a user $(a, v)$, where $a \in \mathcal{A}$ and $v \in \mathcal{V}$, to represent that the activity $a$ mentions the user $v$. In Figure 2b, the directed edge $(a_2, v_1)$ can be interpreted as that user $v_1$ was mentioned in the tweet $a_2$. Note that an activity can mention *multiple* users. The set

$$\mathcal{M} \triangleq \{(a, v) | a \in \mathcal{A}, v \in \mathcal{V}\}$$

denotes a set of all directed edges indicating mention relationships. We use a directed edge from $a_i \in \mathcal{A}$ to $a_j \in \mathcal{A}$, i.e., $(a_i, a_j)$, to represent that the activity $a_i$ follows the activity $a_j$. Here, the following behavior can be interpreted as replying or retweeting in Twitter, or commenting one's post in Facebook, etc. In Figure 2b, the edge $(a_2, a_1)$ can be interpreted as that the tweet $a_2$ is a retweet of the tweet $a_1$. The set

$$\mathcal{F} \triangleq \{(a_i, a_j) | a_i, a_j \in \mathcal{A}\}$$

denotes a set of all directed edges indicating following relationships.

***Definition* 1 (Interaction).** *We define each directed edge in $\mathcal{M}$ or $\mathcal{F}$ as an interaction.*

[1]For directed OSNs like Twitter, we can transform it to an undirected one via keeping an edge between two nodes only if they follow each other.



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

Namely, each directed edge in $\mathcal{M}$ or $\mathcal{F}$ corresponds to one interaction, and the set of all interactions is $\mathcal{M} \cup \mathcal{F}$. Note that an activity may involve multiple interactions. For example, In Figure 2b, the activity $a_2$ involves two interactions, i.e., $(a_2, a_1)$ and $(a_2, v_1)$.

### *B.* The Sybil Attack Model

In the social-and-activity-based sybil attack model, sybils can launch both friendship attacks and activity attacks.

**Friendship attacks.** Let $\mathcal{G}_h \triangleq (\mathcal{V}_h, \mathcal{E}_h)$ denote the *honest region*, which is the subgraph induced by honest nodes $\mathcal{V}_h$ in $\mathcal{G}$. Likewise, we refer to the subgraph induced by sybil nodes $\mathcal{V}_s$ in $\mathcal{G}$ as the *sybil region*, denoted by $\mathcal{G}_s \triangleq (\mathcal{V}_s, \mathcal{E}_s)$. In Figure 2a, we have $\mathcal{V}_h = \{v_1, v_2, v_3\}$, $\mathcal{E}_h = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$, $\mathcal{V}_s = \{v_4, v_5\}$ and $\mathcal{E}_s = \{(v_4, v_5)\}$.

***Definition* 2.** *We define friendship attack edges as the friendship links between the honest region and sybil region, i.e., $\mathcal{E} \setminus (\mathcal{E}_h \cup \mathcal{E}_s)$, and define the number of friendship attack edges as*

$$N_A \triangleq |\mathcal{E} \setminus (\mathcal{E}_h \cup \mathcal{E}_s)|$$

In Figure 2a, we have $N_A = 4$.

***Property* 1.** *$N_A$ can take any value in $\{0, 1, \ldots, |\mathcal{V}_h| \times |\mathcal{V}_s|\}$.*

Note that we do not restrict sybils' capabilities in establishing friendship attack edges. Property 1 is practical and addresses the fundamental limitation (as described in Section II) of the previous graph-based sybil attack model [7]–[14].

**Activity attack.** Let $\mathcal{A}_h \subseteq \mathcal{A}$ and $\mathcal{A}_s \subseteq \mathcal{A}$ denote a set of all activities created by honest users (i.e., nodes in $\mathcal{V}_h$), and created by sybils (i.e., nodes in $\mathcal{V}_s$) respectively, where $\mathcal{A}_h \cap \mathcal{A}_s = \emptyset$ and $\mathcal{A}_h \cup \mathcal{A}_s = \mathcal{A}$. In Figure 2b, we have $\mathcal{A}_h = \{a_1, a_2\}$ and $\mathcal{A}_s = \{a_3, a_4\}$. Let $\widetilde{\mathcal{G}}_h \triangleq (\mathcal{V}_h, \mathcal{A}_h, \mathcal{C}_h, \mathcal{M}_h, \mathcal{F}_h)$ denote the subgraph induced by $\mathcal{V}_h \cup \mathcal{A}_h$ in $\widetilde{\mathcal{G}}$. Namely, $\widetilde{\mathcal{G}}_h$ is the activity graph restricted to the honest region. In Figure 2b, we have $\mathcal{C}_h = \{(v_1, a_1), (v_3, a_2)\}, \mathcal{M}_h = \{(a_2, v_1), (a_1, v_2)\}$, and $\mathcal{F}_h = \{(a_2, a_1)\}$. Based on this activity graph, define

$$W_h \triangleq |\mathcal{M}_h| + |\mathcal{F}_h|$$

as the number of interactions among honest users. In Figure 2b, we have $W_h = 1 + 2 = 3$. The larger the $W_h$ is, the more active the honest users are in interacting with other honest users. Similarly, let $\widetilde{\mathcal{G}}_s \triangleq (\mathcal{V}_s, \mathcal{A}_s, \mathcal{C}_s, \mathcal{M}_s, \mathcal{F}_s)$ denote the subgraph induced by $\mathcal{V}_s \cup \mathcal{A}_s$ in $\widetilde{\mathcal{G}}$. Namely, $\widetilde{\mathcal{G}}_s$ is the activity graph restricted to the sybil region only. In Figure 2b, we have $\mathcal{C}_s = \{(v_5, a_4), (v_4, a_3)\}, \mathcal{M}_s = \emptyset$, and $\mathcal{F}_s = \{(a_4, a_3)\}$. We further define

$$W_s \triangleq |\mathcal{M}_s| + |\mathcal{F}_s|$$

as the number of interactions among sybils. In Figure 2b, we have $W_s = 1 + 0 = 1$.

***Property* 2.** *$W_s$ can be arbitrarily large.*

Namely, we consider the general scenario that sybils can create an arbitrary number of interactions among themselves so as to reduce the chance to be detected.

One type of attack which can be launched by sybil nodes is the "*incoming interaction attack*".

***Definition* 3.** *We define the incoming interaction attack as the directed edges from the honest activity graph $\widetilde{\mathcal{G}}_h$ to the sybil activity graph $\widetilde{\mathcal{G}}_s$, i.e., $\mathcal{F}_{h\to s} \cup \mathcal{M}_{h\to s}$ , where $\mathcal{F}_{h\to s}, \mathcal{M}_{h\to s}$ denote the following edges and mentioning edges respectively:*

$$\mathcal{F}_{h\to s} \triangleq \{(a_h, a_s) | a_h \in \mathcal{A}_h, a_s \in \mathcal{A}_s, (a_h, a_s) \in \mathcal{F}\},$$
$$\mathcal{M}_{h\to s} \triangleq \{(a_h, v_s) | a_h \in \mathcal{A}_h, v_s \in \mathcal{V}_s, (a_h, v_s) \in \mathcal{M}\}.$$

Namely, $\mathcal{F}_{h\to s} \cup \mathcal{M}_{h\to s}$ contains all the interactions that are initiated from honest users to sybils. In Figure 2b, we have $\mathcal{F}_{h\to s} = \emptyset$ and $\mathcal{M}_{h\to s} = \emptyset$. We also define the intensity that honest nodes initiate interactions to sybil nodes as

$$\alpha \triangleq |\mathcal{F}_{h\to s} \cup \mathcal{M}_{h\to s}| / W_h$$

The smaller the $\alpha$ is, the less willing the honest nodes are in initiating interactions to sybil nodes. As we have discussed in Section II that honest users are quite selective in initiating interactions to sybil users. We use the following assumption to capture this observation.

***Assumption* 1.** *The value $\alpha$ is usually small, i.e., $\alpha << 1$.*

According to the experiments in [22], $\alpha \approx 4.2 \times 10^{-5}$.

Another activity attack is the "*outgoing interaction attack*".

***Definition* 4.** *We define the outgoing interaction attack as the directed edges from the sybil activity graph $\widetilde{\mathcal{G}}_s$ to the honest activity graph $\widetilde{\mathcal{G}}_h$, i.e., $\mathcal{F}_{s\to h} \cup \mathcal{M}_{s\to h}$, where $\mathcal{F}_{s\to h}, \mathcal{M}_{s\to h}$ denote the following edges and mentioning edges respectively:*

$$\mathcal{F}_{s\to h} \triangleq \{(a_s, a_h) | a_s \in \mathcal{A}_s, a_h \in \mathcal{A}_h, (a_s, a_h) \in \mathcal{F}\},$$
$$\mathcal{M}_{s\to h} \triangleq \{(a_s, v_h) | a_s \in \mathcal{A}_s, v_h \in \mathcal{V}_h, (a_s, v_h) \in \mathcal{M}\}.$$

Namely, $\mathcal{F}_{s\to h} \cup \mathcal{M}_{s\to h}$ contains all the interactions that are initiated from sybils to honest users. In Figure 2b, we have $\mathcal{F}_{s\to h} = \{(a_4, a_2)\}, \mathcal{M}_{s\to h} = \emptyset$. Similarly, we define the intensity that sybils initiate interactions to honest nodes as

$$\beta \triangleq |\mathcal{F}_{s\to h} \cup \mathcal{M}_{s\to h}| / W_h$$

The larger the $\beta$ is, the more aggressive the sybils are in initiating interactions to honest users.

***Property* 3.** *The value of $\beta$ can be arbitrarily large.*

Namely, we consider the general case that sybils can initiate arbitrarily number of interactions to honest users.

### *C.* Our objective

*Given $\mathcal{G}$, $\widetilde{\mathcal{G}}$ and a small set of labelled seed nodes $\mathcal{S}$, design an algorithm to detect the sybil nodes.* The $\mathcal{S}$ can contain both honest nodes and sybil nodes, or only one type of them.

## IV. Sybil Detection Algorithm Design

Here, we present our Sybil_SAN algorithm, which propagates the trust (distrust) from honest (sybil) seed nodes to other nodes through social and activity network via coupling three random walks [24], We also design an iterative algorithm to compute the trust scores and distrust scores.



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

### A. *Design Overview*

We first consider the case that $\mathcal{S}$ contains honest nodes only. Then we extend to consider that $\mathcal{S}$ contains sybil nodes only, and finally the both honest nodes and sybil nodes case.

Given a small set $\mathcal{S}$ of known honest users, our objective is to evaluate the trustworthiness (i.e., numerical trust scores) of other user nodes, which may be honest or sybil nodes, based on the seeds $\mathcal{S}$ and the SAN graph $\mathcal{G}$, $\widetilde{\mathcal{G}}$. We rank the user nodes according to their trustworthiness in a descending order, and take users with low rank as suspects of sybils. In particular, we apply the "*random walk framework*" to evaluate the trust score for each node, because this framework is easy to implement, computationally efficient and easy to interpret. More precisely, at the beginning of the random walk, the total trust score for the seeds $\mathcal{S}$ is normalized to be one, and each seed evenly shares the trust score, i.e.,

$$s_i = \begin{cases} 1/|\mathcal{S}|, & \text{if } v_i \in \mathcal{S}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $i = 1, \ldots, |\mathcal{V}|$ and $s_i$ denotes the trust score for node $v_i \in \mathcal{V}$. Here $s_i = 0, \forall v_i \notin \mathcal{S}$ models that we assign the minimum trust score for the users outside the seed set. Let $s_{|\mathcal{V}|+i}$, where $i = 1, \ldots, \mathcal{A}$, denote the trust score of activity $a_i$. Initially, we set $s_{|\mathcal{V}|+i} = 0$, for all $i = 1, \ldots, \mathcal{A}$, capturing that the initial trust score for each activity is zero. We denote the initial trust score vector as $\mathbf{s} \triangleq [s_1, \ldots, s_{|\mathcal{V}|+|\mathcal{A}|}]^T$. From the random walk perspective, the initial trust score vector $\mathbf{s}$ corresponds to initial probability distribution, i.e., the walk starts from user node $v_i$ with probability $s_i$. Walking on the SAN graph corresponds to the propagation of trust.

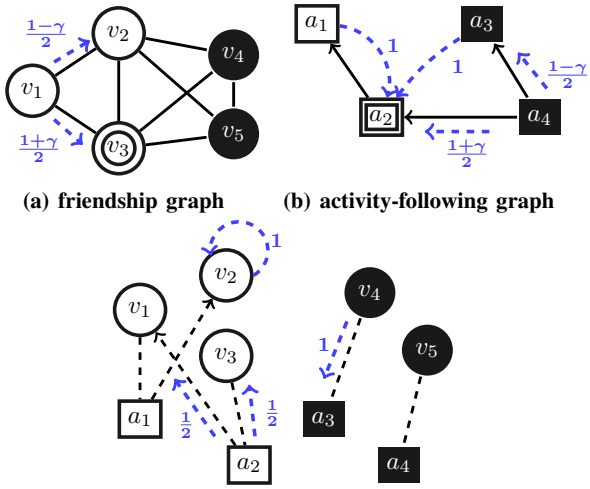
***Definition* 5 (Trust score).** *We define the stationary probability distribution (or landing probability) of the random walk, which starts from seeds nodes, as nodes' trust score.*

There are two challenges: *(1) How to design the walking strategy to capture the trust propagation on SAN graph? (2) How to prove the convergence of the random walk and derive the number of rounds needed to converge?*

Our design of random walk strategy is motivated by the mutual reinforcement relationship between users and activities: the activities of a trusted user can be trusted, while an activity with high trust score can certify the trustiness of its creator. Thus, we first decompose the SAN into three subnetworks. For each subnetwork, we design a random walk to propagate trust independently on it. Finally, we present a unified algorithm to couple these three random walks to capture the mutual reinforcement relationship between users and activities.

### B. Decomposed Random Walk

We decompose the SAN network into three subgraphs: (1) the friendship graph $(\mathcal{V}, \mathcal{E})$; (2) the activity-following graph $(\mathcal{A}, \mathcal{F})$; (3) the user-activity graph $(\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M})$. For example, Figure 3 presents a decomposition of the SAN network in Figure 2b. Figure 3 also presents the one-step transition probabilities, which correspond to the random walk strategies in each subgraph (presented in Algorithm 1).




**(a) friendship graph** **(b) activity-following graph**

**(c) user-activity mapping graph**

**Fig. 3: Decomposed random walks in toy example**

**Algorithm 1:** Decomposed random walks

**Procedure** `WalkOnFriendGraph` $(\mathcal{V}, \mathcal{E}, \mathbf{s}, \gamma, v_i)$**:**
  Given the walker is at the node $v_i$
  Walk to $v_j$ with prob. $(1-\gamma)\frac{\mathbb{1}_{\{(v_i,v_j)\in\mathcal{E}\}}}{\sum_{v_\ell\in\mathcal{V}}\mathbb{1}_{\{(v_i,v_\ell)\in\mathcal{E}\}}} + \gamma s_j$

**Procedure** `WalkOnActivityFollowingGraph` $(\mathcal{A}, \mathcal{F}, \mathcal{S}_A, \gamma, a_i)$**:**
  Given the walker is at the activity $a_i$
  $Outdeg(a_i) \leftarrow |\{a_j|(a_i, a_j) \in \mathcal{F}\}|$
  If $Outdeg(a_i) \geq 1$, walk to activity $a_j$ with probability $(1-\gamma)\frac{\mathbb{1}_{\{(a_i,a_j)\in\mathcal{F}\}}}{Outdeg(a_i)} + \gamma\frac{\mathbb{1}_{\{a_j\in\mathcal{S}_A\}}}{|\mathcal{S}_A|}$,
  else walk to $a_j$ with probability $\mathbb{1}_{\{a_j\in\mathcal{S}_A\}}/|\mathcal{S}_A|$

**Procedure** `WalkOnUserActivityGraph` $(\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M})$**:**
  **if** *The walker is at user node* $v_i$ **then**
    $deg(v_i) \leftarrow |\{a_j|(v_i, a_j) \in \mathcal{C}\}|$
    If $deg(v_i) = 0$, stays at node $v_i$, else walks to $a_j$ with probability $\mathbb{1}_{\{(v_i,a_j)\in\mathcal{C}\}}/deg(v_i)$
  **if** *The walker is at activity node* $a_i$ **then**
    $deg(a_i) \leftarrow |\{v_j|(v_j, a_i) \in \mathcal{C}\}| + |\{v_j|(a_i, v_j) \in \mathcal{M}\}|$
    Walks to $v_j$ with probability $\mathbb{1}_{\{(v_j,a_i)\in\mathcal{C}\}}/deg(a_i) + \mathbb{1}_{\{(a_i,v_j)\in\mathcal{M}\}}/deg(a_i)$

**Random walk on the friendship graph** $(\mathcal{V}, \mathcal{E})$**.** Note that we need to make a balance between exploiting the trust seeds and exploring nodes with unknown trust score, i.e., $v_i \notin \mathcal{S}$. We use the naive random walk to do the exploration, i.e., the walker jumps to one of its neighbor with equal probability, i.e., $\mathbb{1}_{\{(v_i,v_j)\in\mathcal{E}\}}/\sum_{v_\ell\in\mathcal{V}}\mathbb{1}_{\{(v_i,v_\ell)\in\mathcal{E}\}}$. The physical interpretation is that each node distributes its trust score equally to its neighbors. To exploit the trust of the seeds, the walker jumps to one of the seed nodes with equal probability, i.e., $1/|\mathcal{S}|$. This captures that the node distributes all its trust score to one of the seeds, which can be further used to assign more credits to those



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

nodes that the seed trusts more. With probability $\gamma \in [0, 1]$ the walker walks according to the naive random walk, and with probability $1 - \gamma$ the walker jumps to one of the seeds. We present the detail of this walking strategy in Algorithm 1. Figure 3a illustrates the one-step transition probability of the walker on node $v_1$ with one seed node $v_3$.

**Random walk on the activity-following graph $(\mathcal{A}, \mathcal{F})$.** The trust can be propagated from $a_i$ to $a_j$ only if $a_i$ follows $a_j$, i.e., $a_j$ is an outgoing neighbor of $a_i$. Recall that $a_i$ follows $a_j$ can be interpreted as that $a_i$ is a retweet of $a_j$ in Twitter. Then propagating trust from $a_i$ to $a_j$ captures the behavior that a user retweets $a_j$ only when he trusts the tweet $a_j$. For the activities having at least one outgoing neighbors, the walking strategy is the same as that on the friendship graph. The activity $a_4$ in Figure 3b illustrates this case, where $a_2$ is the trust seed, i.e., created by the seed node $v_3$. It may happen that an activity does not have any outgoing neighbor. For such activities, the walker jumps to one of the activity seeds with equal probability, i.e., $1/|\mathcal{S}_A|$, where $\mathcal{S}_A$ denotes the activities created by the seed nodes, i.e., $\mathcal{S}_A \triangleq \{a | a \in \mathcal{A}, (v, a) \in \mathcal{C}, v \in \mathcal{S}\}$. The activity $a_3$ and $a_1$ in Figure 3b illustrate this case. We present the detail of this walking strategy in Algorithm 1.

**Walking on the user-activity graph $(\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M})$.** This is a mixed graph, i.e., containing directed edges and undirected edges. The trust can be propagated from a user node to an activity node only if this user creates this activity, i.e., there is an undirected edge between them. The trust can be propagated from an activity node to a user node only if the user is its creator or there is a directed edge from the activity to the user (capturing that the activity mentions the user out of trust). Thus, we interpret each undirected edge as two directed edges. If a node (user or activity) has at least one outgoing neighbors, the walker walks to one of these neighbors uniformly at random. The node $v_4$ and node $a_2$ in Figure 3c illustrate this case. If a node (user or activity) does not have any outgoing neighbor, the walker remains at this node. The node $v_2$ in Figure 3c illustrates this case. We present the detail of this walking strategy in Algorithm 1.

### C. Coupling the Random Walks

Now, we couple these three random walks together to capture the mutual reinforcement relationships between users and activities. Recall that the walker starts with node $v_i$ with probability $s_i$. Then at each step, we couple the random walk as Algorithm 2. We will show in Section V, the coupled random walk converges to a unique landing probability distribution, or the trust scores converge.

Let $\mathbf{s}^*$ denote the converged trust score. To compute it, we need to derive the transition matrix associated with the coupled random walk. Let the square matrix $\mathbf{P} = [P_{i,j}]$ with order $|\mathcal{V}|$ denote the one-step transition matrix associated with the random walk on the friendship graph, where the $i$-th column (or row) corresponds to user node $v_i$. For example, in Figure 3a, $P_{1,2} = (1 - \gamma)/2$. Let the square matrix $\widetilde{\mathbf{P}} = [\widetilde{P}_{i,j}]$ with order $|\mathcal{A}|$ denote the one-step transition matrix

**Algorithm 2:** Coupling random walk

1 The walker starts with node $v_i$ with probability $s_i$.
2 **repeat**
3  **if** *The walker is at a user node* $v_i \in \mathcal{V}$ **then**
4  With probability $\lambda_i$ it walks one step on the friendship graph according to the *WalkOnFriendGraph* algorithm.
5  With probability $(1 - \lambda_i)$, it walks $2k + 1$ steps on the user-activity graph according to the *WalkOnUserActivityGraph* algorithm.
6  **if** *The walker is at an activity node* $a_i \in \mathcal{A}$ **then**
7  With probability $\lambda_{|\mathcal{V}|+i}$, the walker walks $n$ steps on the activity-following graph according to the *WalkOnActivityFollowingGraph* algorithm.
8  With probability $(1 - \lambda_{|\mathcal{V}|+i})$ it takes $2k + 1$ steps on the user-activity graph according to the *WalkOnUserActivityGraph* algorithm.
9 **until** *converge*

associated with the random walk on the activity-following graph, where the $i$-th column (or row) corresponds to activity $a_i$. In Figure 3b, $\widetilde{P}_{4,2} = \frac{1+\gamma}{2}$. Let $\widehat{\mathbf{P}} = [\widehat{P}_{i,j}]$ denote the one-step transition matrix associated with the random walk on the user-activity graph, which is a square matrix of order $(|\mathcal{V}| + |\mathcal{A}|)$. We index the element of $\widehat{\mathbf{P}}$ such that in each column (and each row) the indexes from 1 to $|\mathcal{V}|$ correspond to users, i.e., index $i$ corresponds to $v_i$, and the indexes from $|\mathcal{V}| + 1$ to $(|\mathcal{V}| + |\mathcal{A}|)$ correspond to activities, i.e., index $i$ corresponds to $a_{i-|\mathcal{V}|}$. For example, in Figure 3c, we have $\widehat{P}_{2,2} = 1, \widehat{P}_{7,3} = 1/2, \widehat{P}_{4,8} = 1$. Note that from Algorithm 1 one can easily write down the closed form of $P_{i,j}, \widetilde{P}_{i,j}, \widehat{P}_{i,j}$. Here we omit them for brevity. Let $\mathbf{P}_{cr}$ denote the transition matrix associated with the coupled random walk. Then,

$$\mathbf{P}_{cr} = \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & (\widetilde{\mathbf{P}}^T)^n \end{bmatrix} \mathbf{\Lambda} + (\widehat{\mathbf{P}}^T)^{2k+1}(\mathbf{I} - \mathbf{\Lambda})$$

where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_{|\mathcal{V}|+|\mathcal{A}|})$. As we will show in Section V, the converged trust score $\mathbf{s}^*$ is a unique solution of the following linear system: $\mathbf{P}_{cr}\mathbf{s}^* = \mathbf{s}^*, ||\mathbf{s}^*||_1 = 1$. Note that we will set $\lambda_i, i = 1, \ldots, |\mathcal{V}|$ as a small number, since users' activities are more trustful than users' friendships. More details can be seen in Section VI.

Solving the linear system is computationally expensive, thus we develop an iterative algorithm to calculate $\mathbf{s}^*$ (step $6 - 13$ in Algorithm 3). We need to normalize the trust score of users. This design tries to prevent from mistaking honest users with few sources as sybils, and also mistaking sybils with large sources as honest users. Let $\mathcal{N}_i$ denote $v_i$'s friendships, i.e., $\mathcal{N}_i = \{(u, v_i) | (u, v_i) \in \mathcal{E} \text{ or } (v_i, u) \in \mathcal{E}\}$. And let $\mathcal{I}_i$ denote paths where user $v_i$ receives interactions, i.e., $\mathcal{I}_i = \{(a, v_i) | (a, v_i) \in \mathcal{M}\} \cup \{(a_m, a_j, v_i) | (a_m, a_j) \in \mathcal{F}, (v_i, a_j) \in \mathcal{C}\}$. Apparently, $\mathcal{I}_i$ and $\mathcal{N}_i$ forms the sources where $v_i$ receives trust. We normalize the trust score for each user by the number of sources, i.e., by $|\mathcal{N}_i| + |\mathcal{I}_i|$. Finally, we rank users according



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

to the normalized trust in a descending order, and take users with low rank as suspects of sybils.

**Algorithm 3:** Sybil_SAN

**Input:** $\mathcal{G}$, $\widetilde{\mathcal{G}}$, $\mathcal{S}$, $\mathbf{P}, \widetilde{\mathbf{P}}, \widehat{\mathbf{P}}$
**Output:** users' normalized trust scores

$\hat{\mathbf{s}} \leftarrow$ `CoupleWalk` $(\mathbf{P}, \widetilde{\mathbf{P}}, \widehat{\mathbf{P}}, \mathbf{s})$
Get $\mathbf{q}$: $q_i = \hat{s}_i/(|\mathcal{N}_i| + |\mathcal{I}_i|), \forall i = 1, \ldots, |\mathcal{V}|$
**return** $\mathbf{q}$

**Procedure** `CoupleWalk` $(\mathbf{P}, \widetilde{\mathbf{P}}, \widehat{\mathbf{P}}, \mathbf{s})$**:**
  $\mathbf{s}^{(0)} \leftarrow \mathbf{s}, t = 0$
  **repeat**
    $\mathbf{s}^{(t+1)} = \mathbf{P}_{cr}\mathbf{s}^{(t)}$
  **until** $||\mathbf{s}^{(t+1)} - \mathbf{s}^{(t)}|| \leq \epsilon$
  **return** $\mathbf{s}^{(\mathbf{t+1})}$

### *D.* Extensions to sybil seeds

Recall that for an honest user $u$, if he actively initiates activities to user $v$, we say that $u$ trusts $v$, so Sybil_SAN distributes trust from $u$ to $v$. However, given a known sybil, we should not punish (i.e., propagate distrust score to) who he sends activities to, but rather who actively sends activities to the sybil. Thus, for the case that $\mathcal{S}$ contains sybil nodes only, we assign a score of $1/|\mathcal{S}|$ to each sybil, and apply Algorithm 3 on reversed SAN (i.e., reverse the link directions of SAN), to get the distrust vector $\mathbf{s_{dis}}$. For the case that $\mathcal{S}$ contains both honest nodes and sybil nodes, we decompose it into two disjoint subsets each containing only one type of nodes. We then compute the corresponding $\mathbf{s}$ and $\mathbf{s_{dis}}$ for these subsets accordingly. Finally, we compute the trust score vector as $(\mathbf{s} - \mathbf{s_{dis}})$. Note that the calculation of $\mathbf{s}$ and $\mathbf{s_{dis}}$ can be easily paralleled to reduce computational cost.

## V. Convergence and Sensitivity Analysis

In this section, we first prove the convergence of Algorithm 2, then we derive the number of rounds needed to guarantee that Algorithm 3 terminates. Finally, we apply the Matrix perturbation theory to bound the detection error of Sybil_SAN. Due to page limit, please refer to [25] for detailed proofs. Note that for brevity, we only analyze the algorithm for the case that $\mathcal{S}$ contains only honest nodes. This is because the analysis can be easily extended to the case that $\mathcal{S}$ contains only sybil nodes or both honest nodes and sybil nodes.

### *A.* Convergence Analysis

We first apply Markov chain techniques to show the convergence of Algorithm 2.

***Theorem* 1.** *Suppose the friendship graph $\mathcal{G}$ is connected, $0 < \gamma < 1$, and if $i > |\mathcal{V}|$ or user $v_i$ has activities $(\{a|(v_i, a) \in \mathcal{C}\} \neq \emptyset)$, $0 < \lambda_i < 1$. Then Algorithm 2 converges to a unique stationary probability (or landing probability) distribution, which is a unique solution of $\mathbf{P}_{cr}\mathbf{s}^* = \mathbf{s}^*, ||\mathbf{s}^*||_1 = 1$.*

**Remark.** Theorem 1 states sufficient conditions under which Algorithm 2 converges to a unique landing probability distribution, i.e., each user will have a unique trust score.

It is not practical to compute the exact converged landing probability. Algorithm 3 approximates it in an iterative manner. We next derive the number of rounds needed to guarantee that Algorithm 3 hits the stopping condition.

***Theorem* 2.** *Suppose $||\mathbf{s}^{(t+1)} - \mathbf{s}^{(t)}||$ in Algo. 3 is measured by 1-norm, then Algo. 3 stops in at most $1 + \frac{1}{\nu}\ln\left(4/(\epsilon s^*_{\min})\right)$ rounds, where $s^*_{\min} \triangleq \min_i s^*_i$ and $\mathbf{s}^*$ denotes the stationary landing probability distribution. Here $\nu$ denotes the spectral gap of the Markov chain with transition matrix $\mathbf{P}_{cr}$.*

**Remark:** Theorem 2 states that Algo. 3 stops in a finite number of rounds, which is linear in $1/\nu$ ($\nu$ is spectral gap) and $\ln\left(4/(\epsilon s^*_{\min})\right)$ ($s^*_{\min}$ denotes the minimum converged landing probability). For other norms besides the 1-norm, we can derive the number of rounds needed similarly.

### *B.* Sensitivity Analysis

Let $\mathbf{q}^* \triangleq \mathbf{C}\boldsymbol{s}^*$ denote the normalized trust score associated with the converged landing probability $\boldsymbol{s}^*$, where $\mathbf{C}$ is a $|\mathcal{V}| \times (|\mathcal{V}| + |\mathcal{A}|)$ matrix with

$$C_{ij} = \begin{cases} 1/(|\mathcal{N}_i| + |\mathcal{I}_i|), & \text{if } i = j = 1, \cdots, |\mathcal{V}|, \\ 0, & \text{otherwise.} \end{cases}$$

Namely, $\mathbf{q}^*$ is the asymptotically limit of the output of Algorithm 3 when $\epsilon$ goes to zero. Note that $\mathbf{q}^*$ is calculated under sybil attacks. To understand more about the detection accuracy of Algorithm 3, we compare $\mathbf{q}^*$ with the normalized trust score without sybils attacks, denoted by $\widetilde{\mathbf{q}}^*$. Ideally, when sybils do not launch any attacks, we can easily distinguish sybils from honest users with the normalized trust score vector $\widetilde{\mathbf{q}}^*$, because $\widetilde{\mathbf{q}}^*_i > 0$ if $i \in \mathcal{V}_h$, and $\widetilde{\mathbf{q}}^*_i = 0$ if $i \in \mathcal{V}_s$. We use a generic metric to quantify the detection error, i.e., $||\mathbf{q}^* - \widetilde{\mathbf{q}}^*||/||\widetilde{\mathbf{q}}^*||$. The smaller $||\mathbf{q}^* - \widetilde{\mathbf{q}}^*||/||\widetilde{\mathbf{q}}^*||$ is, the smaller error the detection algorithm has.

Let $\widetilde{\mathbf{P}}_{cr}$ denote the transition matrix associated with the coupled random walk, when sybils do not launch any attacks. We define the error matrix caused by sybil attacks as $\mathbf{E} \triangleq \widetilde{\mathbf{P}}_{cr} - \mathbf{P}_{cr}$. We next apply matrix perturbation theory to quantify the impact of $\mathbf{E}$ on the converged trust score.

***Theorem* 3.** *Let $\widetilde{\boldsymbol{s}}^*$ denote the converged trust score under $\widetilde{\mathbf{P}}_{cr}$. We have*

$$||\boldsymbol{s}^* - \widetilde{\boldsymbol{s}}^*||/||\widetilde{\boldsymbol{s}}^*|| \leq \epsilon_{sd} \tag{2}$$

*where $\epsilon_{sd}$ is defined as*

$$\begin{aligned} \epsilon_{sd} \triangleq & ||[(\mathbf{P}_{cr} - \mathbf{I})(\mathbf{P}_{cr}^T - \mathbf{I}) + \mathbf{e}^T\mathbf{e}]^{-1} \times \\ & \left(\mathbf{E}(\mathbf{P}_{cr} - \mathbf{I} + \mathbf{E}^T) + (\mathbf{P}_{cr} - \mathbf{I})\mathbf{E}^T\right)|| \end{aligned} \tag{3}$$

*the notation $||\cdot||$ denotes a general matrix norm, $\mathbf{e} \triangleq [1, \ldots, 1]$ denotes a $|\mathcal{V}|+|\mathcal{A}|$ dimensional row vector with all 1 elements and $\mathbf{I}$ denotes an identity matrix with order $|\mathcal{V}| + |\mathcal{A}|$.*

**Remark.** Theorem 3 states an upper bound on the error in converged landing probability (i.e., trust score) caused by $\mathbf{E}$ (i.e., sybil attacks). It states that the error will be small when the norm of $\mathbf{E}$ is small (i.e., sybils launch few attacks).

We next apply Theorem 3 to quantify the impact of $\mathbf{E}$, i.e., sybil attacks, on the normalized trust score.



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

***Corollary* 1.** *We can bound the error between* $\mathbf{q}^*$ *and* $\widetilde{\mathbf{q}}^*$ *as*

$$||\mathbf{q}^* - \widetilde{\mathbf{q}}^*||/||\widetilde{\mathbf{q}}^*|| \leq \epsilon_{sd}||\mathbf{C}||||\widetilde{\boldsymbol{s}}^*||/||\widetilde{\mathbf{q}}^*|| \quad (4)$$

**Remark.** Corollary 1 states an upper bound on the detection error caused by sybil attacks (i.e., $\mathbf{E}$). It states that the detection error will be small when the norm of $\mathbf{E}$ is small (i.e., few sybil attacks).

## VI. Experiments on Synthetic Data

In this section, we conduct experiments on synthetic data to extensively evaluate the impact of various factors, e.g., number of attack edges, on the accuracy of our Sybil_SAN algorithm.

### A. Experimental setups

**Datasets**: We use a real-world social network as the honest region, while synthesizing the sybil region. This method has been used in previous works [7], [9], [12], [13], [21], Our honest region is a public Twitter dataset [26] with $543,785$ nodes, $28,397,413$ reciprocal following edges and $214,267,09$ interactions among users, i.e., $W_h = 214,267,09$. Two types of interactions exist: 1) user $v_i$ retweets user $v_j$'s tweets; 2) user $v_i$ mentions user $v_j$. Given a set of configuration parameters $(N_s, N_A, \alpha, \beta, M)$, we synthesize the sybils as follows.

- **Sybil region**: Instead of focusing on one connected sybil region, here we consider a more practical scenario where sybils region is formed by several disconnected clusters, since in reality attackers at different company/country create their own sybil region and such regions may not always be connected to other sybils region. Specifically, we create $M$ identical clusters, and all together $N_s \in \mathbb{N}_+$ sybil nodes are created. In each cluster, we synthesize their friendship network using the *Preferential Attachment* (PA) model [27], which is a widely used method to generate networks. The number of interactions between any two sybils is a random number in $[0, w]$, where $w \in \mathbb{N}_+$. And the type of each activity is randomly chosen from two types existed in honest region.
- **Attacks on friendships and activities**: For each sybil cluster, we randomly attach $\lfloor \frac{N_A}{M} \rfloor$ friendship attack edges between honest region and the sybil cluster. We also initiate $\lfloor \frac{\alpha W_h}{M} \rfloor$ interactions from honest region to the sybil cluster, as well as $\lfloor \frac{\beta W_h}{M} \rfloor$ interactions from the sybil cluster to honest region.

Unless we state otherwise, we use the following default parameters to synthesize the data: $N_s = 10000, w = 2, N_A = 200000, \alpha = 0.00001, \beta = 0.0001, M = 5$. We set $w = 2$ by default because we find that the number of interactions between any two sybils in our crawled subnetwork of Twitter is no more than 2.

**Performance Metric** Follow previous works [15], [16], [21], we use *Area Under the Receiver Operating Characteristic Curve* (AUC) to evaluate the generated rank of users in sybil detection algorithms, which ranks users in a descending order according to users' trustworthiness (i.e., normalized trust score). In essense, AUC is the probability that a randomly selected honest user is ranked higher than a randomly chosen sybil. Let $\mathbf{q} \triangleq [q(v_i) : i = 1 \ldots, |\mathcal{V}|]$ denote the vector of users' trustworthiness. Formally, the AUC is defined as:

$$AUC(\mathbf{q}) = \frac{\sum_{v_i \in \mathcal{V}_h, v_j \in \mathcal{V}_s} \left( \mathbb{1}_{q(v_i) > q(v_j)} + 0.5 \times \mathbb{1}_{q(v_i) = q(v_j)} \right)}{|\mathcal{V}_h| \times |\mathcal{V}_s|}$$

Higher AUC indicates a higher accuracy of the sybil detection algorithm. The case AUC$= 1.0$ ($100\%$) indicates a perfect classifier, i.e., all sybils are ranked lower than honest users. On the other hand, AUC$= 0$ indicates that all sybils are ranked higher than honest users. AUC$= 0.5$ means a random ranking of honest users and sybils.

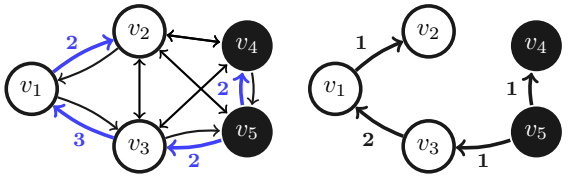
**Compared methods**: We compare Sybil_SAN algorithm with the following five state-of-the-art detection methods:

- **SR-U**: SybilRank [7] on the friendship graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
- **SR-W**: The first simple way to leverage both friendships and social activities to detect sybils: (1) Construct the *weighted friendship graph* via using users' activities to adjust the weights of links; (2) Apply SybilRank on the constructed weighted friendship graph to detect sybils. To illustrate, Figure 4a shows the weighted friendship graph constructed from the social-and-activity network in Figure 2. In Figure 2, $v_2$ and $v_1$ are friends and $v_2$ doesn't initiate any interaction to $v_1$, thus in Figure 4a, the edge $v_2 \to v_1$ has a weight of 1, represented by the black and thinner edge. Meanwhile, since $v_3$ and $v_1$ are friends and $v_3$ initiates 2 interactions to $v_1$ ($(a_2, a_1) \in \mathcal{F}, (a_2, v_1) \in \mathcal{M}$), thus the edge $v_3 \to v_1$ has a weight of 3 in Figure 4a.
- **Inter**: The second way to leverage both friendships and social activities to detect sybils: (1) Construct a *strong friendship graph* via deleting some friendship links based on users' activities; (2) Apply SybilRank to the constructed strong friendship graph to detect sybils. To illustrate, Figure 4b shows the strong friendship graph constructed on the social-and-activity graph in Figure 2. Here, we use the directed version, i.e., we take each undirected friendship link as two directed links and determine whether to remove each directed link. In Figure 2, $v_2$ initiates no interaction to $v_1$, thus the edge $v_2 \to v_1$ is deleted in Figure 4b. Furthermore, $v_3$ actively initiates $2 > 0$ interactions to $v_1$, thus the edge $v_3 \to v_1$ with a weight of 2 exists in Figure 4b. We like to remark that this directed version is better than the undirected version, where we will remove all links and get five isolated nodes, since none of two nodes initiates $\geq 1$ interactions to each other.
- **SScar**: The best belief-propagation-based detection method so far [16].
- **SWalk**: A recent robust random-walk-based detection method [15].

**Parameters of our Sybil-SAN algorithm.** Our seed selection strategy follows previous works [7], [15], [16]. And in each run, all the methods use the same selected seed set. The parameters $(n, k, \lambda_i, \forall i = 1, \ldots, |\mathcal{V}| + |\mathcal{A}|)$ stay fixed in each run of the simulations, more details are described in our technical report [25].



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.




**(a) weighted friendship graph (b) strong friendship graph**

**Fig. 4: An illustrative example of Twitter**

### B. *Impact of Friendship Attack Edges* $N_A$

We consider the default setting as described in Section VI-A, except that we vary the number of friendship attack edges ($N_A$). Figure 5a shows the AUC for six algorithms as $N_A$ varies. One can observe that Sybil_SAN has the highest AUC. Namely, it has the highest accuracy. As we increase $N_A$, the AUC of Sybil_SAN drops slightly, while the AUC for other algorithms drop drastically. Namely, our Sybil_SAN is much more robust against friendship attack edges than other algorithms. It is interesting to observe that when the number of friendship attack edges is around $3 \times 10^6$ (300 attack edge per node), the AUC of our Sybil_SAN algorithm is still above 0.8 (i.e., a high accuracy), while the AUC for SWalk, SSCAR, SR-U, SR-W are below 0.4 (i.e., a low accuracy). The accuracy of Inter algorithm is roughly stable under different $N_A$ around 0.7, because Inter mistakes those users, who are not active in interacting with others as sybils. The low accuracy of SR-W and Inter also shows that naively incorporating the activities into state-of-the-art algorithm does not work.

**Lessons learned.** Sybil_SAN has the highest accuracy, and is robust against a large number of friendship attack edges.

### C. *Impact of Incoming Interaction Attack* $\alpha$

We consider the default setting described in Section VI-A, except that we vary the $\alpha$ from $10^{-6}$ to $10^{-3}$. Recall that authors in [22] found that $\alpha \approx 4.2 \times 10^{-5}$ in real-world OSNs, thus above range can show the robustness of Sybil_SAN in a more general scenario. Remark that a larger $\alpha$ means that a larger number of incoming interaction attack edges exist. Figure 5b shows as $\alpha$ increases, Sybil_SAN always outperforms the other algorithms and its AUC drops slightly as $\alpha$ increase. This shows that Sybil_SAN is also highly robust against incoming interaction attacks. The accuracy of Inter and SR-W drops since more trust will be propagated to sybils as $\alpha$ increase. The AUC for the SR-U and SScar algorithm is flat, because the $\alpha$ does not influence the underlying friendship graph. SWalk is not sensitive to $\alpha$, but its accuracy is much lower than Sybil_SAN. This again shows that: (1) Naively incorporating users' activities doesn't work (SR-W, Inter); (2) Friendship based detection algorithms are quite limited (SScar, SR-U); (3) Sybil_SAN works well to combine the users' friendships and their activities.

**Lessons learned.** Sybil_SAN has the highest accuracy, and is robust against incoming interaction attacks.

### D. *Impact of Number of Sybils* $N_s$

Similar as above, we vary the number of sybils under default settings. One can have two main observations from Figure 5c: (1) Sybil_SAN has the highest AUC, i.e., it outperforms the other methods; (2) As $N_S$ increase, the AUC of each algorithm increases, because the number of friendship attack edges is fixed, and the friendship attack edges become sparse as the number of sybils increases.

**Lessons learned.** Under different number of sybils, our Sybil_SAN algorithm always has the highest accuracy.

### E. *Impact of Outgoing Interaction Attack* $\beta$*:*

We vary the value of $\beta$ to see the impacts of outgoing interaction attack. Note that larger $\beta$ suggests that sybils initiate more interactions to honest users. From Figure 5d, one can observe that Sybil_SAN has the highest AUC, i.e., it outperforms other algorithms under different $\beta$. Furthermore, the AUC for Sybil_SAN, SR-W and Inter increase in $\beta$. This is a good property, which can prevent some sybils from sending too many spam messages to honest users for advertising. On the opposite, SWalk may fail to detect such advertising sybils.

**Lessons learned.** Sybil_SAN significantly outperforms the other algorithms under different number of outgoing interaction attacks. Meanwhile, the more spam messages sybils send to honest users, the more easily sybils will be detected.

### F. *Impact of Structure of Sybil Region*

Figure 5e shows that the AUC decreases in $M$. In other words, it would be more difficult to detect sybils out, if the sybil region is split into a larger number of disconnected clusters. Note that Sybil_SAN still outperforms other algorithms.

We also investigate the effect of number of interactions among sybils ($w$). For more details, one can refer to our technical report [25].

## VII. Experiments On Real Dataset

We conduct experiments on a real-world dataset (from Twitter) and show that our Sybil_SAN algorithm improves the accuracy of the state-of-art algorithms by at least 17.7%.

### A. Experimental Setting

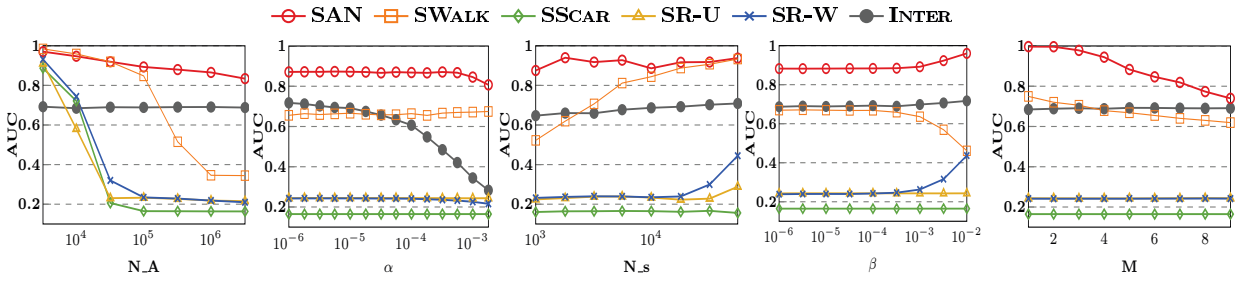
**Datasets.** Starting from 991 public sybils [28] in Twitter, we crawled a network of $450,242$ users and $222,944,310$ links, which contains $409,694$ honest users, $40,548$ sybils and $17,581,069$ friendship attack edges. There are $102,693,769$ interactions among users, containing $714,392$ incoming interaction attacks. More details about crawling process, seed selection and parameter settings can be seen in [25].

### B. *Results and Implications*

We run six algorithms described in Sectioin VI on our real dataset. Table I presents the AUC for each algorithm. One can observe that the AUC of our Sybil_SAN algorithm is the highest, i.e., around 0.73. The AUC of the SR-W, SR-U, SScar and SWalk algorithm are lower or around 0.5, i.e., low accuracy in practice. The Inter algorithm has a higher accuracy



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.




(a) **Impact of $N_A$ on AUC** (b) **Impact of $\alpha$ on AUC** (c) **Impact of $N_s$ on AUC** (d) **Impact of $\beta$ on AUC** (e) **Impact of $M$ on AUC**

**Fig. 5: Experiments on synthetic datasets**

compared to the other four algorithms, and our Sybil_SAN improves the AUC over the Inter by 17.7%. These results again suggest that in practice, detecting sybils solely based on the friendship only can have a very low accuracy, and naively incorporating users' activities does not work well. One remark is that the groundtruth labels of real dataset have some noises. We took those users that are still active in Twitter as honest users, however, there exist some sybils who haven't been suspended and are still active. This may be the possible reason why the detection accuracy in real dataset is not as good as the one in synthetic datasets.

**TABLE I: AUC on real datasets**

| | SR-W | SR-U | Inter | SScar | SWalk | Sybil_SAN |
|---|---|---|---|---|---|---|
| AUC | 0.48 | 0.52 | 0.62 | 0.15 | 0.44 | **0.73** |
| improved ratio | **52.08%** | **40.38%** | **17.74%** | **386%** | **66%** | – |

**Lessons learned.** Our Sybil_SAN has a high accuracy in the real world dataset, and improves the AUC over the state-of-art algorithms by at least 17.7%.

## VIII. Related Work

Feature-based sybil detection uses machine learning techniques to classify users into sybils and honest users according to the features extracted from user-level activities and account details (e.g., profiles, user logs). And this line of works are divided into two branches: unsupervised detection and supervised detection. Unsupervised detection usually clusters sybils according to different features, such as loosely synchronized actions [29], activity patterns [30], users' profiles and tweets [31], click-streams [32], etc. Supervised detection usually uses features of labelled users (e.g., users' profiles, fraction of accepted requests, etc.) to train classifiers to identify sybils [17], [33]. However, sybils can easily evade the detection by adversely changing their behavior accordingly, since classifiers depend on features of known sybils to identify unknown one.

Compared to feature-based detection, graph-based detection is more general to detect sybils with various behaviors. Usually, an OSN is modelled as a graph, with nodes representing users and edges representing users' friendships. Given the assumption that sybils can only befriend with a small number of honest users (i.e., establish few attack edges), the graph is partitioned into honest region and sybil region, with a narrow passage between two regions. A large number of methods leveraged the narrow passage between two regions to detect sybils, for example, [7]–[16], just to name a few. However, recent works [17]–[19] found that sybils are able to befriend with a large number of honest users in real OSNs, which makes sybils in real OSNs easily evade the typical graph-based detection [20], [21]. Our work in this paper is to explore activities among users to handle the sybil detection in real OSNs with a large number of attack edges.

Some recent works also dealt with a large number of attack edges from other aspects. Algorithms in [34], [35] enhanced sybil detection by detecting victims, honest users who befriend with sybils. Effendy et. al. [21] pruned attack edges by exploiting the structure of mutual friendship in OSNs. And Cao et.al [36] and Xue et. al [37] used acceptance and rejection of friend requests to enhance sybil detection. However, above methods still have their own drawbacks. For example, sybils can evade Integro by randomly selecting victims, or degenerate the other two methods by sending friend requests to already established victims to increase mutual friends or probability of requests to be accepted. Different from these works, our work explores activities among users to enhance sybil detection in OSNs. And our work can work in line with above methods for more accurate sybil detection.

## IX. Conclusion

In this paper, we present a practical sybil attack model and design algorithms to detect sybils with performance guarantees. We first develop a SAN to characterize users' friendships and historical activities. Our sybil attack model is based on the SAN and it allows sybils to launch both friendship attacks and activity attacks. We design Sybil_SAN to detect sybils via coupling three random walk-based algorithms on the SAN, and prove the convergence of Sybil_SAN. We develop an iterative algorithm to calculate detection metric for Sybil_SAN and apply Markov chain mixing time to derive the number of rounds needed to guarantee the termination of iteration. We also use matrix perturbation theory to bound the detection error when sybils launch more attacks. Extensive experiments on both synthetic and real-world datasets show that Sybil SAN can detect sybils accurately under practical scenarios, where current state-of-art sybil defenses have a low accuracy. Furthermore, Sybil_SAN is robust against sybil attacks.



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.

## X. Acknowledgment

The work of John C.S. Lui was supported in part by the GRF 14208816 and the Huawei Research Grant.

## References

[1] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 447–462.

[2] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 551–560.

[3] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Catchsync: catching synchronized behavior in large directed graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 941–950.

[4] "Hacking election," https://www.bloomberg.com/features/2016-how-to-hack-an-election/, May 2016, online.

[5] "Sybils in twitter," https://www.nbcnews.com/business/1-10-twitter-accounts-fake-say-researchers-2D11655362, online.

[6] "Sybils in facebook," https://www.nbcbayarea.com/blogs/press-here/Facebook-83-Million-Fake-Users-164848046.html, online.

[7] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 15–15.

[8] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks." in *NDSS*. The Internet Society, 2009.

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

[10] N. Tran, J. Li, L. Subramanian, and S. S. Chow, "Optimal sybil-resilient node admission control," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 3218–3226.

[11] W. Wei, F. Xu, C. C. Tan, and Q. Li, "Sybildefender: Defend against sybil attacks in large social networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1951–1959.

[12] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 3–17.

[13] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 267–278.

[14] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, "Sok: The evolution of sybil defense via social networks," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 382–396.

[15] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*. IEEE, 2017, pp. 273–284.

[16] B. Wang, L. Zhang, and N. Z. Gong, "Sybilscar: Sybil detection in online social networks via local rule based propagation," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[17] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 1, p. 2, 2014.

[18] V. Sridharan, V. Shankar, and M. Gupta, "Twitter games: How successful spammers pick targets," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 389–398.

[19] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proceedings of the 27th annual computer security applications conference*. ACM, 2011, pp. 93–102.

[20] D. Koll, M. Schwarzmaier, J. Li, X.-Y. Li, and X. Fu, "Thank you for being a friend: An attacker view on online-social-network-based sybil defenses," in *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 157–162.

[21] S. Effendy and R. H. Yap, "The strong link graph for enhancing sybil defenses," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on Distributed Computing Systems*. IEEE, 2017, pp. 944–954.

[22] J. Zhang, R. Zhang, J. Sun, Y. Zhang, and C. Zhang, "Truetop: A sybil-resilient system for user influence measurement on twitter," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2834–2846, 2016.

[23] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proceedings of the 4th ACM European conference on Computer systems*. Acm, 2009, pp. 205–218.

[24] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles, "Co-ranking authors and documents in a heterogeneous network," in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 739–744.

[25] J. C. L. Xiaoying Zhang, Hong Xie, "Sybil detection in social-activity networks: Modeling, algorithms and evaluations." https://1drv.ms/b/s!AkqQNKuLPUbEc7bBfXwu9uBZM3w, online.

[26] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality prediction and community structure in social networks," *Scientific reports*, vol. 3, p. 2522, 2013.

[27] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[28] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 963–972.

[29] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 477–488.

[30] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Dna-inspired online behavioral modeling and its application to spambot detection," *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 58–64, 2016.

[31] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, "Twitter spammer detection using data stream clustering," *Information Sciences*, vol. 260, pp. 64–73, 2014.

[32] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection."

[33] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th annual computer security applications conference*. ACM, 2010, pp. 1–9.

[34] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov, "Integro: Leveraging victim prediction for robust fake account detection in osns." 2015.

[35] J. Höner, S. Nakajima, A. Bauer, K.-R. Müller, and N. Görnitz, "Minimizing trust leaks for robust sybil detection," in *International Conference on Machine Learning*, 2017, pp. 1520–1528.

[36] Q. Cao, M. Sirivianos, X. Yang, and K. Munagala, "Combating friend spam using social rejections," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 235–244.

[37] 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 Proceedings IEEE*. IEEE, 2013, pp. 2400–2408.



Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on May 24,2020 at 01:40:56 UTC from IEEE Xplore. Restrictions apply.