Enhancing Sybil Detection via Social-Activity Networks: A Random Walk Approach

Xiaoying Zhang, Hong Xie, Pei Yi, and John C.S. Lui

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. Lastly, we present two extensions of Sybil_SAN to further improve its accuracy.

Index Terms—Sybil detection, social-activity networks, random walk

1 Introduction

Online social network (OSNs), such as Twitter and Facebook, 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.

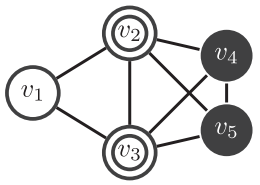
Among various methods, the graph-based sybil detection is the mainstream one, due to its computational efficiency and generality to detect sybils with different activity behavior. Typically, the graph-based sybil detection can be described as: (1) Model an OSN as a graph, in which nodes represent user 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], [8], [9], [10], [11], [12], [13], [14], [15], [16], just to name a few. However, recent works [17], [18], [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.

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 Fig. 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*

- *Xiaoying Zhang and John C.S. Lui are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: xyzhang@cse.cuhk.edu.hk, xiehong2018@cqu.edu.cn.*
- *Hong Xie and Pei Yi are with the Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400044, China. E-mail: hxie@cse.cuhk.edu.hk, pyi@cqu.edu.cn.*

Manuscript received 21 Aug. 2020; revised 20 Jan. 2022; accepted 31 Jan. 2022. Date of publication 16 Feb. 2022; date of current version 14 Mar. 2023.
The work of Hong Xie was supported in part by the National Natural Science Foundation of China under Grant 61902042. The work of John C.S. Lui was supported in part by the under Grant RIF 4032-18.
(Corresponding author: Hong Xie.)
Recommended for acceptance by K. Butler.
Digital Object Identifier no. 10.1109/TDSC.2022.3151701



(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.

v_1. Fig. 1b shows the historical activities of the users in Fig. 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 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.
- 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.
- 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.
- 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. Lastly, we present two extensions of Sybil_SAN to further improve its accuracy.

This paper organizes as follows. Section 2 presents the background and intuition of our design. Section 3 presents the social-activity graph model and the attack model. Section 4 presents the design of sybil detection algorithm on the social-activity graph. Section 5 presents the experimental results on synthetic data. Section 6 presents the experimental results on real world data. Section 7 presents two extension of the sybil detection algorithm. Section 8 presents the related work and Section 9 concludes.

2 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.

2.1 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], [8], [9], [10], [11], [12], [13], [14], [15], [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 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*.

2.2 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 3.

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 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 can not 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*.

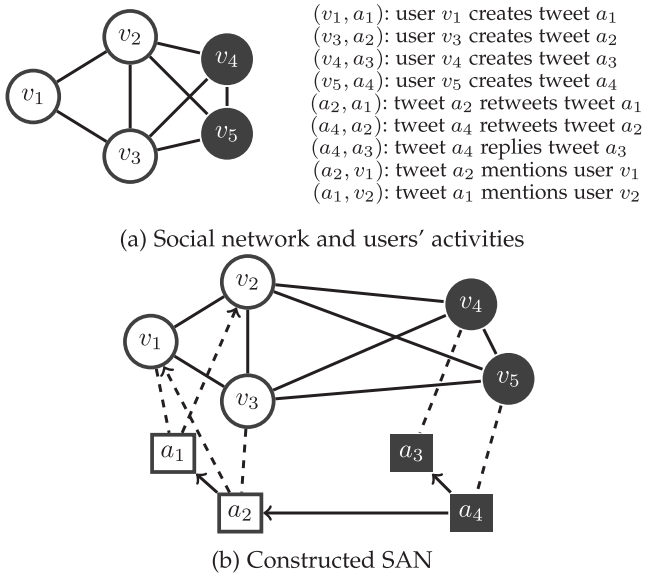


Fig. 2. An example in Twitter for constructing a SAN.

3 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.

3.1 The Social and Activity Network Model

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, Fig. 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, Fig. 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. Fig. 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 Fig. 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 Fig. 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.,

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.

(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 Fig. 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.*

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 Fig. 2b, the activity a_2 involves two interactions, i.e., (a_2, a_1) and (a_2, v_1).

3.2 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 Fig. 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 the 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 Fig. 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 2) of the previous graph-based sybil attack model [7], [8], [9], [10], [11], [12], [13], [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 Fig. 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 Fig. 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 Fig. 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, $\widetilde{\mathcal{G}}_h$ is the activity graph restricted to the sybil region only. In Fig. 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 Fig. 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 Fig. 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 α is, the less willing the honest nodes are in initiating interactions to sybil nodes. As we have discussed in Section 2 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 α 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 Fig. 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

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

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

Property 3. *The value of β can be arbitrarily large.*

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

3.3 Our Objective

Given $\mathcal{G}$, $\widetilde{\mathcal{G}}$ and a small set of labeled 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.

4 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.

4.1 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.,

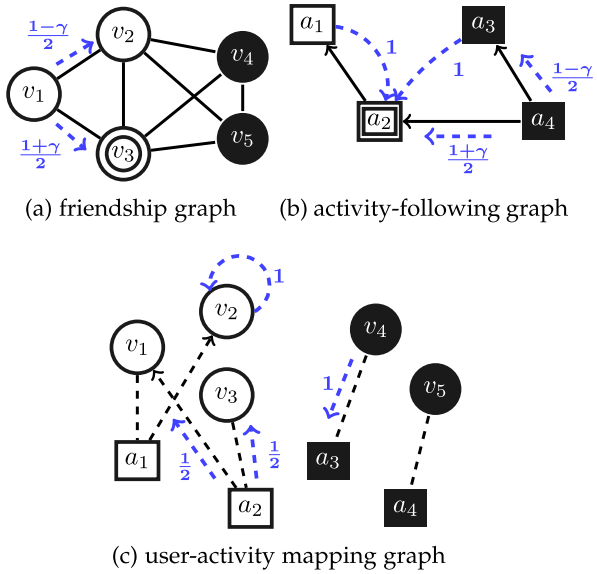
$$s_i = \begin{cases} 1/|\mathcal{S}|, & \text{if } v_i \in \mathcal{S}, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

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.



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

(c) user-activity mapping graph

Fig. 3. Decomposed random walks in toy example.

4.2 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, Fig. 3 presents a decomposition of the SAN network in Fig. 2b. Fig. 3 also presents the one-step transition probabilities, which correspond to the random walk strategies in each subgraph (presented in Algorithm 1).

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 nodes that the seed trusts more. With probability $1-\gamma$ the walker walks according to the naive random walk, and with probability γ the walker jumps to one of the seeds, where $\gamma\in[0,1]$. We present the detail of this walking strategy in Algorithm 1. Fig. 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 Fig. 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 Fig. 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 Fig. 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 Fig. 3c illustrates this case. We present the detail of this walking strategy in Algorithm 1.

4.3 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 supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2022.3151701, the coupled random walk converges to a unique landing probability distribution, or the trust scores converge.

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 λ_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 **untill** *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 ith column (or row) corresponds to user node v_i. For example, in Fig. 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 associated with the random walk on the activity-following graph, where the ith column (or row) corresponds to activity a_i. In Fig. 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 Fig. 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 supplementary file, available online, 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 5.

Solving the linear system is computationally expensive, thus we develop an iterative algorithm to calculate $\mathbf{s}^*$ (step 4 to 9 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., $\mathcal{N}_i=\{(u,v_i)|(u,v_i)\in\mathcal{E}$ 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 to the normalized trust in a descending order, and take users with low rank as suspects of sybils. In the following theorem we state the computational complexity of Algorithm 3.

Algorithm 3. Sybil_SAN

Input: $\mathcal{G}$, $\widetilde{\mathcal{G}}$, $\mathcal{S}$, $\mathbf{P}, \widetilde{\mathbf{P}}, \widehat{\mathbf{P}}, \epsilon$
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 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 $\mathbf{s}^{(\mathbf{t+1})}$

Lemma 1. *The computational complexity of Algorithm* $O(t_{\mathrm{Term}}(\epsilon)(|\mathcal{V}|^2 + |\mathcal{A}|^2))$, *where* $t_{\mathrm{Term}}(\epsilon)$ *denotes the round that Algorithm 3 terminates.*

Remark. Lemma 1 states that the computational complexity of Algorithm 1 is quadratic in the number of users and quadratic in the number activities. It implies that when the number of users and the number of activities are at the same order, the computational complexity of Algorithm 1 has the same order as that without activities, otherwise, the activities incurs a significant amount of extra computation. The extra computation incurred by activities is not a burden, because the core part of Algorithm 1 is a matrix multiply a vector, which can be parallelized.

4.4 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. *Due to page limit, we present theoretical analysis on the convergence and sensitivity analysis of Algorithm 2 and Algorithm 3 in our supplementary file, available online.*

5 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.

5.1 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 [25] 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.

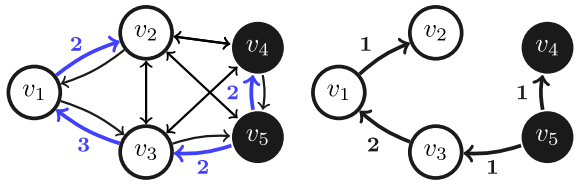
- *Sybil region:* Instead of focusing on one connected sybil region, here we consider a more practical scenario where sybils region is formed by several disconnected clusters, since in reality attackers at different company/country create their own sybil region and such regions may not always be connected 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 [26], 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 $\lfloor \frac{\beta W_s}{M} \rfloor$ 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. Following 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.



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

Fig. 4. An illustrative example of Twitter.

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 approach that extends SybilRank to deal with social-and-activity-based sybil attack model via adding weights to edges: (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, Fig. 4a shows the weighted friendship graph constructed from the social-and-activity network in Fig. 2. In Fig. 2, v_2 and v_1 are friends and v_2 doesn't initiate any interaction to v_1, thus in Fig. 4a, the edge $v_2 \rightarrow v_1$ 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 $v_3 \rightarrow v_1$ has a weight of 3 in Fig. 4a.
- *Inter:* This second approach that extends SybilRank to deal with social-and-activity-based sybil attack model via deleting edges: (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, Fig. 4b shows the strong friendship graph constructed on the social-and-activity graph in Fig. 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 Fig. 2, v_2 initiates no interaction to v_1, thus the edge $v_2 \rightarrow v_1$ is deleted in Fig. 4b. Furthermore, v_3 actively initiates $2 > 0$ interactions to v_1, thus the edge $v_3 \rightarrow v_1$ with a weight of 2 exists in Fig. 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 ≥ 1 interactions to each other.
- *SScar:* a state-of-art belief-propagation-based detection method [16].
- *SWalk:* a recent robust random-walk-based detection method [15].

Parameter Setting. 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 of *SR-U* and *SScar* are the same as the open source code,[2] which is published by the authors who propose *SR-U* and *SScar*. The parameters of *SWalk* is the same as that in the previous work [15]. For Sybil_SAN, we set the jumping constant $\gamma = 0.15$ in walks on the friendship graph and on the activity-following graph. For coupling the random walks, we set $n = 1, k = 0$, since accuracy of Sybil_SAN decreases as n, k increase. It meets our intuition, since neighbors are more trusty than users that are several-hops far. For user v_i who does not have any activities, i.e., $i \in \{j | \{a | (v_j, a) \in \mathcal{C}\} = \emptyset\}$, we set $\lambda_i = 1$. Namely, it will distribute trust only through friendship graph. For user v_i who has some activities, i.e., $i \in \{j | \{a | (v_j, a) \in \mathcal{C}\} \neq \emptyset\}$, we set $\lambda_i = 0.05 \times 0.9^{log_2(|N(v_i)|)}$ to further restrict the trust propagated to sybils. Here $N(v_i)$ is a set that contains all neighbors of v_i, and we choose smaller λ_i since the activities are more trusty than friendships. Meanwhile, the user with more friends should be more active, thus we let more trust being propagated through his activities. The $\lambda_i, \forall i > |\mathcal{V}|$ is special because it depends on the composition of users' interactions. In our synthetic dataset, the number of two types of interactions are roughly equal, hence we set $\lambda_i = 0.5, \forall i > |\mathcal{V}|$. The parameters (n, k, λ_i) stay fixed in each run of the simulations.

To reduce the bias caused by the seed nodes, we run the algorithms multiple times with different seeds and compute the average AUC. In each round, we select 50 honest seeds in all: 1 of them are chosen from the top 10 nodes with largest degree, the rest are randomly chosen from remaining honest nodes. And we also randomly select 10 sybil seeds. We need to remark that above selected strategy follows previous works [7], [15], [16]. And in each run, all the methods use the same selected seed set.

5.2 Impact of Friendship Attack Edges N_A

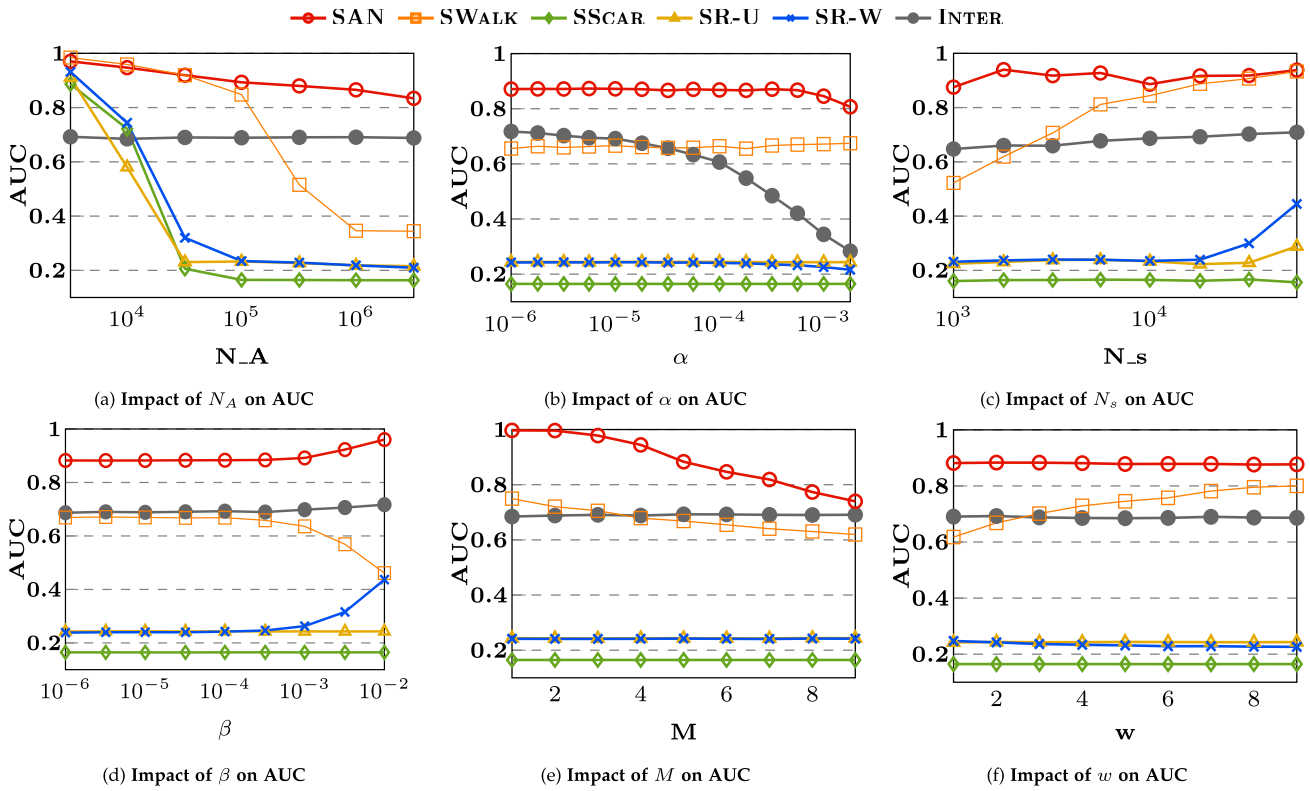
We consider the default setting as described in Section 5.1, except that we vary the number of friendship attack edges (N_A). Fig. 5a shows the AUC for six algorithms as N_A varies. One can observe that Sybil_SAN has the highest AUC. Namely, it has the highest accuracy. As we increase N_A, the AUC of 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 3×10^6 (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.

Summary of Observations.Sybil_SAN has the highest accuracy, and is robust against a large number of friendship attack edges.

5.3 Impact of Incoming Interaction Attack α

We consider the default setting described in Section 5.1, except that we vary the α 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

2. https://github.com/binghuiwang/sybildetection/



(a) Impact of N_A on AUC (b) Impact of α on AUC (c) Impact of N_s on AUC

(d) Impact of β on AUC (e) Impact of M on AUC (f) Impact of w on AUC

Fig. 5. Experiments on synthetic datasets.

Sybil_SAN in a more general scenario. Remark that a larger α means that a larger number of incoming interaction attack edges exist. Fig. 5b shows as α increases, Sybil_SAN always outperforms the other algorithms and its AUC drops slightly as α 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 α increase. The AUC for the SR-U and SScar algorithm is flat, because the α does not influence the underlying friendship graph. SWalk is not sensitive to α, 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, SWalk); (3) Sybil_SAN works well to combine the users' friendships and their activities.

Summary of Observations. Sybil_SAN has the highest accuracy, and is robust against incoming interaction attacks.

5.4 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 Fig. 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.

Summary of Observations. Under different number of sybils, our Sybil_SAN algorithm always has the highest accuracy.

5.5 Impact of Outgoing Interaction Attack β

We vary the value of β to see the impacts of outgoing interaction attack. Note that larger β suggests that sybils initiate more interactions to honest users. From Fig. 5d, one can observe that Sybil_SAN has the highest AUC, i.e., it outperforms other algorithms under different β. Furthermore, the AUC for Sybil_SAN, SR-W and Inter increase in β. 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.

Summary of Observations. 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.

5.6 Impact of Structure of Sybil Region

Fig. 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.

5.7 Impact of Interactions Among Sybils

We also investigate the effect of number of interactions among sybils (w). From Fig. 5f, we can see that all algorithms stay stable under different w except SWalk. This is because sybils in a denser sybil region will get to the extra sybil node more quickly, thus will be more easily to be detected out. Meanwhile, we can observe that Sybil_SAN still outperforms other algorithms.

6 Experiments On Real Dataset

In this section, 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%.

6.1 Experimental Setting

*Datasets.*We have 991 public fake accounts [27] in Twitter. Based on these 991 fake accounts we extracted a friendship graph and an activity graph from Twitter in three steps:

- First, starting from 991 public fake accounts [27], we performed a 2-level BFS (Breadth First Search) to crawl all the followers of above 991 fake accounts as well as the followers of the crawled followers. We do the 2-level BFS, because crawling friendships in Twitter is quite limited (15 requests/per 15 minutes). Through this we obtain a network centered around these 991 sybils.
- Second, we used a public Twitter network dataset [28] to enlarge the above network. We discarded the crawled users who are not in the Twitter network dataset in [28], leaving 6644 crawled users. We extracted all the friendships of these 6644 crawled users from the Twitter network dataset [28]. By combining the three sources of nodes (991 known sybils, 6644 our crawled users and their friends in Twitter network [28]) together, we obtain all the user nodes. We removed those nodes with degree <3, since their friendship structure may not be well captured by this subnetwork.
- We developed a crawler to visit the profile of each user except these known sybils using Twitter API, and extract the status (i.e., active, suspended, deleted) of each user. And we took suspended users as sybils, active users as honest users, and removed deleted users. Then we transformed this network to be an undirected one via keeping an edge between two nodes only if they follow each other, and extracted the largest connected component. Finally, we extracted the activities of all the users in network. For each user, we extract the recent 3,000 activities.

In summary, we get a network of 450,242 users and 222,944,310 links, which contains 409,694 honest users, 40,548 sybils and 17,581,069 attack edges. And totally there are 102,693,769 activities among user, which contains 714,392 incoming interaction attacks.

Seed Selection. Since the trust flows from seeds to others, thus it can contribute to more accurate sybil detection to select seeds from nodes that can reach many users. for example, of the three honest nodes in Fig. 4b, v_3 is the best candidate to be taken as a seed. We adopted a previous seed selection method, which has been used in [22], [29]. Formally, we first process our dataset to obtain strong friendship graph following the method described in Section 5.1 Then, we assign each node $\frac{1}{|V|}$ bit credits, then apply the following credit diffusion process on the strong friendship graph:

$$c_i^{t+1} = \sum_{v_i \to v_j} \frac{c_j^t}{indeg(j)}, \tag{2}$$

where c_i^t denote the credit of node v_i at tth iteration, and $indeg(j)$ denotes the in-degree of node v_j, i.e., the number of nodes that point to v_j. We perform this iteration until it converges. Nodes with more outlinks will get higher credits. In other words, the credit of each user naturally reflects his ability to reach other users. Then starting from the user with highest credits, we inspect user one by one, until we obtain 50 verified honest users. These verified honest users are taken as seeds.

For sybil seeds, we randomly select 10 sybil nodes from the public 991 sybils as sybil seeds.

Parameter Settings of Our Sybil-SAN Algorithm. For n, k, γ, and $\lambda_i, \forall i \leq |\mathcal{V}|$, we use the same settings as in synthetic datasets. In real dataset, 70,074,436 of 102,693,769 (around 70%) interactions are links in $\mathcal{E}_m$, i.e., mentioning activities, thus we set $\lambda_i = 0.3, \forall i > |\mathcal{V}|$. Note that in setting the above parameters, we do not need any a-priori knowledge on sybils.

TABLE 1
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%**	–

6.2 Results and Implications

We run six algorithms described in Section 5 on our real dataset. Table 1 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 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 that the detection accuracy in real dataset is not as good as the one in synthetic datasets.

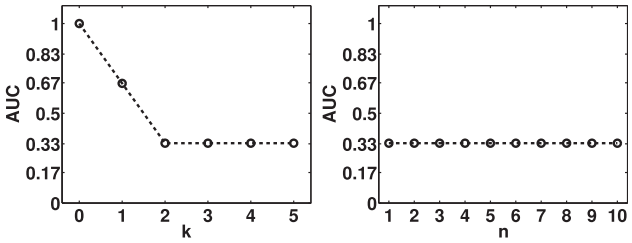
Summary of Observations. Our Sybil_SAN has a high accuracy in the real world dataset, and improves the AUC over the state-of-art algorithms by at least 17.7%.

7 Extensions

To understand potential limitations of the Sybil_SAN algorithm, we evaluate it on a carefully designed small social-activity network. Two potential limitations of the Sybil_SAN algorithm are revealed, and we give fundamental understandings on these limitations. Finally, we present two extensions of the Sybil_SAN algorithm and use numerical results to show that they can relieve the limitation of the Sybil_SAN algorithm significantly.

7.1 Limitations of Sybil_SAN

To gain fundamental understandings on the limitations of the Sybil_SAN algorithm, we evaluate the AUC of Sybil_SAN over the simple social-activity graph shown in Fig. 2b.



(a) Impact of k ($n = 1$) (b) Impact of n ($k = 5$)

Fig. 6. Evaluating the AUC of Sybil_SAN on Fig. 2b.

Unless we vary them explicitly, the parameters of Sybil_SAN is the same as that in Section 5. Fig. 6a shows the impact of k (recall that the number of random walks on user-activity graph is $2k+1$) on the AUC of Sybil_SAN. One can observe that as k increases from 0 to 2, the AUC of Sybil_SAN decreases significantly, and then the AUC becomes flat when k increases from 2 to 5. It is important to note that when k is larger than or equal to 2, the AUC is recall 0.33. Recall that larger AUC implies a higher accuracy of detecting the sybil and randomly sorting all the nodes can achieve an AUC of 0.5. Namely, the accuracy of Sybil_SAN drops significantly when allowing more walks on the user-activity graph and the accuracy can be even lower than randomly sorting the nodes. It reveals one limitation of the Sybil_SAN that its accuracy is sensitive to the number of walks on the user-activity graph, making it difficult to tune the parameter k and utilize the user-activity graph. One interesting question is: *is it possible to utilize the walks on the activity-following graph to relieve this limitation?* To answer this question, Fig. 6b shows the impact of n (recall that the number of walks on the activity-following graph is n) on the AUC of Sybil_SAN. We consider the case of $k = 5$, where the AUC of Sybil_SAN is quite low, i.e., 0.33. From Fig. 6b one can observe that as n increases from 1 to 10, the AUC of Sybil_SAN is flat. Namely, when the accuracy of Sybil_SAN is low, tuning the number of walks n on the activity-following graph may not increase the accuracy of Sybil_SAN.

To gain insights on the above limitations of Sybil_SAN, we present part of the random walk transition probability over the user-activity graph in Fig. 7. From Fig. 7, one can observe that node v_2 does not create any activity. When the walker over the user-activity graph jumps to node v_2, it will stay at node v_2 permanently. Furthermore, increasing the number of walks (i.e., k) on the user-activity graph would increase the chance that the walker over the user-activity graph moves to node v_2. Namely, a large amount of trust score from v_1 and v_3

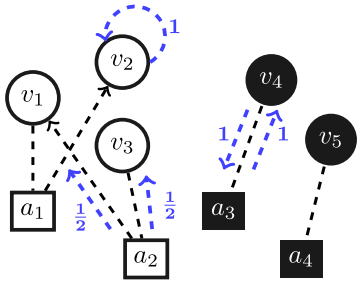


Fig. 7. The limitation of the SAN algorithm.

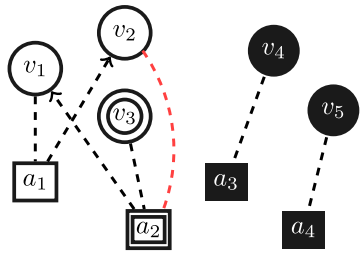


Fig. 8. Illustrating virtual user to activity edges.

will be propagated to node v_2. Note from Fig. 2b that v_2 has connections to sybil node v_4 and v_5. Thus, eventually a large amount of trust score will be propagated to v_4 and v_5 through v_2, leading to low accuracy of Sybil_SAN. Furthermore, Fig. 7 shows that the trust score does not propagates from the activity of sybil nodes to honest nodes, as once the walker over the user-activity graph starts from the sybil region (i.e., sybil nodes and activities of sybil nodes), it is impossible for it to move to the honest region (i.e., honest nodes and activities of honest nodes). We next propose extensions of Sybil_SAN to overcome the above limitations.

7.2 Extensions to Overcome the Limitation

To overcome the above limitations of Sybil_SAN, we propose two natural extensions of Sybil_SAN, resulting in Sybil_SAN_VUA and Sybil_SAN_VAU respectively.

Sybil_SAN_VUA. Sybil_SAN with virtual user to activity edge. Recall that the walker over the user-activity graph gets trapped by node v_2, due to that node v_2 does not create any activity. To relieve the limitation that v_2 traps the walker, we add some virtual edges between the nodes that do not create any activities to the activities created by trusted seeds. More precisely, for each node that does not create any activity, we add an undirected edges between it and the activities created by trusted seeds. Fig. 8 shows an example of adding such virtual edges. After adding such virtual edges, when the walker over the user-activity graph moves to node v_2, it will not gets trapped by node v_2 as Sybil_SAN, but instead it will moves to a_2. Through this trust scores will be propagated to trusted activities.

Based on the above idea of adding virtual edges, Algorithm 4 outlines an extension of Sybil_SAN to relieve the limitation of Sybil_SAN. Formally, Algorithm 4 first adds virtual edges as above. The added virtual edges expands the edge set $\mathcal{C}$, which indicates the creation relationship among users and activities. Then, it runs Sybil_SAN on this extended graph to compute the trust score for each node.

Algorithm 4. Sybil_SAN_VUA (Sybil_SAN With Virtual User to Activity Edge)

Input: $\mathcal{G}, \widetilde{\mathcal{G}}$
for $v \in \mathcal{V}$ **do**
  if $\{a|(v,a) \in \mathcal{C}\} = \emptyset$**do**
    for $a \in \{a|(s,a) \in \mathcal{C}, s \in \mathcal{S}\}$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(v,a)\}$
Update graph: $\widetilde{\mathcal{G}} \leftarrow (\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \mathcal{F})$
Run Sybil_SAN, i.e., Algorithm 3, on $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ to compute trust score vector $\mathbf{q}$;

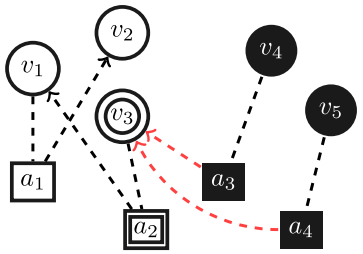


Fig. 9. Illustrating virtual activity to user edges.

Sybil_SAN_VAU. Sybil_SAN with virtual activity to user edge. Recall that as shown in Fig. 7, one potential limitation of Sybil_SAN is that the trust score may not propagate from the activity of sybil nodes to honest nodes. To relieve this limitation, we add virtual edges from activities that do not mention any user node to trusted seed user nodes. Fig. 9 shows an example of adding such virtual edges. Through this the trust score will propagate from the activities of sybil nodes to trusted seeds.

Based on the above idea of adding virtual edges, Algorithm 5 outlines an extension of Sybil_SAN to relieve the limitation of Sybil_SAN. Formally, Algorithm 5 first adds virtual edges as above. The added virtual edges expands the edge set $\mathcal{M}$, which indicates the mentioning relationship among users and activities. Then it runs Sybil_SAN on this extended graph to compute the trust score for each node.

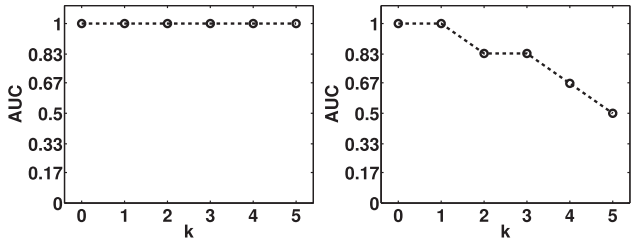
Algorithm 5. Sybil_SAN_VAU (Sybil_SAN With Virtual Activity to User Edge)

Input: $\mathcal{G}, \widetilde{\mathcal{G}}$
for $a \in \mathcal{A}$ **do**
  if $\{v|(a,v) \in \mathcal{M}\} = \emptyset$**then**
    for $s \in \mathcal{S}$ **do**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{(a,v)\}$
Update graph: $\widetilde{\mathcal{G}} \leftarrow (\mathcal{V}, \mathcal{A}, \mathcal{C}, \mathcal{M}, \mathcal{F})$
Run Sybil_SAN, i.e., Algorithm 3, on $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ to compute $\mathbf{q}$;

7.3 Evaluation

To reveal fundamental understandings on the limitations of the Sybil_SAN algorithm as well as when and how our Sybil_SAN_VUA and Sybil_SAN_VAU are effective in mitigate these limitations, we conduct experiments on carefully synthesized data. It is difficult to reveal such fundamental understandings on a real-world data. The reason is that we need to test these algorithms on all possible cases systematically, but it is really difficult for one to collect real-world data covering all these cases. In particular, we first evaluate Sybil_SAN_VUA and Sybil_SAN_VAU on a small graph, i.e., Fig. 2b, where we obtain explainable experiment results. Then we evaluate them on a larger graph, showing that findings on the small graph can be generalized to large graphs.

Evaluation on a Small Graph. Fig. 10 shows the AUC of both Sybil_SAN_VUA and Sybil_SAN_VAU over the social-activity graph presented in Fig. 2b. From Fig. 10a, one can observe that as k increases from 0 to 5, the AUC curve of Sybil_SAN_VUA is flat. Furthermore, the AUC is 1. Recall that the maximum possible value of AUC is 1. Namely, Sybil_SAN_VUA has a higher accuracy and it is



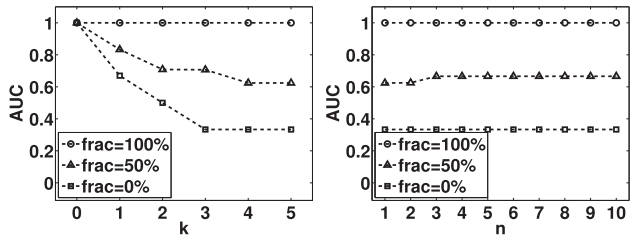
(a) Sybil_SAN_VUA (b) Sybil_SAN_VAU

Fig. 10. Evaluating the AUC of Sybil_SAN_VUA and Sybil_SAN_VAU on Fig. 2b with $n = 1$.

stable against k. Fig. 10b shows that the AUC of Sybil_SAN_VAU decreases as k increases from 0 to 5. Compared to Fig. 6a, one can observe that the AUC of Sybil_SAN_VAU is significantly higher than that of Sybil_SAN, implying the efficiency of Sybil_SAN_VAU.

Evaluation on Large Graphs. We first generate one hundred copies of Fig. 2b and put these copies in a line. Then, for each adjacent two copies, we add an edge between their sybil nodes, i.e., v_4 of each copy, and add an edge between their seed nodes, i.e., node v_3 of each copy. A copy is called extended copy, if one activity node is added, which is created by node v_2 and mentions node v_3 of the corresponding copy. This extension of graph is motivated by that one limitation of Sybil_SAN is caused by that v_2 node does not create any activities. We will vary the fraction of such extended copy from 0% to 100% to study their impact on the AUC of Sybil_SAN. Through the study, we aim to answer the following two questions:

To reveal fundamental understandings on the limitations of the Sybil_SAN algorithm, Fig. 11 shows the AUC of Sybil_SAN under different fraction of extended copies. Fig. 11a shows that when the fraction of extended copies is 0%, i.e., there are no extended copies, the AUC of Sybil_SAN decreases sharply when k increases. Furthermore, the AUC of Sybil_SAN is around 0.3, i.e., a very small AUC, when k exceeds 3. In other words, Sybil_SAN has a limitation and sensitive to the parameter k when the fraction of extended copies is 0%. When the fraction of extended copies is 100%, i.e., all copies are extended, the AUC of Sybil_SAN is almost fixed at around 1 when k increases from 0 to 5. In other words, Sybil_SAN works well and robust against the parameter k when the fraction of extended copies is 100%. When the fraction extended copies increases from 0% to 100%, the AUC curve of Sybil_SAN moves from bottom to top. In other words, Sybil_SAN gets more robust to the



(a) Impact of k (n=1) (b) Impact of n (k=5)

Fig. 11. Impact of the fraction of extended copies on the AUC of Sybil_SAN.

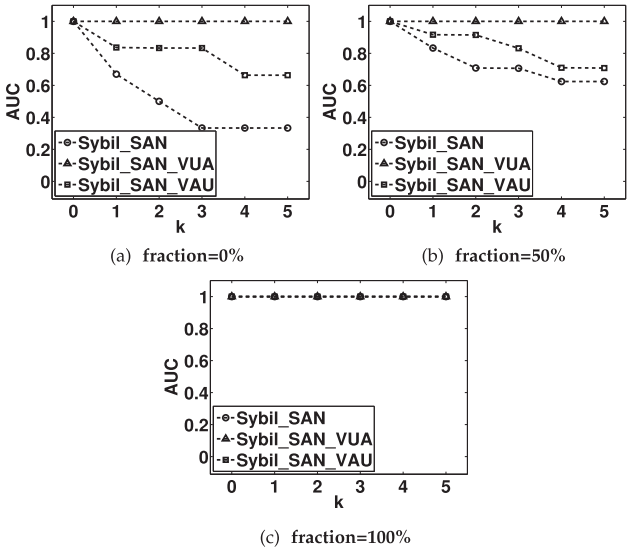


Fig. 12. Impact of the fraction of extended copies on the AUC of Sybil_SAN_VUA and Sybil_SAN_VAU.

parameter k becomes less limited in detecting sybils when the fraction extended copies increases from 0% to 100%. Fig. 11b shows that when the faction of extended copies is 0%, the AUC of Sybil_SAN is nearly the same as n increases from 1 to 10. This statement also holds when the faction of extended copies is 50% or 100%. This results shows that the findings in Fig. 11a is reliable.

To reveal fundamental understandings on when and how our Sybil_SAN_VUA and Sybil_SAN_VAU are effective in mitigate the limitations of Sybil_SAN, Fig. 12 shows the AUC of Sybil_SAN_VUA and Sybil_SAN_VAU under different fraction of extended copies. Fig. 12a shows that when the fraction of extended copies is 0%, the AUC curve of Sybil_SAN_VUA lie in the top while that of Sybil_SAN lies in the bottom. The AUC curve of Sybil_SAN_VAU lies between that of Sybil_SAN and Sybil_SAN_VUA. Furthermore, the AUCs of both Sybil_SAN_VUA and Sybil_SAN_VUA are significantly larger than that of Sybil_SAN. This shows that Sybil_SAN_VUA and Sybil_SAN_VUA are high effective in address the limitation of Sybil_SAN. When the fraction of extended copies increases from 0% to 100%, the AUC improvements by both Sybil_SAN_VUA and Sybil_SAN_VUA decreases. This implies that Sybil_SAN_VUA and Sybil_SAN_VUA are effective in improve the AUC of Sybil_SAN when the faction of extended copies is small.

7.4 Summary

Two extensions of Sybil_SAN are proposed to relieve the limitations of Sybil_SAN algorithm. We also show the efficiency of each extension in terms of improving the accuracy of sybil detection. One may also combine these two extensions to achieve further improvement.

8 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 [30], activity patterns [31], users' profiles and tweets [32], click-streams[33], etc. Supervised detection usually uses features of labeled users (e.g., users' profiles, fraction of accepted requests, etc.) to train classifiers to identify sybils [17], [34]. 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. Xia *et al.* [35] utilized the patterns friendship requests, which is mined from a dataset with over one million labeled data from WLink, to detect malicious accounts via machine learning.

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], [8], [9], [10], [11], [12], [13], [14], [15], [16], just to name a few. However, recent works [17], [18], [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. Gao *et al.* [36] proposed Sybilfuse, which contains a local phase and global phase. In the local phase, Sybilfuse compute local trust scores for nodes and edges via training local classifiers. In the global phase, Sybilfuse uses via weighted random walk and loopy belief propagation to propagate local trust scores over the global network. Algorithms in [37], [38] 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.* [39] and Xue *et al.* [40] used acceptance and rejection of friend requests to enhance sybil detection. Yang *et al.* [41] utilized interactions among users to detect sybils. More specifically, they formulate a signed graph to model friend requests of users, where a directed edge with weight -1 indicates that a user rejects a request and weight 1 indicates that a users accepts a request. Based on this interaction graph, they detect sybils via graph embedding. Yuan *et al* [42] utilized registration patterns mined from a dataset from WeChat (the largest online social network in China) to detect sybil accounts. Liang *et al* [43] identified the cluster pattern of fake accounts in a dataset from WeChat, which is helpful for fake account 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.

9 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. Experimental results further verify that our Sybil_SAN is highly robust against sybils' attacks.

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 *Proc. IEEE Symp. Secur. Privacy*, 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 *Proc. 18th Int. Conf. World Wide Web*, 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 *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 941–950.

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

[5] Sybils in Twitter, Nov. 2013, [Online]. Available: https://www.nbcnews.com/business/1–10-twitter-accounts-fake-say-researchers-2D11655362

[6] Sybils in Facebook, Aug. 2012. [Online]. Available: https://www.nbcbayarea.com/blogs/press-here/Facebook-83-Million-Fake-Users-164848046.html

[7] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, p. 15.

[8] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2009, pp. 1–15.

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

[10] N. Tran, J. Li, L. Subramanian, and S. S. Chow, "Optimal sybil-resilient node admission control," in *Proc. IEEE Conf. Comput. Commun.*, 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 *Proc. IEEE Conf. Comput. Commun.*, 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 *Proc. IEEE Symp. Secur. Privacy*, 2008, pp. 3–17.

[13] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: Defending against sybil attacks via social networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 267–278, 2006.

[14] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, "SoK: The evolution of sybil defense via social networks," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 382–396.

[15] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 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 *Proc. IEEE Conf. Comput. Commun.*, 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 Trans. Knowl. Discov. Data*, vol. 8, no. 1, 2014, Art. no. 2.

[18] V. Sridharan, V. Shankar, and M. Gupta, "Twitter games: How successful spammers pick targets," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 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 *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, 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 *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops*, 2017, pp. 157–162.

[21] S. Effendy and R. H. Yap, "The strong link graph for enhancing sybil defenses," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 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 Trans. Netw.*, vol. 24, no. 5, pp. 2834–2846, Oct. 2016.

[23] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 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 *Proc. 7th IEEE Int. Conf. Data Mining*, 2007, pp. 739–744.

[25] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality prediction and community structure in social networks," *Sci. Rep.*, vol. 3, 2013, Art. no. 2522.

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

[27] 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 *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 963–972.

[28] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, A social network or a news media?," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.

[29] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 576–587.

[30] Q. Cao, X. Yang, J. Yu, and C. Palow, "Uncovering large groups of active malicious accounts in online social networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 477–488.

[31] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "DNA-inspired online behavioral modeling and its application to spambot detection," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 58–64, Sep./Oct. 2016.

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

[33] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proc. 22nd USENIX Conf. Secur.*, 2013, pp. 241–256.

[34] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, 2010, pp. 1–9.

[35] Z. Xia, C. Liu, N. Z. Gong, Q. Li, Y. Cui, and D. Song, "Characterizing and detecting malicious accounts in privacy-centric mobile social networks: A case study," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2012–2022.

[36] P. Gao, B. Wang, N. Z. Gong, S. R. Kulkarni, K. Thomas, and P. Mittal, "SybilFuse: Combining local attributes with global structure to perform robust sybil detection," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2018, pp. 1–9.

[37] Y. Boshmaf *et al.*, "Integro: Leveraging victim prediction for robust fake account detection in OSNs." in *Proc. Network Distrib. Syst. Secur. Symp.*, vol. 15, 2015, pp. 8–11.

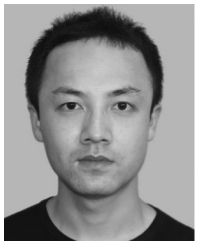
[38] J. Höner, S. Nakajima, A. Bauer, K.-R. Müller, and N. Görnitz, "Minimizing trust leaks for robust sybil detection," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1520–1528.

[39] Q. Cao, M. Sirivianos, X. Yang, and K. Munagala, "Combating friend spam using social rejections," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 235–244.

[40] 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 *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 2400–2408.

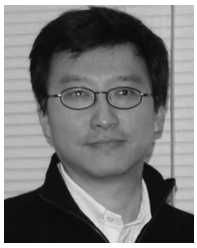
[41] Z. Yang, Y. Zhang, and Y. Dai, "Defending against social network sybils with interaction graph embedding," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2018, pp. 1–9.
[42] D. Yuan *et al.*, "Detecting fake accounts in online social networks at the time of registrations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1423–1438.
[43] X. Liang *et al.*, "Unveiling fake accounts at the time of registration: An unsupervised approach," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 3240–3250.

Xiaoying Zhang received the BEng degree from the School of Computer Science and Technology, The University of Science and Technology of China, and the PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, under the supervision of Prof. John C.S. Lui. She is currently a researcher with Bytedance. Her research interests include data science and recommender systems.



Hong Xie received the BEng degree from the School of Computer Science and Technology, The University of Science and Technology of China in 2010, and the PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong in 2015, proudly under the supervision of Prof. John C.S. Lui. He is currently a research professor with the College of Computer Science, Chongqing University. He was a postdoctoral research fellow with The Chinese University of Hong Kong and National University of Singapore.

Pei Yi received the BEng degree from Hohai University, China. He is currently working toward the master's degree with the College of Computer Science, Chongqing University, under the supervision of Prof. Hong Xie. His research interests include graph analytics and Markov Chain Monte Carlo.



John C.S. Lui received the PhD degree in computer science from the University of California, Los Angeles. From 2005 to 2011, he was a chairman with CSE Department. He is currently the chohming li chair professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include communication networks, system security (such as, cloud security and mobile security), network economics, network sciences, large-scale distributed systems, and performance evaluation theory. He is an elected member of IFIP WG 7.3 and a croucher senior research fellow. He was the recipient of various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He was also the co-recipient of IFIP WG 7.3 Performance 2005 and IEEEIFIP NOMS 2006 best student paper awards. He is in the editorial board of the *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Computers*, and *IEEE transactions on Parallel and Distributed Systems*.