SybilExposer: An Effective Scheme to Detect Sybil Communities in Online Social Networks

Satyajayant Misra, Abu Saleh Md Tayeen, Wen Xu†

***Abstract*—The popularity of online social networks (OSNs) has resulted in them being targeted with Sybil attacks, where an adversary forges many fake identities (called Sybils) to disrupt or control the normal functioning of the system. Several schemes have been proposed to defend against Sybil attacks. Most of these schemes work by computing the landing probability or statistical distribution of visiting frequency of random walks. These schemes usually have high running time cost and are highly dependent upon the proper choice of known trusted nodes. To address these limitations, in this paper we present SybilExposer, an efficient and effective Sybil community detection algorithm, which relies on the properties of social graph communities to rank communities according to their perceived likelihood of being fake or Sybil. Our experiments on several real-world OSN graphs illustrate that SybilExposer has close to 100% true positive rate and nearly zero false positive rate in identifying Sybil communities, and the best running time complexity compared to the state of the art.**

Keywords: **Sybil attack, online social network, OSN security.**

1. Introduction

With millions of daily active users, online social networks (OSNs), such as Facebook, Google+, Twitter, Orkut, and Pokec have become an integral part of our daily life. Facebook, the biggest OSN worldwide had 1.2 billion monthly active users, as of December 2013 [1]. These OSNs offer membership to any users with email addresses and some basic personal information. Their popularity and open membership nature, make these networks candidates for several malicious attacks.

Sybil attack is one of such attacks on these systems. In Sybil attacks, an attacker can forge multiple identities called Sybils (or Sybil nodes) to adversely influence the functioning of the targeted system. For instance, adversary-owned identities can outvote the benign users in reputation or voting systems of OSNs. Socialbot-controlled [5] user accounts can help attackers acquire sensitive user data and distribute customized spams in the messaging system of OSNs. It is estimated that for Facebook, approximately 1.2% of its worldwide monthly active users (14 million) are fake accounts [1]. Therefore, Sybil attack is a serious threat in OSNs.

One main goal of defense against Sybil attacks is to detect and remove Sybil identities. Recently a number of schemes have been proposed to uncover Sybils by leveraging OSN graphs, in which a node represents a user and the edges between nodes correspond to the trust relationship established by the users. The basic foundation of these schemes is based on the inherent difficulty of a Sybil node connecting with honest nodes. Even when an attacker can create multiple Sybil identities in social networks and connect them arbitrarily, because of the trust relationship that must exist between two identities in the real-world to be connected, an attacker will have difficulty establishing many connections with honest identities. As a result, Sybil nodes are poorly connected to the rest of the network, which causes the social graph to partition into two regions: one comprised of Sybil nodes and the other of honest nodes. Most of the Sybil defense schemes leverage the sparse cut-set consisting of all *attack edges* (links between Sybil nodes and honest nodes) to identify Sybil nodes. However, [8] confirmed that this over-simplified social structure is not accurate in real-world social networks. The honest region actually appears as a collection of tightly knit local communities relatively loosely coupled with one another rather than one large community.

Moreover, researchers have observed [15] that the success rate of creating links to honest nodes through sending out link-establishment requests from attacker-controlled forged profiles and engineering bots [5] are more frequent than believed. Hence the cardinality of the cut-set connecting the Sybil region/community to the honest regions/communities is not necessarily that small. This makes state of the art Sybil detection schemes, which rely on this small cutset cardinality to identify Sybil nodes ineffective. Furthermore, instead of one region composed of Sybil nodes, there may exist multiple Sybil communities which are connected to the honest nodes, thus leaving the network vulnerable to the attackers who are controlling those Sybil communities, which is so far unexplored.

Most of the recent Sybil defense mechanisms ([6], [12] and [14]) use a number of random walks of different length from known benign users or trusted nodes to distinguish between Sybil nodes and honest nodes. Performing random walks for large datasets with million to billion nodes, is a time-consuming and computationally expensive operation. Besides, the quality and performance of these schemes significantly depend on the selection of the *known* trusted nodes in social networks, which may require user monitoring to identify Sybil nodes. Motivated by the above issues, in this paper we propose *SybilExposer*, which addresses the limitations of the previous works. SybilExposer identifies Sybil nodes in attack scenarios where Sybil nodes form one or more communities and are strongly integrated with the honest communities.

Contributions: The main contributions of this paper include: **a)** We propose SybilExposer, a computationally efficient algorithm to detect Sybil communities. **b)** SybilExposer does not require the initial manual identification of trusted nodes, which is non-scalable for large OSNs. **c)** We perform comprehensive evaluation of SybilExposer along with other state of the art algorithms (SybilRank, SybilDefender and SybilShield) on several real-world OSN datasets.

The rest of this paper is organized as follows. Section 2 provides the background and the related work on Sybil defense schemes. Section 3 presents our model and assumptions. We describe SybilExposer in Section 4. Section 5 shows the effec-

† The authors are with the Computer Science department of New Mexico State University, Las Cruces, NM. Email:{misra, atayeen, wxu}@cs.nmsu.edu. This work was supported in part by the the U.S. NSF grants: 1248109 and 1345232 and the U.S. ARO grant number W911NF-15-1-0393.

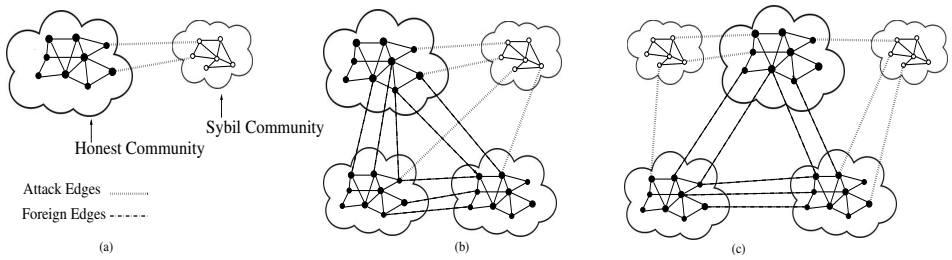


Fig. 1: Threat Model: (a) Single honest and single Sybil community, (b) Multiple honest and single Sybil communities, (c) Multiple honest and multiple Sybil communities.

tiveness of SybilExposer using experiments, which is followed by concluding remarks in Section 6.

2. Background and Related Work

Sybil detection approaches in OSNs can be classified into two categories: feature-based approach and social-graph-based approach. In feature-based Sybil detection approaches, researchers have focused on extracting features (acceptance rate of incoming/outgoing requests, invitation frequency, etc.) from users' profiles [15] and observing user activity patterns (Clickstream data) [13] to derive a machine-learning based model to identify fake/Sybil user accounts. In social-graph-based approaches, researchers have exploited the topological properties of OSN graphs such as fast-mixing, and conductance to identify Sybil nodes. Our work studies Sybil community detection in the absence of user profile and interaction data, and falls into the second category leveraging structural properties of social graphs. Yu *et al.* proposed SybilGuard [17] followed by SybilLimit [16]. Both algorithms rely on the strong assumption of simplified social network structure and fast mixing property [17]. However, as observed in [11] many OSNs are not that strongly fast mixing, which reduces the effectiveness of SybilGuard and SybilLimit.

SybilShield [12] accounts for the multi-community structure of social networks. It employed multiple agents (verifiers) to verify potential Sybil nodes. It demonstrated significantly less false positives than SybilGuard, but without any improvement in false negatives. Also, in SybilShield there are several parameters whose choice is tightly tied to the properties of social graphs. SybilInfer [7], a centralized approach, uses a Bayesian inference technique that assigned to each node of the network its probability of being Sybil. It has high false negatives and high computation overhead. In [3], the authors have identified that the ACL (Anderson, Chung, Lang) [3] algorithm can be used to white-list a local region in the graph, but there are no explanations on how multiple Sybil communities can be identified system-wide.

SybilDefender [14] calculates the statistical distribution of the visiting frequency for nodes in the honest region using a fixed number of short random walks originated from multiple known trusted nodes. A node whose statistical distribution differs from that of the calculated distribution is identified as sybil. SybilDefender does not scale for large OSNs due to the large overhead of computing several random walks. It also has parameters that vary significantly for different graphs. SybilRank [6], the best algorithm in literature, aims to derive a ranking of all the nodes in a social graph where Sybil nodes will most likely have the low rankings. However, SybilRank uses $\log n$ power iterations [10], where n is the number of nodes in the social graph. These iterations result in high computational cost for large graphs. SybilRank also requires the selection of several trusted seeds by manual inspection of nodes in the social graph, thus is non-scalable.

SybilExposer, our Sybil defense algorithm addresses the weaknesses—high computation cost, need for manual inspection, do not consider multiple sybil communities—of the existing social graph based Sybil defense schemes. It also improves identification, it has more than 20% improvement in Sybil communities detection compared to SybilRank. SybilExposer can serve as a scalable first line of identification of multiple Sybil communities in large networks with the more data-intensive feature-based approaches following it.

3. System and Threat Model and Assumptions

System and Threat Model: The OSN is modeled as an undirected graph $G = (V, E)$, where V represents a set of unique users and E represents dyadic social relationships of the users. The social graph G has order $n = |V|$ and the size $m = |E|$. We denote the degree of a node $v \in V$ by $deg(v)$. We define two different types of degree of a node v: *inter-community degree* and *intra-community degree*. *Intra-community degree*, **intra-deg**(v) of $v \in V$, refers to the number of nodes adjacent to v that also belong to the same community as v. *Inter-community degree*, **inter-deg**(v) of $v \in V$, refers to the number of nodes adjacent to v but not in the same community as v. The honest nodes may form a single cluster or multiple clusters of different sizes, which are termed as honest communities/regions. These communities may be interconnected; an edge between two honest nodes in two different honest communities is named as a *foreign edge*.

In our threat model, there can be one or more malicious users in the network, each capable of launching Sybil attack by creating a number of unique but fake identities called Sybil identities. Communities formed by Sybil nodes are termed Sybil communities/regions. The adversaries can create arbitrary edges within the Sybil communities. Sybil regions are connected to one or more honest communities by attack edges emanating from their corresponding nodes. Fig. 1 shows a social network topology with our threat model.

Assumptions: *To our best knowledge, we are the first to account for multiple Sybil communities in an OSN, interacting with multiple honest communities*. We assume that the online social network provider, e.g., Facebook has access to the entire social graph. Since an edge in social network represents a real-world dyadic human relationship, it is difficult for a Sybil node to directly connect to many honest nodes. Consequently, we

assume that an adversary may create many Sybil identities, but it cannot establish arbitrarily many links with honest nodes. We also assume that large number of fake identities can be created into a community to initiate large-scale attacks.

4. SybilExposer Design

In this section, we present our algorithm, *SybilExposer*. Our algorithm uses the ratio of *inter-community degree* (number of edges in a community) and *intra-community degree* (number of edges emanating from a community) of each community in the social graph to differentiate Sybil communities from honest communities. The intuition is that usually a community has fewer inter-community connections than intra-community connections. Also, generally nodes from an honest community tend to have larger number of inter-community edges with nodes in other honest communities compared to nodes in Sybil communities. Again, Sybil nodes may have high intra-community degree, but they tend to have on an average fewer connections with honest nodes in other communities. Hence the ratio of *inter-community degree* to *intra-community degree* of the Sybil nodes in a Sybil community will be smaller than that of the honest nodes in an honest community. This intuition is also the basis for the other random walk based algorithms in the literature [12], [14], [6]. Sybil communities have few inter-community edges with the honest communities, whereas the honest communities have high inter-community edges among themselves. This ensures that the random walks originating within honest communities seldom terminate in sybil communities. Our proposed solution, *SybilExposer*, leverages these observations and operates in two stages, which we present as two algorithms. In the first algorithm (stage), we extract the communities in the full graph by using a modified version of the Louvain method [4]. In the second algorithm (stage), we rank the communities by utilizing their inter-/intra-community degree information.

A. Definitions

Definition 4.1. (Partition modularity [4]) Partition modularity of an OSN $G(V, E)$ measures the number of links inside communities as compared to the number of links between communities. It is a scalar value in the range [-1, 1]. It can be defined as

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i k_j}{2m}\right] \delta(c_i, c_j), \tag{4.1}$$

where A_{ij} represents the weight of the edge between i and j, $k_i = \sum_j A_{ij}$ is the sum of weights of all edges attached to vertex i, c_i is the community to which vertex i is assigned, the δ function, $\delta(c_i, c_j)$ defined for a pair of community (c_i, c_j), is 1 if $c_i = c_j$ and 0 otherwise, and $m = \frac{1}{2}\sum_{i,j \in V} A_{ij}$. □

Definition 4.2. (Gain in modularity [4]) The gain in modularity ΔQ for a graph $G = (V, E)$, obtained by moving an isolated node i into a community C can be computed as

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right], \tag{4.2}$$

where $\sum_{in}$ is the sum of link weights of C, $\sum_{tot}$ is the sum of link weights of all links incident to nodes in C, k_i is the sum of weights of all links incident on node i, $k_{i,in}$ is the sum of weights of links from i to nodes in C, and m is the sum of weights of all links in the network. □

Definition 4.3. (Node Diversity) For $v \in V$ in the OSN graph $G = (V, E)$, the node diversity, $ND(v)$, is defined as $ND(v) =$ *inter-deg*(v)/*intra-deg*(v), where $ND(v) \geq 0$, and $ND(v) = 0$ when v is only connected to nodes within its community. □

Definition 4.4. (Community Diversity) Let C_i be the i-th community in the OSN graph, $G = (V, E)$, and $v \in V$ and $v \in C_i$. Then $CD(C_i)$, the community diversity of community C_i is defined as $CD(C_i) = \sum_{v \in C_i}$ *inter-deg*(v)/$\sum_{v \in C_i}$ *intra-deg*(v); $CD(C_i) \geq 0$, and $CD(C_i) = 0$ when C_i is isolated. □

B. First stage: Community Extraction algorithm

In the first stage, we modify the Louvain method [4], a community detection algorithm that clusters the graph into communities by iteratively optimizing the graph's partition modularity (ref. Definition 4.1). This method is a combination of two phases. In the first phase, which comprises of Steps 5 to 24 of Algorithm 1, initially each node represents a community. The algorithm iteratively merges the communities in a way that increases the partition modularity (Steps 9 to 23).

Steps 25 to 28 constitute the second phase. In this phase, a new graph $G' = (V', E')$ is constructed, where each node v'' represents a community in G' and each edge $e'' = (u'', v'')$ represents the inter-community edges between two communities $C_{u''}$ and $C_{v''}$. The weight of each e'' is the sum of weights of edges between $C_{u''}$ and $C_{v''}$. Then the first phase is re-applied for the newly created graph. The two phases of this algorithm are repeated until the cardinality of $\mathcal{C}$ can no longer be reduced. The computational cost of each iteration in this method is $\mathcal{O}(m)$. However, due to the rapid merging of communities this method requires a small number of passes, hence the total running time on amortization is $\mathcal{O}(m)$.

C. Second stage: Community Ranking algorithm

In the second stage, communities generated in the first stage are ranked in increasing order according to their likelihood of being Sybil. As we discussed in the beginning of this section, the ratio of inter-community degree to intra-community degree of Sybil communities would be much smaller than the ratio for honest communities. In other words, an honest community has greater diversity (connections with other communities) than a Sybil community, thus community diversity can be used as a candidate of ranking index. We further observed that in some cases, small honest communities may have few inter-community edges and more intra-community edges resulting in a lower community diversity than some Sybil communities; but in this case their node diversity may be larger. So we multiply the average node diversity by the community diversity and use the product value as the ranking index. The average node diversity also helps normalize the results of community size. It is likely to produce a smaller rank value for Sybil communities than for honest communities.

Algorithm 2 presents the community ranking algorithm. We find the total node diversity (Steps 5 to 8). Then we compute the

Algorithm 1 Community Extraction

Input: $G = (V, E)$, where G is the social graph
Output: $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, where C_k is the k-th community

1: Create $G'(V', E') = G(V, E)$.
2: For each edge, $e' \in E'$, assign weight $w(e) = 1$.
3: Assign modularity difference, $Q_{diff} = 0.000001$. {As per [4]}
4: **repeat**
5: Assign each node $i \in V'$ a unique community number C_i. {community number can start with 1}
6: $\mathcal{C} = \{C_1, C_2, \ldots, C_{|V'|}\}$.
7: Compute new modularity, Q_{new} as per Equation 4.1.
8: **repeat**
9: Current modularity, $Q_{current} = Q_{new}$.
10: **for each** node $i \in V'$ **do**
11: $max\Delta Q = 0, maxNbr = null$.
12: **for each** node $j \in NG(i)$ **do**
13: Compute modularity gain, ΔQ_{ij} from adding i to C_j as per Equation 4.2.
14: **if** $\Delta Q_{ij} \geq 0$ and $\Delta Q_{ij} > max\Delta Q$ **then**
15: $max\Delta Q = \Delta Q_{ij}$.
16: $maxNbr = j$.
17: **end if**
18: **end for**
19: **if** $max\Delta Q > 0$ **then**
20: $C_{maxNbr} \leftarrow C_{maxNbr} \cup C_i$.
21: **end if**
22: **end for**
23: Compute new modularity, Q_{new} as per Equation 4.1.
24: **until** $Q_{new} - Q_{current} \leq Q_{diff}$.
25: Create a new graph $G'' = (V'', E'')$, where $v \in V''$ represents $C_v \in \mathcal{C}$. {communities set is now smaller}
26: Each edge $e'' = (u, v) \in E''$ represents the inter-community connections between nodes in communities C_u and C_v.
27: Assign weight, $w(e'')$ = cardinality of inter-community edges between C_u and C_v.
28: $G'(V', E') = G''(V'', E'')$.
29: **until** the cardinality of $\mathcal{C}$ can no longer be reduced.
30: Output $\mathcal{C}$.

community diversity, $CD(C_i)$ and determine the rank value, $r(C_i)$ of community C_i in Steps 10 and 11. Steps 2 to 11 are repeated until we determine the rank values of all the communities. Step 13 sorts the communities in descending order of rank values. Then, we refine the ranking further as follows. Isolated Sybil communities can not adversely affect the network. So we remove all communities with zero rank values (Step 14).

Since small communities with small number of inter-community connections can have limited adverse effects on the network at best, and it is difficult to differentiate small Sybil from small honest communities, we drop them from the ranking. We filter out those communities by setting thresholds Th_{size}, on the community size, and Th_{intr}, on the total number of foreign edges of a community in Step 15. We discuss the choice of these threshold values that we obtain from an extensive training set in the next section. In Step 16, the algorithm proceeds from the bottom of the ranked list and fixes the pivot point at the previous rank value when the increase from the previous rank value to the current rank value is greater than a threshold τ, an input in this algorithm. Again, we discuss the choice of τ in the next section. In Step 17, communities with rank values less than or equal to the pivot are marked as Sybil communities, and returned as the Sybil set $\mathcal{S}$.

Algorithm 2 Community Ranking

Input: $G = (V, E)$, where G is the social graph, $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$, where C_n is the n-th community and $C_n \subset V$, Th_{size}, Th_{intr}, τ.
Output: $\mathcal{S} = \{C_1, C_2 \ldots C_k\}$, where $\mathcal{S} \subset \mathcal{C}$ and C_k is the k-th probable Sybil community.

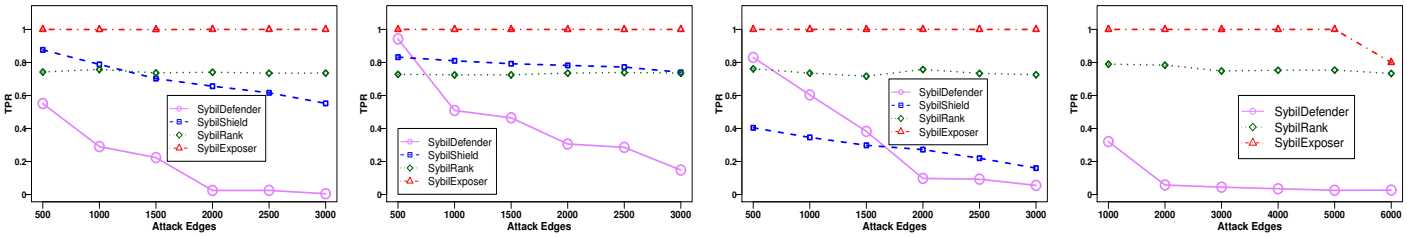
1: **for each** $C_i \in \mathcal{C}$ **do**
2: $r(C_i) \leftarrow 0$. {initialize rank of C_i}
3: $CD(C_i) \leftarrow 0$, $TND(C_i) \leftarrow 0$.
4: **for each** node $v \in C_i$ **do**
5: *intra-deg(v)* $= |Ita_v|$, where $Ita_v = \{(v, u) | v, u \in C_i$ and $(v, u) \in E\}$.
6: *inter-deg(v)* $= |Ite_v|$, where $Ite_v = \{(v, u) | v \in C_i, u \in C_j$ and $(v, u) \in E\}$.
7: $ND(v) = \frac{\textit{inter-deg}(v)}{\textit{intra-deg}(v)}$.
8: $TND(C_i) \leftarrow TND(C_i) + ND(v)$.
9: **end for**
10: $CD(C_i) = \sum_{v \in C_i} \textit{inter-deg}(v) / \sum_{v \in C_i} \textit{intra-deg}(v)$.
11: $r(C_i) \leftarrow \frac{TND(C_i)}{|C_i|} \times CD(C_i)$.
12: **end for**
13: Sort r values in descending order.
14: Remove all isolated communities C_i from the list ($r(C_i) = 0$). {isolated communities}
15: Remove all communities C_i from the ranked list where $|C_i| \leq Th_{size}$ and $\sum_{v \in C_i}$ inter-deg$(v) \leq Th_{intr}$.
16: Traverse the list starting from the bottom (C_0) and set pivot point at C_{i-1}, if $r(C_i) - r(C_{i-1}) \geq \tau$.
17: Set $\mathcal{S} = \{C_{i-1}, \ldots, C_0\}$, and return $\mathcal{S}$.

D. Computational complexity

The computational complexity of SybilExposer is $\mathcal{O}(n + c \log c)$, where n is the total number of nodes in the social graph and c is the total number of extracted communities. Overall, the running time complexity of the Steps 1 through 11 is $\mathcal{O}(n)$, because even though the computation of diversities is done per cluster or community, essentially it is performed per node in Steps 5 to 8. The cost for sorting the communities in descending order of value is $\mathcal{O}(c \log c)$. To find the pivot rank value the cost is $\mathcal{O}(c)$, where $c << n$. Thus, the overall computational complexity is $\mathcal{O}(n + c \log c)$. In comparison, the best algorithm in the literature, SybilRank, has a complexity of $\mathcal{O}(n \log n)$.

5. Evaluation

We evaluate the effectiveness of SybilExposer by performing experiments on several real-world SN datasets [9] of varying size, and analyzing the results in terms of true positive rate,



(a) ca-AstroPh (b) Facebook (c) soc-Epinions1 (d) Pokec

Fig. 2: True Positive Rate for different graphs (collaboration graph: *ca-AstroPh* and social-interaction graph: *Facebook, Epinion, Pokec*) with varying number of attack edges.

false positive rate, and running time. We compare SybilExposer against the best three current algorithms: SybilRank [6], SybilShield [12], and SybilDefender [14].

A. Datasets and Experiment Setup

Table I shows some properties of the social graphs used in our experiments. Note that l_0 and R are parameters for SybilDefender. In the experiments, we simulate two types of

TABLE I: Social graph Datasets used in our Experiments

Dataset	Nodes	Edges	l_0	R
ca-AstroPh	18,772	198,110	6	100
Facebook	63,731	817,035	20	500
soc-Epinions	75,879	508,837	24	500
soc-Pokec	1,632,803	30,622,564	540	1000

attack strategies to construct the Sybil communities: Single-Community strategy (**SC**), where only one Sybil community exists in the OSN; and Multi-Community (**MC**) strategy, where there are multiple Sybil communities. As per the literature ([6], [12], [14]), to generate the Sybil regions for both attack strategies, we choose the Barabasi-Albert (BA) scale-free model (small-world property) [2]. Each newly added Sybil node preferentially attaches to m other Sybil nodes with the attachment probability depending on their respective node degrees. Similar to [6], we chose $m = 4$ for all Sybil communities. The number of attack edges for a Sybil region is represented as g. An attack edge is created by connecting a Sybil node from the region and an honest node, both chosen randomly.

In our experiments, for all graphs with less than 1 million nodes we chose the number of attack edges as $g \in \{500, 1000, 1500, 2000, 2500, 3000\}$. In the **MC** strategy, we built 3 Sybil communities of the same size. The Sybil community had 5000 nodes for all graphs with nodes less than 1 million. But for the graph with nodes 1 million or greater, the Sybil community had 10,000 nodes and the number of attack edges was chosen as $g \in \{1000, 2000, 3000, 4000, 5000, 6000\}$. In every graph, we created 5 instances of Sybil communities for each attack edge configuration and averaged the results.

In SybilExposer, we used thresholds Th_{size} and Th_{intr} to discard small sized honest communities with limited number of total inter-community connections. From our observation over a subset of the graphs, setting $Th_{size} = 50$ and $Th_{intr} = 50$ for graphs with $n \leq 50,000$ and setting $Th_{size} = 100$ and $Th_{intr} = 100$ for graphs with $n > 50,000$, works well. By using learning on a few of the datasets, we identified the value of τ to be 0.01, which represents the minimum difference between two consecutive ranks for placement of the pivot. For SybilRank, we follow the method used in the paper to select the trusted seeds (t_s), thus $|t_s| = 50$. One trusted seed was chosen randomly from the top-10 honest nodes with highest degrees and the remaining 49 seeds were chosen randomly.

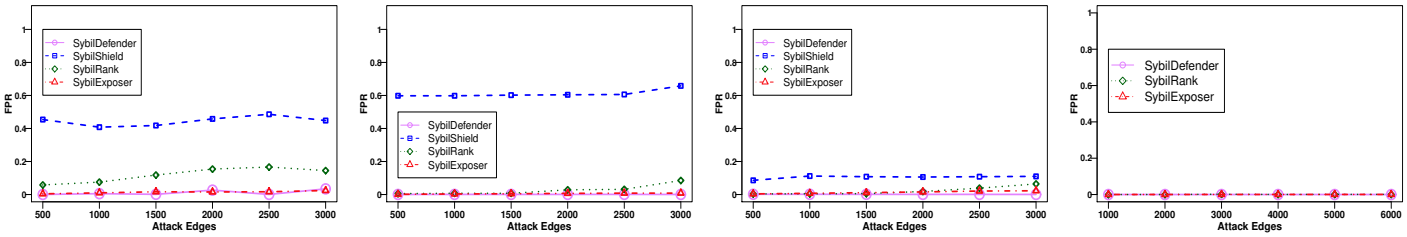
For SybilDefender, we compared with the community detection algorithm whose parameters are as follows: l_0, R and β. We used $\beta = 0.95$ on all datasets; the other two parameters were scaled down based on dataset size to appropriate values, following the mechanism in [14] (ref. Table I).

B. Experiment Results

We ran all the algorithms on a machine with 8GB memory and an Intel Core i7 2.93 GHz processor. We define the true positive rate (**TPR**) as the percentage of Sybil nodes correctly identified to be Sybil and the false positive rate (**FPR**) as the percentage of the honest nodes falsely identified to be Sybil. For the pair-wise verification procedure, SybilShield, it was infeasible to examine all pairs of verifier and suspect nodes to obtain the exact TPR and FPR. So to estimate the rates in each experiment we randomly select 100 verifier, honest suspect, and Sybil suspect nodes respectively.

*1) Single-Community (**SC**) Attack Strategy:* Figure 2 shows the results of TPR for different datasets in the **SC** attack scenario when the number of attack edges increased from 500 to 3000. The TPR of SybilDefender degrades drastically with the increase of attack edges for all graphs. This is because of its strong reliance on the random walks not traversing the limited attack edges between the Sybil and the honest regions to identify Sybil nodes. With increasing attack edges, the probability of random walks reaching the honest region from a Sybil node also increase. This results in more Sybil nodes being identified as honest nodes, thus reducing the TPR. SybilShield's TPR follows a similar trend. As the attack edges increase, the probability of intersections of the random walks from the verifiers and suspect nodes increase in SybilShield. This reduces the TPR. Note that, due to the very high running time of SybilShield (even for 100 suspect nodes) we were not able to represent its performance on the Pokec graph.

On an average the TPR of SybilRank was 71%-78%, while in SybilExposer it is lower bounded by 80% and goes up to almost 100%. SybilRank uses $t_s = 50$ trusted nodes to propagate trust values to all nodes and identifies nodes with low trust value as Sybil. However, if a Sybil node is attached close to a few trusted nodes, it may gain a high trust value and thus will be identified as honest. The chance of this happening also increases with the number of attack edges. On the other hand,



(a) ca-AstroPh (b) Facebook (c) soc-Epinions1 (d) Pokec

Fig. 3: False Positive Rate for different graphs with varying number of attack edges

SybilExposer does not use any trusted nodes in the social graph and does not depend on the random walks. Instead, it uses the node diversity and community diversity properties of the social graph to identify Sybil nodes as a community. Also, SybilExposer aims to identify the Sybil nodes as part of a Sybil community, rather than trying to identify each Sybil node individually. Our algorithm takes advantage of the intrinsic nature of communities to distinguish between honest and Sybil communities. Sybil communities and honest communities differ in terms of the inter-community edges cardinality, which does not change appreciably with the increase of attack edges; this helps SybilExposer in getting consistently better results than the other algorithms. To our best knowledge, SybilExposer is the first algorithm in the literature that leverages the intrinsic nature of communities, which helps it obtain better outcomes.

Figure 3 presents the average FPR of all algorithms. SybilExposer maintains about 0%-3% FPR on an average for all datasets, whereas SybilRank maintains about 0%-9% FPR and SybilDefender almost 0% FPR. As we can see in Figure 3, SybilShield averages about 11%-65% FPR for the medium order datasets, which is higher than the other algorithms. Hence, Figures 2 and 3 show that SybilExposer outperforms other algorithms significantly in terms of TPRs and FPRs.

*2) Multi-Community (**MC**) Attack Strategy:* The TPR of SybilExposer decreases to 93.33% for soc-Epinions1 dataset at 3000 attack edges when there are multiple Sybil communities, but it is still 22% higher than for SybilRank. SybilRank also has the best FPR, 0%-7% only. Due to space limitation we could not present the performance comparisons in the MC attack scenario.

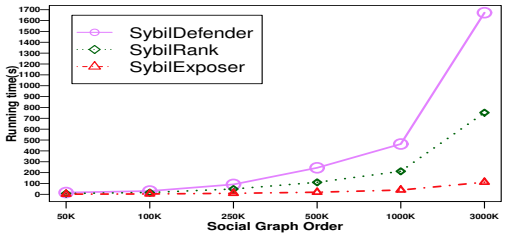


Fig. 4: Running time of different algorithms.

3) Running Time: To compare the running time of the algorithms we created five instances of six BA model based synthetic graphs. Figure 4 shows the average running time. SybilExposer performs much faster in comparison to the other schemes, whose running time rises rapidly with the number of nodes. This is attributable to two facts. First, the other algorithms in the literature run several random walks to identify the Sybil communities, which contributes significantly to the running time (best case running time is $\mathcal{O}(n \log n)$). Second, our algorithm does not use random walks, instead it focuses on intrinsic community properties in the network, which can be calculated in $\mathcal{O}(n + c \log c)$, where n and c are the number of nodes and clusters respectively. This demonstrates the scalability of SybilExposer.

6. Conclusions and Future Work

In this paper we introduce SybilExposer, a Sybil communities detection algorithm for OSNs, which uses the topological properties of communities in OSNs for the Sybil identification. Our simulation results show that SybilExposer performs better in effectiveness and computational cost compared to the state of the art. In the future, we will extend our scheme to handle non-community Sybil nodes and edges between Sybil communities.

References

[1] Facebook, Inc.: Annual report. http://www.sec.gov/Archives/edgar/data/1326801/000132680114000007/fb-12%312013x10k.htm, January 2014.

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

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

[4] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[5] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. Design and analysis of a social botnet. *Computer Networks*, 57(2):556–578, 2013.

[6] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, pages 197–210, 2012.

[7] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[8] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Link mining: models, algorithms, and applications*, pages 337–357. Springer, 2010.

[9] J. Kunegis. Konect: the koblenz network collection. In *Proceedings of the WWW*, pages 1343–1350, 2013.

[10] A. Langville and C. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[11] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 383–389, 2010.

[12] L. Shi, S. Yu, W. Lou, and Y. Hou. Sybilshield: An agent-aided social network-based sybil defense among multiple communities. In *IEEE INFOCOM*, pages 1034–1042, 2013.

[13] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Zhao. You are how you click: Clickstream analysis for sybil detection. In *USENIX Security*, pages 241–256, 2013.

[14] W. Wei, F. Xu, and *et al.* Sybildefender: Defend against sybil attacks in large social networks. In *IEEE INFOCOM*, pages 1951–1959, 2012.

[15] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Zhao, and Y. Dai. Uncovering social network sybils in the wild. *ACM TKDD*, 8(1):2, 2014.

[16] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.

[17] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 267–278, 2006.