SybilDefender: Defend Against Sybil Attacks in Large Social Networks

Wei Wei*, Fengyuan Xu*, Chiu C. Tan†, Qun Li*
*The College of William and Mary, †Temple University
*{wwei, fxu, liqun}@cs.wm.edu, †cctan@temple.edu

***Abstract*—Distributed systems without trusted identities are particularly vulnerable to sybil attacks, where an adversary creates multiple bogus identities to compromise the running of the system. This paper presents SybilDefender, a sybil defense mechanism that leverages the network topologies to defend against sybil attacks in social networks. Based on performing a limited number of random walks within the social graphs, SybilDefender is efficient and scalable to large social networks. Our experiments on two 3,000,000 node real-world social topologies show that SybilDefender outperforms the state of the art by one to two orders of magnitude in both accuracy and running time. SybilDefender can effectively identify the sybil nodes and detect the sybil community around a sybil node, even when the number of sybil nodes introduced by each attack edge is close to the theoretically detectable lower bound. Besides, we propose two approaches to limiting the number of attack edges in online social networks. The survey results of our Facebook application show that the assumption made by previous work that all the relationships in social networks are trusted does not apply to online social networks, and it is feasible to limit the number of attack edges in online social networks by relationship rating.**

I. Introduction

Distributed systems are vulnerable to sybil attacks [7], in which an adversary creates many bogus identities, called sybil identities, and compromises the running of the system or pollutes the system with fake information. The sybil identities can "suppress" the honest identities in a variety of tasks, including online content ranking, DHT routing, file sharing, reputation systems, and Byzantine failure defenses. There are some similar attacks in ad hc and sensor networks [16], [17].

Sybil attacks can be mitigated by assuming the existence of a trusted authority, which can rate-limit the introduction of fake identities by requiring the users to provide some credentials, like social security number, or by requiring payment. However, such requirements will prevent users from accepting these systems, as they impose additional burdens on users.

Recently, there has been an increasing interest in defending against sybil attacks in social networks [6], [12], [18], [19], [20]. In a social network, two user identities share a link if a relationship is established between them. Each identity is represented as a node in the social graph. To prevent the adversary from creating many sybil identities, all the previous sybil defense schemes are built upon the assumption that the number of links between the sybil nodes and the honest nodes, also known as *attack edges*, is limited. As a result, although an adversary can create many sybil nodes and link them in an arbitrary way, there will be a *small cut* between the honest region and the sybil region. The small cut consists of all the attack edges and its removal disconnects the sybil nodes from the rest of the graph, which is leveraged by previous schemes to identify the sybil nodes. Note that the solution to this problem is non-trivial, because finding small cuts in a graph is an NP hard problem. To limit the number of attack edges, previous schemes assume that all the relationships in social networks are trusted and they reflect the trust relationships among those users in the real world, and thus an adversary cannot establish many relationships with the honest users. However, it has been shown that this assumption does not hold in some real-world social networks [5].

In the past few years, online social networks have gained great popularity and are among the most frequently visited sites on the Web [3]. The large sizes of these networks require that any scheme aiming to defend against sybil attacks in online social networks should be efficient and scalable. Some previous schemes can achieve good performance on a very small network sample (2000 nodes in [18] and 30000 nodes in [6]), but their algorithms are computationally intensive and cannot scale to networks with millions of nodes. For the schemes that performed evaluation on million-node samples of online social networks, SybilGuard [20] admits $O(\sqrt{n}\log n)$ sybil nodes per attack edge, where n is the number of honest nodes; SybilLimit [19] improves over SybilGuard by accepting $O(\log n)$ sybil nodes per attack edge, but it is still away from the theoretical lower bound by a $\log n$ factor. Besides, both SybilGuard and SybilLimit identify one sybil node at a time, and thus to detect the sybil region all the nodes in the social graph need to be examined.

To address the weaknesses of previous work, in this paper we propose SybilDefender, a centralized sybil defense mechanism. It consists of a sybil identification algorithm to identify the sybil nodes, a sybil community detection algorithm to detect the sybil community surrounding a sybil node, and two approaches to limiting the number of attack edges in online social networks. Our scheme is based on the observation that a sybil node must go through a small cut in the social graph to reach the honest region. An honest node, on the contrary, is not restricted. Now if we start from a sybil node to do random walks, the random walks tend to stay within the sybil region. The main contributions of this work include:

- Based on performing a limited number of random walks within the social graphs, the sybil identification algorithm and the sybil community detection algorithm are efficient

and scalable to large social networks.

- We evaluate SybilDefender using two large-scale social network samples from Orkut and Facebook, respectively. The results show that the performance of our sybil identification algorithm approaches the theoretical bound, and it outperforms SybilLimit, the state of the art sybil defense mechanism that applies to large social networks, by one to two orders of magnitude in both accuracy and running time. Besides, our sybil community detection algorithm can effectively detect the sybil community around a sybil node with short running time.
- We propose two practical techniques to limit the number of attack edges in online social networks, and develop a Facebook application to demonstrate the feasibility of one of the techniques. The survey results of our Facebook application show that the assumption made by previous work that all the relationships in social networks are trusted does not hold in online social networks, and it is feasible to limit the number of attack edges in online social networks by relationship rating.

II. Related Work

One promising way to defend against sybil attacks in social networks is to leverage the social network topologies. Yu et al. proposed decentralized algorithms, SybilGuard [20] and SybilLimit [19], to determine whether a suspect node is sybil or not. SybilGuard and SybilLimit both rely on the assumption that social networks are fast mixing (explained later), and the number of attack edges is limited. To identify sybil nodes, the schemes make use of random routes, a special kind of random walks in which each node uses a pre-computed random permutation as a one-to-one mapping from incoming edges to outgoing edges. SybilGurad suffers from high false negatives, as each attack edge may introduce $O(\sqrt{n}\log n)$ sybil nodes without being detected. The improved version of SybilGuard, SybilLimit, reduces this value to $O(\log n)$, which is still larger than the proved lower bound $\Omega(1)$ [19] by a $\log n$ factor. Moreover, to detect the sybil region with SybilGuard or SybilLimit, all the suspect nodes in the social graph need to be tested. By contrast, with our sybil community detection algorithm, the sybil community around a sybil node can be detected in one run of the algorithm. GateKeeper [12] is another decentralized sybil defense scheme that heavily relies on the assumption that the social networks are random expander. This is a strong assumption which has not been validated by previous research. Our evaluation shows that GateKeeper suffers from high false positive and negative rates and cannot effectively identify sybil nodes on the real-world asymmetric social topologies.

SybilInfer [6], a centralized sybil defense algorithm, leverages a Bayesian inference approach that assigns a sybil probability, indicating the degree of certainty, to each node in the network. It achieves low false negatives at the cost of high computation overhead. The overall time complexity of SybilInfer is $O(|V|^2 \log |V|)$, where V is the set of vertices in the social graph. In the evaluation SybilInfer handled networks with up to 30K nodes, which is much smaller than the size of regular online social networks. The algorithm proposed by Xu et al. [18] calculates the shortest path between every pair of nodes within the network in each round, which makes it impractical for even small-sized social networks. In contrast, SybilDefender only relies on performing a limited number of random walks in the social graph, and it is scalable to large networks.

III. System Model

We denote the social network as a graph G consisting of vertices V and edges E. There are n honest users in the social network, each with one identity, denoted as an *honest node* in V. There are also one or more malicious users in the social network, each with a number of sybil identities. Each sybil identity is denoted as a *sybil node* in V. A relationship between two identities in the social network is represented as an edge connecting the two corresponding nodes in G. The edges in G are undirected. We name the edge between a sybil node and an honest node an *attack edge*. The *sybil region* consists of all the sybil nodes, while the *honest region* consists of all the honest nodes. All the sybil nodes are controlled by an adversary. Thus the adversary can create arbitrary edges within the sybil region.

SybilDefender is built upon the following assumptions:

The honest region is fast mixing. Fast mixing means a random walk of length $\Theta(\log n)$ is long enough such that with probability at least $1 - \frac{1}{n}$, the last traversed node is drawn from the node stationary distribution of the graph [20]. The stationary distribution is a probability distribution $\overline{\pi}$ for V such that $\overline{\pi} = \overline{\pi}P$, where P is the transition matrix of the random walk process [11]. It can be easily proved that $\overline{\pi}_i$, the stationary probability of node i, is equal to $\frac{d_i}{2|E|}$. Yu et al. have shown that the real-world social networks are fast mixing [19]. The previous sybil defense schemes [6], [19], [20] are also built upon this assumption.

One known honest node. We assume that there is at least one known honest node in the social network. This node is the starting point of our sybil identification algorithm.

The administrator knows the social network topology. This means that SybilDefender is a centralized sybil defense mechanism. Considering that all the current online social networks are under centralized control, it is natural for the administrators of these networks to take charge of mitigating sybil attacks.

The size of the sybil region is not comparable to the size of the honest region. Given the large user base of the current online social networks (Facebook (over 500 million), Twitter (over 200 million), Orkut (over 120 million)), it is reasonable to assume that the adversary cannot create such many sybil identities, especially considering that signing up a new user account always includes verifying an email address, providing some personal information, and solving CAPTCHAs.

The number of attack edges is limited. As a result, when the adversary creates many sybil nodes, there will be a disproportionately small cut between the honest region and the sybil region. The existence of a small cut disturbs the fast-mixing property: the mixing between the honest nodes

is fast, while the mixing between the honest nodes and the sybil nodes is slow. Previous schemes limit the number of attack edges by assuming that the honest users only establish links with their real-world friends [6], [12], [19], [20], which has been shown to not hold in online social networks. The experiment by Bilge et al. [5] shows that on Facebook, the acceptance rate of friendship requests from a bogus account is around 20%. If an adversary launches a sybil attack, all the links created in this way are attack edges. We will address this problem in Section IV-C.

IV. SybilDefender Design

SybilDefender consists of three components: a sybil identification algorithm, a sybil community detection algorithm, and two supporting approaches to limiting the number of attack edges. The three components can be used in conjunction to best mitigate sybil attacks. The task of the sybil identification algorithm presented in Section IV-A is to determine whether a suspect node is sybil or not. Then we show how to efficiently detect the sybil community around a sybil node with our sybil community detection algorithm presented in Section IV-B. The reason why we need the second algorithm is that simply examining all the nodes in the social graph to find the sybil community is impractical. Finally, both algorithms are built upon the assumption that the number of attack edges is limited. In Section IV-C we propose two approaches to supporting this assumption in online social networks.

A. Sybil Identification Algorithm

In this subsection we present a sybil identification algorithm that takes the social graph $G(V, E)$, a known honest node h, and a suspect node u as inputs, and outputs whether u is sybil or not. Our algorithm is based on random walks. A random walk on a graph is defined by the sequence of moves of a particle between nodes of G. If the particle is at node i with degree d_i, then the probability that the particle follows the edge (i, j) and moves to a neighbor j is $1/d_i$.

The intuition of our sybil identification algorithm is that, as there is a small cut between the honest region and the sybil region, the random walks originating from a sybil node tend to get "trapped" into the sybil region. Also, since we assume that the size of the sybil region is not comparable to the size of the honest region, the number of nodes traversed by the random walks originating from an honest node will be larger than the number of nodes traversed by the random walks originating from a sybil node, as long as the random walks are long enough and we perform the random walks many times. For simplicity, we define the number of times one node being traversed by a set of random walks as the *frequency* of that node. Note that one node may be traversed by the same random walk multiple times.

The sybil identification algorithm consists of two phases, Algorithm 1 and Algorithm 2. The first phase takes G and h as inputs, and outputs the thresholds used by the second phase to identify sybil nodes. It only needs to be invoked once for each social network topology. As shown in Algorithm 1, the

Algorithm 1 PreProcessing(G, h)

```
J = {h}
for i = 1 to f do
    Perform a random walk with length l_s = log n originating from h
    J = J ∪ {the ending node of the random walk}
end for
l = l_min
while l <= l_max do
    for i = J.first() to J.last() do
        Perform R random walks with length l originating from node i
        Get n_i as the number of nodes with frequency no smaller than t
    end for
    output ⟨l, mean({n_i : i ∈ J}), stdDeviation({n_i : i ∈ J})⟩
    l = l + 100
end while
```

algorithm first performs f short random walks with length $l_s = \log n$ originating from the known honest node h. The f ending nodes are drawn from the node stationary distribution of the honest region, since we assume that the honest region is fast mixing. Following the proof in [20], the ending nodes are all honest nodes with high probability. After this the known honest node h and the f ending nodes are treated as *judge nodes*, from which the algorithm sets up the criterion to identify sybil nodes. Note the possibility that sybil nodes may exist in the group of the judge nodes does not influence the effectiveness of the algorithm, due to their very limited number. Starting from a minimum length l_{min} to a maximum length l_{max}, with an interval of 100 hops, for each length l, the algorithm performs R (ranging from 1000 to 2000 in our evaluation) random walks originating from every judge node, and counts the number of nodes whose frequency is no smaller than a threshold t, which is a small constant (5 in our evaluation). The algorithm collects $f + 1$ such values for each length l. Then it computes the mean and standard deviation of the $f + 1$ values and outputs a tuple as $\langle l, mean, stdDeviation\rangle$.

Algorithm 2 SybilIdentification($G, u, tuples\ from\ Alg.1$)

```
l = l_0
while l <= l_max do
    Perform R random walks with length l originating from u
    m = the number of nodes whose frequency is no smaller than t
    Let the tuple corresponding to length l in the outputs of Algorithm 1
    be ⟨l, mean, stdDeviation⟩
    if mean − m > stdDeviation * α then
        output u is sybil
        end the algorithm
    end if
    l = l * 2
end while
output u is honest
```

As shown in Algorithm 2, to determine whether a suspect node u is sybil, the algorithm first performs R random walks with an initial length $l = l_0$ originating from u. l_0 is larger than or equal to l_{min} used in Algorithm 1. The algorithm then compares the number of nodes whose frequency is no smaller than t with the $mean$ value in tuple $\langle l, mean, stdDeviation\rangle$ outputted by Algorithm 1. If the former is smaller than the latter by an amount larger than $stdDeviation * \alpha$ ($\alpha = 20$ in

TABLE I
NOTATIONS USED IN THE ANALYSIS

$G(V, E)$	social graph, V is the set of nodes, E is the set of edges
P	transition matrix of the random walk process
n	number of honest nodes in G
λ	initial state vector of the random walk process
$\overline{\pi}$	stationary distribution of G
l	random walk length
Q_l	accumulated probability distribution of the nodes being traversed by a random walk with length l
t	threshold frequency used in the sybil identification algorithm
R	number of random walks originating from a given node
$\mathcal{D}(d)$	number of nodes with degree d

our evaluation), we consider u is sybil and end the algorithm. Otherwise, the algorithm doubles l and repeats the process, until l is larger than l_{max}. If u is still not identified as sybil when the value of l reaches l_{max}, we consider it honest and end the algorithm.

Given a social graph $G(V, E)$ and a known honest node h, l_{max}, the maximum random walk length that decides when to end the algorithm, can be determined as follows. We do R random walks originating from h with length l_{max}. The number of nodes with frequency no smaller than t should be larger than $|V|/2$. Given that we assume the sybil region is smaller than the honest region, l_{max} determined in this way is large enough for R random walks originating from a sybil node to cover the sybil region, so as to exhibit the difference between the random walks originating from an honest node and from a sybil node. Our algorithm adaptively tests the suspect node while doubling the random walk length each time. This guarantees that the algorithm can identify the sybil nodes in differently sized sybil regions: for small sybil regions short random walks are already enough, while for large regions long random walks need to be performed, since the footprint of short random walks in a large sybil region may be similar to that in the honest region.

1) Analysis of the Sybil Identification Algorithm: In this subsection we investigate the validity of our sybil identification algorithm with theoretical analysis. For the ease of analysis we list the used notations in Table I. A random walk with length l on an undirected graph G can be modeled as a Markov Chain process. The starting state of the random walk is described as λ, the initial state vector of V. $\lambda_v = 1$ if v is the starting node of the random walk, otherwise $\lambda_v = 0$. As defined in Section III, P is the transition matrix of the random walk process. Therefore, the probability distribution of the nodes being visited by the i^{th} hop of the random walk is λP^i. Based on our fast-mixing assumption, λP^i converges to the stationary distribution $\overline{\pi}$ of G with $i \geq \Theta(\log n)$. The accumulated probability distribution of nodes being traversed by a random walk with length l is $Q_l = \sum_{i=0}^{l} \lambda P^i$, and $(Q_l)_j$, the j^{th} element in vector Q_l, is the expected number of times node j being traversed by a random walk with length l. Therefore, $R \cdot (Q_l)_j$ is the expected number of times node j being traversed by R random walks with length l originating from the same honest node, i.e., the expected frequency of j.

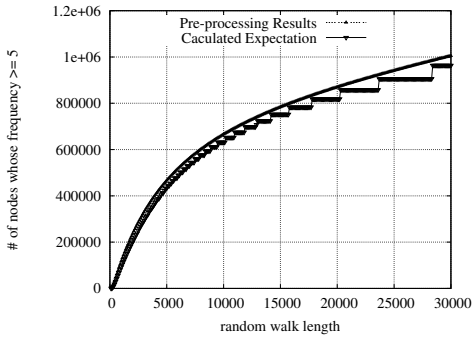

Fig. 1. pre-processing results and calculated expectation values

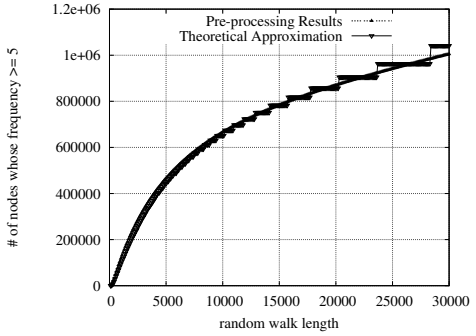

Fig. 2. pre-processing results and theoretical approximate results

The pre-processing phase of our sybil identification algorithm sets up the criterion to identify sybil nodes by performing R random walks originating from each judge node with every length value $l \in \{l_{min}, l_{min} + 100, ..., l_{max}\}$, respectively, and records the mean and the standard deviation of the number of traversed nodes with frequency no smaller than t. Define set S_l as $\{j | R \cdot (Q_l)_j \geq t\}$, then $|S_l| = |\{j | \sum_{i=0}^{l} R \cdot (\lambda P^i)_j \geq t\}|$ is the expected number of nodes whose frequency is no smaller than t with R l-hop random walks. With a randomly chosen source node and $R = 2000$, based on our Facebook data set, we calculate $|S_l|$ for different lengths and draw the *calculated expectation* curve in Figure 1. To demonstrate the validity of our sybil identification algorithm, we set the number of judge nodes to be 10 and draw the *pre-processing results* curve based on the mean value outputs of the pre-processing phase in the same figure. It shows that even with a small number of judge nodes, the two curves match well when the random walk length is smaller than 10000 hops. As the random walk length increases there shows some horizontal segments in the calculated expectation curve. This is because in a fast-mixing network, $(\lambda P^i)_j$ converges to $\frac{d_j}{2|E|}$ with $i \geq \Theta(\log n)$, where d_j is the degree of node j and E is the set of edges. This means that with $l \geq \Theta(\log n)$ the value of $\sum_{i=0}^{l} (\lambda P^i)_j$ for all the nodes with the same degree increases by the same amount when l increases by 1, and thus their expected frequency, $\sum_{i=0}^{l} R \cdot (\lambda P^i)_j$, will reach the threshold t at the same random walk length, which leads to the jumps in the calculated expectation curve. Note that although the calculated expectation curve is divided into horizontal segments when the

random walk length is large, its inflection points still match well with the pre-processing results curve. Figure 1 illustrates that *with a small number of judge nodes and limited R*, the results derived from the pre-processing phase of the sybil identification algorithm are already accurate enough to match with the expectation values.

Moreover, we will show that the results derived from the pre-processing phase *starting from a random honest node* are generic enough to serve as the criterion to identify sybil nodes. Since the network is fast mixing, given a random starting node, i.e., a random initial state vector λ, we have

$$\begin{aligned} Q_l &= \sum_{i=0}^{l} \lambda P^i \\ &\approx \lambda + \lambda P + ... + \lambda P^{\Theta(\log n)-1} + \underbrace{\overline{\pi} + ... + \overline{\pi}}_{l-\Theta(\log n)+1} \\ &\approx l\overline{\pi}. \end{aligned}$$

Besides, $\overline{\pi}_j = \frac{d_j}{2|E|}$, then we have $(Q_l)_j \approx l\frac{d_j}{2|E|}$. Recall that $R \cdot (Q_l)_j$ is the expected frequency of node j. To make this value no smaller than t, we have

$$R \cdot (Q_l)_j \approx R \cdot l\frac{d_j}{2|E|} \geq t \Rightarrow d_j \geq \frac{2t|E|}{lR}. \quad (1)$$

Define $S'_l = \{j | d_j \geq \frac{2t|E|}{lR}\}$. Then $|S'_l|$ is the approximate number of nodes whose frequency is no smaller than t with R l-hop random walks. Let $\mathcal{D}(d)$ be the number of nodes with degree d. Then

$$|S'_l| = \sum_{d=\lceil \frac{2t|E|}{lR} \rceil}^{\max} \mathcal{D}(d). \quad (2)$$

Following Equation 2 we draw the *theoretical approximation* curve in Figure 2 based on our Facebook data set, and we compare it with the pre-processing results curve identical to that in Figure 1. Note that Equation 2 is irrelevant to the initial state vector λ, so the shape of the theoretical approximation curve does not rely on the starting node. Figure 2 shows that the pre-processing results also match well with the theoretical approximate results. Similar to the caculated expectation curve, there are horizonal segments in the theoretical approximation curve. This is because node degrees are integers and l needs to increase by a certain amount such that the value of $\frac{2t|E|}{lR}$ reaches the next integer. Nevertheless, the middle point of each horizonal segment still matches with the pre-processing results curve. Figure 2 illustrates that the pre-processing results drawn from a random honest node can be effectively used as the criterion to identify sybil nodes.

To gain an understanding of the difference between the footprint of random walks originating from an honest node and from a sybil node, assume φ is the expected number of hops for the random walks starting from a sybil node to enter the honest region. If we draw the curve for the number of nodes with frequency no smaller than t based on the random walks starting from that sybil node, it is approximately like moving the pre-processing results curve in Figures 1 and 2 to the right by φ and then raising it by the size of the sybil region. In the evaluation we will show that this difference is large enough to identify the sybil nodes.

B. Sybil Community Detection Algorithm

After one sybil node is identified, our sybil community detection algorithm can be used to detect the sybil community surrounding it. The sybil community detection algorithm takes the social graph $G(V, E)$ and a known sybil node s as inputs, and outputs the sybil community around s. The sybil node s can be identified by our sybil identification algorithm or any previous scheme. We define a sybil community as a subgraph of G consisting of only sybil nodes, and there is no small cut in this subgraph. The reason why we make this definition is that if a small cut does divide the sybil region into two parts S_1 and S_2, and the known sybil node s is in S_1, then, from the point of view of s, the honest region and S_2 are similar, since there is already a small cut between S_1 and the honest region and also a small cut between S_1 and S_2. When there is a small cut in the sybil region, our algorithm can detect the sybil community s belongs to.

Our algorithm relies on performing *partial* random walks originating from s. Each partial random walk behaves the same as the *standard* random walks used in the previous subsection, except that it does not traverse the same node more than once. Therefore, when a partial random walk reaches a node with all the neighbors traversed by itself, this partial random walk is "dead" and cannot proceed. This property makes a partial random walk originating from a sybil node less likely to leave the sybil region, compared with a standard random walk, since many such walks "die" when they hit the border of the sybil region. Similar to the sybil identification algorithm, the intuition behind this algorithm is that the partial random walks originating from a sybil node tend to be trapped within the sybil region, and thus we can detect the sybil community by examining the nodes traversed by the partial random walks.

Algorithm 3 WalkLengthEstimation(G, s)

```
1: l = l_0/2
2: deadWalkRatio = 0
3: while deadWalkRatio < β do
4:     l = l * 2
5:     deadWalkNum = 0
6:     for i = 1 to R do
7:         Perform a partial random walk originating from s with length l
8:         if the partial random walk is dead before it reaches l hops then
9:             deadWalkNum++
10:        end if
11:    end for
12:    deadWalkRatio = deadWalkNum / R
13: end while
14: output l
```

The sybil community detection algorithm consists of two phases, Algorithm 3 and Algorithm 4. The task of Algorithm 3 is to estimate the needed length of the partial random walks used in Algorithm 4. Starting from an initial length l_0, the algorithm performs R partial random walks originating from s and counts the ratio of dead walks, which are the walks

that cannot proceed before they reach the required length. If this ratio is smaller than β, a threshold close to 1 (0.95 in our evaluation), the algorithm doubles the current length and performs the partial random walks again. This process is repeated until the dead walk ratio is no smaller than β. Then the algorithm outputs the current random walk length l. The reasoning is that the number of untraversed sybil nodes is very small (often equals to 0 in our evaluation) when the dead walk ratio is close to 1 and with a relatively large R (2000 in our evaluation).

Algorithm 4 SybilRegionDetection($G, s, l\ from\ Alg.3$)

```
Set the frequency of all the nodes to be 0
for i = 1 to R do
    Perform a partial random walk originating from node s with length l
    s.frequency++
    for j = 1 to l do
        Let the j^th hop of the partial random walk be node k
        k.frequency++
    end for
end for
traversedList = Sort the traversed nodes by their frequency in decreasing order
counter = 0
S = ∅
do
    counter = conductance(S)
    for i = traversedList.first() to traversedList.last() do
        if node i ∈ S then
            continue
        end if
        if conductance({i} ∪ S) <= conductance(S) then
            S = {i} ∪ S
        end if
    end for
while (counter > conductance(S))
output S
```

Algorithm 4 takes G, s, and the estimated length l as inputs and outputs the sybil community surrounding s. The reason why we need Algorithm 4 is that not all the nodes traversed by the partial random random walks in Algorithm 3 are sybil nodes, as some walks pass the small cut and enter the honest region, and we need an algorithm to select the sybil nodes from the set of traversed nodes. To achieve this, Algorithm 4 leverages a metric called *conductance* [9], defined as follows. Let d be the sum of the degrees of all the nodes in set S, and a be the number of edges with one endpoint in S and one endpoint in $\overline{S}$. Then the conductance of S is a/d. The conductance of a set S measures the quality of the cut between S and $\overline{S}$: the smaller the conductance is, the smaller the cut is. Since we assume that there is a small cut between the honest region and the sybil region, using conductance as the objective of the greedy algorithm fits the problem well. In this algorithm we let the conductance of an empty set be 1.

Algorithm 4 runs by first performing R partial random walks originating from the known sybil node s, with the length decided by Algorithm 3. Then the algorithm sorts all the traversed nodes by their frequency in decreasing order. Starting from the first node, which is always s, the algorithm iterates the sorted list and adds the encountered node to set S if doing so does not increase the conductance of S. After all the nodes in the sorted list are examined, the algorithm records the current conductance value, starts a new iteration from the top of the list and examines each node that is not in S. This process is repeated until the conductance value stays the same at the end of two consecutive iterations. Then the algorithm outputs S as the detected sybil community. The intuition is that by performing the partial random walks originating from a sybil node with suitable length many times, the sybil community surrounding the sybil node is covered by the partial random walks. Also, the sybil nodes tend to be in front of the honest nodes in the sorted list, since a large number of partial random walks cannot enter the honest region. As a result, the greedy algorithm will first try to add the nodes that are more likely sybil to S. This algorithm only relies on performing R partial random walks originating from a sybil node, which makes it very efficient and scalable to large-sized social networks.

C. Limiting the Number of Attack Edges

It has been shown that not all the relationships in online social networks are trusted [5]. One approach to limiting the number of attack edges in these networks is to allow the users to rate their relationships. To demonstrate this we develop a Facebook application named *Rate Your Relationships* [2]. The users of the application can rate each of their relations on Facebook either as "Friend" or "Stranger", where "Stranger" means the user hardly has any impression about this relation. The number of attack edges can be limited by removing the relationships rated as stranger from the social graph when applying the sybil defense schemes. The rationale is that even if an adversary can create many links between the sybil identities and the honest identities, it is hard for him to convince the honest users that those sybil identities are their acquaintances. The survey results of our Facebook application are presented in Section V-D.

We can also use the concept of *activity network* [13], [15] to limit the number of attack edges. Activity network is a network graph that is based on the interaction between users, rather than mere relationship. Two nodes share an edge in an activity network if and only if they have interacted directly through the communication mechanisms or applications provided by the corresponding social network. If the sybil defense schemes leverage the topologies of the activity networks, the number of attack edges an adversary can create can be further limited.

V. Evaluation

A. Data Sets and Experiment Setup

In this section we evaluate the effectiveness of SybilDefender using two data sets [10], [15] from Orkut and Facebook, respectively. The Orkut data set consists of 3,072,441 nodes and 117,185,083 edges, with an average degree of 76.28, while the Facebook data set consists of 3,097,165 nodes and 28,377,481 edges, with an average degree of 18.32. To the best of our knowledge, these are the largest data sets that have ever been used in evaluating the sybil defense schemes that leverage social network topologies.

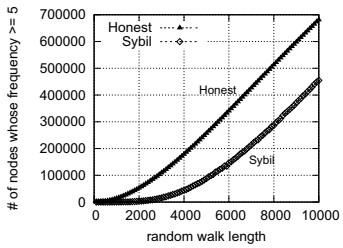

Fig. 3. Difference between the coverage of random walks originating from honest nodes and from sybil nodes

In the experiments we use two models to construct the sybil regions respectively: the preferential attachment (PA) model [4] and the Erdös-Rényi (ER) model [8]. Both models are widely used in network analysis. The networks constructed with the PA model are scale free, which means their node degrees follow a power-law distribution, a well accepted property of social networks [10], [15]. It has been used in previous research to build the sybil regions [6], [14]. The topologies built through the ER model, on the other hand, are random networks with no particular bias, which emulate the arbitrary structures of the sybil regions. In our experiments, to build a sybil region and connect it to a real-world social network sample, we follow the suggestion by Yu et al. [20] that the most effective way for an adversary to launch a sybil attack is to first compromise a small number of existing nodes, so as to quickly increase the number of attack edges. We first randomly select nodes from the data set to be compromised nodes, until the number of edges between the compromised nodes and the other nodes is g_0, which is the number of attack edges. The compromised nodes are all sybil nodes. They introduce γ additional sybil nodes, and establish a connected scale-free topology through the PA model, or a connected random topology through the ER model among all the sybil nodes. We label all the other nodes in the data set as honest nodes. The average degree of the sybil region built with the PA model is set to be equal to the average degree of the corresponding data set, while the average degree of the sybil region built with the ER model ranges from 8 to 11, representing a sparse topology compared with the realistic social networks. Note that in the evaluation of some previous schemes, the social network samples are first pre-processed by removing the nodes with small degrees [6], [19], to prevent such nodes from degrading the effectiveness of these schemes. Instead, we do not make any modification on the published data sets.

B. Evaluation of the Sybil Identification Algorithm

Yu et al. [19] proved that for all the sybil defense mechanisms that leverage the fast-mixing property, the number of admitted sybil nodes per attack edge is lower bounded by $\Omega(1)$. In this subsection we will show that the performance of our sybil identification algorithm approaches this theoretical bound, and our algorithm outperforms SybilLimit by one to two orders of magnitude in both accuracy and running time.

The intuition of our sybil identification algorithm is that, because of the existence of a small cut between the honest region and the sybil region, there is a difference between the coverage of random walks originating from an honest node and from a sybil node. Figure 3 illustrates this difference. Here, we use the PA model to construct the sybil region. We set the size of the sybil region to be 10000 nodes, and the number of attack edges to be 1000. In the experiments we perform 1000 random walks originating from each randomly selected source node. The upper curve in Figure 3 is the number of nodes traversed by random walks originating from an honest node no smaller than 5 times, while the lower curve is the number of nodes traversed by random walks originating from a sybil node no smaller than 5 times. Each point in the curves represents the mean value of 20 experiments. It is easy to see that the difference is larger than 200,000 nodes when the random walk length reaches 10000 hops. As described in Algorithm 2, we use $\tau = mean - \alpha * stdDeviation$ as the threshold to identify sybil nodes. In our experiments we observe that $stdDeviation < 1500$, so the sybil nodes can be identified even with a relatively large α, to limit the number of falsely identified honest nodes.

To evaluate our sybil identification algorithm, the parameters we used in the experiments are as follows: $l_{min} = 100$, $l_{max} = 10000$, $l_0 = 1000$, $t = 5$, $\alpha = 20$, $l_s = 20$, $f = 100$, $R \in \{1000, 1500, 2000\}$. When building the sybil regions, we set the number of attack edges to be 1000. We define the *false positive rate* as the percentage of the honest nodes identified to be sybil, and the *false negative rate* as the percentage of the sybil nodes identified to be honest. In the experiments we obtain the false positive and negative rates of our algorithm. As we use large-scale topologies in the experiments, it is infeasible to examine all the honest nodes to get the exact false positive rate. To estimate the false positive rate of the algorithm, in each experiment we randomly select 1000 honest nodes as suspects and use our sybil identification algorithm to test them. To get the false negative rate, in each experiment we use our algorithm to test every sybil node. In the experiments we vary the number of sybil nodes per attack edge. For each value we evaluate the algorithm on two real-world topologies, using two sybil region construction models, and with three values of R, the number of random walks performed in the algorithm, respectively.

Table II shows the results when each attack edge introduces 10 sybil nodes and 5 sybil nodes, respectively. It is easy to see that our algorithm achieves very low false positive and negative rates in all the cases. We find that all the sybil nodes that cannot be correctly identified are compromised nodes, as they are on the small cut between the honest region and the sybil region. Similarly, all the falsely identified honest nodes are close to the small cut.

Table III shows the results when each attack edge introduces only one sybil node. The false negative rate for the Facebook data set is higher than the results shown in Table II. This is because the difference between the coverage of the random walks originating from an honest node and from a sybil node becomes smaller compared with the cases when each attack

TABLE II
FALSE POSITIVE AND NEGATIVE RATES OF THE SYBIL IDENTIFICATION ALGORITHM

	10 sybil nodes per attack edge (10000 sybil nodes)								5 sybil nodes per attack edge (5000 sybil nodes)							
	Orkut				Facebook				Orkut				Facebook			
	PA Model		ER Model		PA Model		ER Model		PA Model		ER Model		PA Model		ER Model	
	F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-
1000RWs	0	0	0	0.11%	0	0.07%	0.1%	0.16%	0	0.02%	0	0.28%	0	0.22%	0.1%	0.54%
1500RWs	0	0.01%	0	0.11%	0.4%	0.08%	0.2%	0.1%	0	0.02%	0	0.32%	0.3%	0.12%	0.2%	0.44%
2000RWs	0	0	0	0.04%	0.3%	0.1%	0.5%	0.1%	0	0	0	0.22%	0.5%	0.04%	0.5%	0.4%

TABLE III
1 SYBIL NODE PER ATTACK EDGE FALSE POSITIVE AND NEGATIVE RATES

	Orkut		Facebook	
	PA Model		PA Model	
	F^+	F^-	F^+	F^-
1000RWs	0	0.6%	0%	6.2%
1500RWs	0	0.7%	0.4%	4.4%
2000RWs	0	0.2%	0%	1.4%

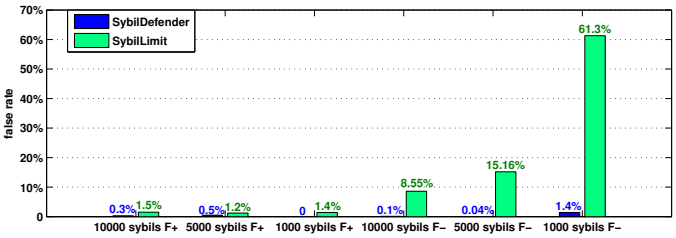

Fig. 4. Comparison between the false positive and negative rates of SybilDefender and those of SybilLimit

TABLE IV
FALSE RATES OF SYBILLIMIT ON THE FACEBOOK DATA SET

	PA Model		ER Model	
	F^+	F^-	F^+	F^-
10000 sybils	1.5%	8.55%	0.6%	15.35%
5000 sybils	1.2%	15.16%	1.4%	32.62%
1000 sybils	1.4%	61.3%	0.8%	85.3%

edge introduces more sybil nodes. The experimental results show that our sybil identification algorithm can identify nearly all the sybil nodes when each attack edge introduces 10 or 5 sybil nodes, and an overwhelming majority of sybil nodes when each attack edge introduces 1 sybil node, both with very low false positive rate.

1) Comparison with existing schemes: We fully implemented SybilLimit and evaluated it using our Facebook data set. Following the method in [19], we found the optimal parameters for SybilLimit on the Facebook data set. We set w, the length of random routes, to be 20 hops, and r, the number of instances of the random route generation protocol, to be 10000. Table IV lists SybilLimit's false positive and negative rates when each attack edge introduces 10 sybil nodes, 5 sybil nodes, and 1 sybil node, respectively. The results show that when each attack edge introduces one sybil node, SybilLimit accepts the majority of the sybil nodes.

Figure 4 compares the false positive and negative rates of SybilDefender with those of SybilLimit, when the sybil region is built with the PA model. It is easy to see that in all the three cases the false positive rate of SybilDefender is lower than that of SybilLimit, and the false negative rate of SybilDefender is lower than that of SybilLimit by one to two orders of magnitude. The reason is SybilLimit assumes that almost all the short random routes originating from an honest node will stay within the honest region, and it bounds the number of admitted sybil nodes by the number of attack edges and random route length. When each attack edge introduces few sybil nodes, SybilLimit cannot effectively identify the sybil nodes. On the other hand, SybilDefender interprets the small cut between the honest region and the sybil region as a bias in the coverage of the random walks originating from an honest node and from a sybil node. It can effectively identify the sybil nodes even when the number of sybil nodes introduced by each attack edge approaches the theoretical lower bound.

On one core of an Intel Xeon 2.93GHz processor, the average running time to test one sybil node by SybilDefender, with $R = 2000$, is 0.87 seconds, comparing to 11.56 seconds by SybilLimit. The average running time to test one honest node by SybilDefender is 7.11 seconds, comparing to 83.55 seconds by SybilLimit. The reason why SybilDefender is faster than SybilLimit by more than 10 times is that SybilLimit invokes a large number ($r = 10000$ for our Facebook data set) of instances of the random route generation protocol [19]. Within each instance a random routing table is generated for every node in the social graph. By contrast, SybilDefender only relies on performing a limited number of random walks.

Viswanath et al. proposed using a community detection algorithm [14] as the ranking algorithm to investigate the similarity between different sybil defense schemes. We evaluated their algorithm using our two data sets, and found that the algorithm alone cannot be used to identify the sybil nodes. The reason is that the algorithm starts from an honest node and iteratively adds nodes that improves the normalized conductance at each step. In our evaluation the normalized conductance always reaches the first inflection point after adding only several honest nodes. As a result, their algorithm cannot distinguish the sybil nodes from the honest nodes without providing a cutoff point.

We also evaluated Gatekeeper [12] using our data sets, which heavily relies on the assumption that the social networks are random expander. This assumption is stronger than our fast-mixing assumption and has not been validated in previous research, which makes Gatekeeper suffer from high false positive and negative rates on the real-world social topologies that exhibit asymmetries. For example, on our Facebook data

TABLE V
PERFORMANCE OF THE SYBIL COMMUNITY DETECTION ALGORITHM

10 sybil nodes per attack edge (10000 sybils)				
	Percentage of found sybil nodes		Number of falsely detected honest nodes	
	Orkut	Facebook	Orkut	Facebook
PA model	99.91%	99.82%	0.3	0.3
ER model	99.85%	99.84%	0	0.7
1 sybil node per attack edge (1000 sybils)				
	Percentage of found sybil nodes		Number of falsely detected honest nodes	
	Orkut	Facebook	Orkut	Facebook
PA model	99.4%	98.4%	0.1	1.1
ER model	98.7%	98.3%	0.1	0.3

set with a 10000-node sybil region built through the PA model, the average false positive rate of Gatekeeper is 11.7%, and the average false negative rate is 17.2%. When the sybil region is built with the ER model, the average false positive rate is 11.7%, and the average false negative rate is 14.7%. In the evaluation we used the parameters ($m = 100, f_{admit} = 0.2$) recommended by [12] and repeated each experiment 20 times.

C. Evaluation of the Sybil Community Detection Algorithm

To evaluate our sybil community detection algorithm, the parameters we used in the experiments are as follows: $l_0 = 100$, $\beta = 0.95$, $R = 2000$. We test the algorithm on two social topologies, with the sybil region built through two models, respectively. The number of attack edges is 1000, and the size of the sybil region depends on how many sybil nodes are introduced by each attack edge. As the goal of our sybil community detection algorithm is to detect the sybil community surrounding a known sybil node, when running each experiment we randomly select a sybil node as the starting node of our algorithm, and we get the percentage of the sybil nodes that can be detected, as well as the number of the honest nodes that are falsely detected. We repeat each experiment 20 times and calculate the mean value. Table V shows the results when each attack edge introduces 10 sybil nodes and 1 sybil node, respectively. It is easy to see that our algorithm can detect an overwhelming majority of the sybil region in all the experiments, and on average less than 1 honest node is falsely detected in each experiment. The undetected sybil nodes are all compromised nodes, the sybil nodes directly connecting to the honest nodes through attack edges. Our sybil community detection algorithm achieves high accuracy with short running time. For example, it takes 16 seconds to detect an 10000-node sybil region connecting to the Facebook data set on one core of an Intel Xeon 2.93GHz processor.

D. Survey Results of the Facebook Application

To investigate the user experience of our Facebook application, we carried out a survey through the Amazon Mechanical Turk platform [1]. In the survey we asked the respondents to use our application to rate all the relations in their Facebook friend lists. We get the results from 214 respondents. Their average number of relations is 118, and the average time to finish the survey is 249 seconds. This indicates that it does not take long for the online social network users to rate their relationships. The average percentage of strangers among all the relations is 19.8%, which shows that the assumption made by previous work that all the links in social networks are trusted does not apply to online social networks. In the survey 76.6% of the respondents would like to rate their relationships when they know this can help to defend against malicious users. This shows that relationship rating is a promising way to limit the capacity of the adversary to create attack edges.

VI. CONCLUSION

We present SybilDefender, a scheme that leverages the network topologies to defend against sybil attacks in large social networks. Our evaluation shows that SybilDefender can correctly identify the sybil nodes even when the number of sybil nodes introduced by each attack edge approaches the theoretically detectable lower bound, and it can effectively detect the sybil community surrounding a sybil node with different sizes and structures.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-1117412 and CAREER Award CNS-0747108.

REFERENCES

[1] Amazon mechanical turk. https://www.mturk.com/mturk.
[2] Rate your relationships. http://apps.facebook.com/ratingrelationships/.
[3] The top 500 sites on the web. http://www.alexa.com/topsites.
[4] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys*, 74:47–97, 2002.
[5] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW*, 2009.
[6] G. Danezis and P. Mit. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.
[7] J. R. Douceur. The sybil attack. In *IPTPS*, 2002.
[8] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathemticae (Debrecen)*, 6:290–297, 1959.
[9] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. In *FOCS*, 2000.
[10] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *ACM/USENIX IMC*, 2007.
[11] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.
[12] N. Tran, J. Li, L. Subramanian, and S. S.M. Chow. Optimal sybil-resilient node admission control. In *IEEE INFOCOM*, 2011.
[13] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *SIGCOMM WOSN*, 2009.
[14] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *SIGCOMM*, 2010.
[15] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *EuroSys*, 2009.
[16] K. Xing and X. Cheng. From time domain to space domain: Detecting replica attacks in mobile ad hoc networks. In *IEEE INFOCOM*, 2010.
[17] K. Xing, F. Liu, X. Cheng, and D. H. Du. Realtime detection of clone attacks in wireless sensor networks. In *IEEE ICDCS*, 2008.
[18] L. Xu, S. Chainan, H. Takizawa, and H. Kobayashi. Resisting sybil attack by social network and network clustering. In *SAINT*, 2010.
[19] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, 2008.
[20] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM*, 2006.