SybilDefender: A Defense Mechanism for Sybil Attacks in Large Social Networks

Wei Wei, Fengyuan Xu, Chiu C. Tan, *Member*, *IEEE*, and Qun Li, *Senior Member*, *IEEE*

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 more than 10 times 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.

Index Terms—Sybil attack, social network, random walk

1 Introduction

Distributed systems are vulnerable to sybil attacks [1], [2], 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.

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 [3], [4], [5], [6], [7]. 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 nontrivial, 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 [8].

In the past few years, online social networks have gained great popularity and are among the most frequently visited sites on the web. 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 (2,000 nodes in [5] and 30,000 nodes in [3]), 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 [7] admits $O(\sqrt{n}\log n)$ sybil nodes per attack edge, where n is the number of honest nodes; SybilLimit [6] 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. In addition, 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 sybil nodes, a sybil community

- *W. Wei, F. Xu, and Q. Li are with the College of William & Mary, McGlothlin-Street Hall 126, Williamsburg, VA 23185. E-mail: {wwei, fxu, liqun}@cs.wm.edu.*
- *C.C. Tan is with Temple University, 1805 North Broad Street, Wachman Hall Room 313, Philadelphia, PA 19122. E-mail: cctan@temple.edu.*

Manuscript received 4 Feb. 2012; revised 26 Nov. 2012; accepted 17 Dec. 2012; published online 10 Jan. 2013.
Recommended for acceptance by K. Wu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2012-02-0082.
Digital Object Identifier no. 10.1109/TPDS.2013.9.

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, our proposed sybil identification and sybil community detection algorithms are more efficient than previous techniques for 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 more than 10 times in both accuracy and running time. In addition, 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.

A preliminary version of this paper was presented in [9]. Herein, we design a Combo algorithm that combines the sybil identification algorithm with the sybil community detection algorithm, which reduces a large portion of computation overhead of the latter. We also present the weighted versions of the sybil identification and sybil community detection algorithms, and evaluate their performance on a weighted social network sample. The behavior of the sybil identification algorithm over several designs of trust-driven random walks is analyzed in the evaluation section. Additionally, we add more evaluation results of our algorithms, such as their performance over much larger sybil regions, as well as the detailed survey results of our Facebook application.

2 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 [7] and SybilLimit [6], to determine whether a suspect node is sybil. 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 leverage random routes, a special kind of random walks in which each node uses a precomputed random permutation as a one-to-one mapping from incoming edges to outgoing edges. SybilGuard 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 $\Omega(1)$ [6] 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 [4] is another decentralized sybil defense scheme that heavily relies on the assumption that the social networks are random expander. This is a strong assumption that 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 [3], 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 30,000 nodes, which is much smaller than the size of regular online 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.

Some previous solutions mitigate sybil attacks through the use of computational puzzles or CAPTCHAs [10]. These approaches can limit the rate at which sybil identities are introduced into the systems, but they cannot identify the existing sybil identities. They can be used in conjunction with the scheme proposed in this paper.

3 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 on 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 [7]. Generally speaking, random walks in a fast mixing

graph converge quickly to the stationary distribution. 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. At each step of the random walk, the transition probability from node i to j is $P_{ij} = \frac{A_{ij}}{d_i}$, where d_i is the degree of node i. $A_{ij} = 1$ if i and j are connected, otherwise $A_{ij} = 0$. 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. [6] have shown that the real-world social networks are fast mixing. The previous sybil defense schemes [3], [6], [7] are also built upon this assumption.

One known honest node. Like previous schemes [3], [6], [7], 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 [3], [4], [6], [7], which has been shown to not hold in online social networks. The experiment by Bilge et al. [8] shows that on Facebook, the acceptance rate of friendship requests from a bogus account is around 20 percent. If an adversary launches a sybil attack, all the links created in this way are attack edges. We will address this problem in Section 4.4.

4 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 4.1 is to determine whether a suspect node is sybil. Then, we show how to efficiently detect the sybil community around a sybil node with our sybil community detection algorithm presented in Section 4.2. 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. In Section 4.3, we present a Combo algorithm that combines the sybil identification algorithm with the sybil community detection algorithm. Finally, both algorithms are built upon the assumption that the number of attack edges is limited. In Section 4.4, we propose two approaches to supporting this assumption in online social networks.

4.1 Sybil Identification Algorithm

In this section, 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 input, 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, because 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 to exhibit the difference between the sybil region and the honest region, 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, Algorithms 1 and 2. The first phase takes G and h as input, 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 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 [11], 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 1,000 to 2,000 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 (five 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 $\langle l, mean, stdDeviation \rangle$.

Algorithm 1. PreProcessing(G, h).

1: $J = \{h\}$
2: **for** $i = 1$ to f **do**
3: Perform a random walk with length $l_s = \log n$ originating from h

4: $J = J \cup \{the\ ending\ node\ of\ the\ random\ walk\}$
5: **end for**
6: $l = l_{min}$
7: **while** $l <= l_{max}$ **do**
8: **for** $i = J.first()$ to $J.last()$ **do**
9: Perform R random walks with length l originating from node i
10: Get n_i as the number of nodes with frequency no smaller than t
11: **end for**
12: output $\langle l, mean(\{n_i : i \in J\}), stdDeviation(\{n_i : i \in J\})\rangle$
13: $l = l + 100$
14: **end while**

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 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.

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

1: $l = l_0$
2: **while** $l <= l_{max}$ **do**
3: Perform R random walks with length l originating from u
4: $m =$ the number of nodes whose frequency is no smaller than t
5: Let the tuple corresponding to length l in the outputs of Algorithm 1 be $\langle l, mean, stdDeviation\rangle$
6: **if** $mean - m > stdDeviation * \alpha$ **then**
7: output u is sybil
8: end the algorithm
9: **end if**
10: $l = l * 2$
11: **end while**
12: output u is honest

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, because the footprint of short random walks in a large sybil region may be similar to that in the honest region. A theoretical analysis of our sybil identification algorithm is provided in Appendix A in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.9.

4.2 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 input, 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, because 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 is in.

Our algorithm relies on performing *partial* random walks originating from s. Each partial random walk behaves the same as the *simple* random walks used in the previous section, 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 simple random walk, because 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.

The sybil community detection algorithm consists of two phases, Algorithms 3 and 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 (2,000 in our evaluation).

Algorithm 3. WalkLengthEstimation(G, s).

$l = l_0/2$
$deadWalkRatio = 0$
while $deadWalkRatio < \beta$ **do**
  $l = l * 2$
  $deadWalkNum = 0$
  for $i = 1$ to R **do**
    Perform a partial random walk originating from s with length l
    if the partial random walk is dead before it reaches l hops **then**
      $deadWalkNum$++
    end if
  end for
  $deadWalkRatio = deadWalkNum/R$
end while
output l

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
$counter = 0$
$S = \emptyset$
do
  $counter =$ conductance(S)
  for $i = traversedList.first()$ to $traversedList.last()$ **do**
    if node $i \in S$ **then**
      continue
    if conductance($\{i\} \cup S$) <= conductance(S) **then**
      $S = \{i\} \cup S$
while ($counter >$ conductance(S))
output S

Algorithm 4 takes G, s, and the estimated length l as input 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 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* [12], 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, because a large number of partial random walks cannot enter the honest region, due to the existence of the small cut between the honest region and the sybil 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.

4.3 Combine Sybil Identification with Sybil Community Detection

Our sybil identification algorithm takes as input a suspect node, and outputs if the suspect node is sybil. In comparison, our sybil community detection algorithm takes as input a sybil node, and outputs the sybil community surrounding the sybil node. Each algorithm consists of a preparation phase and a detection phase. In this section, we consider how to efficiently combine the two algorithms, such that by running the Combo algorithm once, the administrator is able to learn if the suspect node is sybil, and if it is, the sybil community around it.

The Combo algorithm consists of two phases, Algorithms 1 and 5. Algorithm 1 only needs to be run once for each target social graph. Algorithm 5 takes as input a suspect node u, the tuples from Algorithm 1, and the social graph G, and it outputs u's identity (sybil or honest) as well as the sybil community surrounding u (if sybil). The intuition of the Combo algorithm is that, if we find a sybil node, instead of performing partial random walks as in Section 4.2 to detect the surrounding sybil community, we directly analyze the simple random walks derived in the identification method. The analysis is based on the conductance measure. We sort the nodes by their frequency, the number of times being traversed by the simple random walks, in decreasing order, and iteratively add nodes to the detected sybil set, until the conductance value stays the same at the end of two consecutive iterations.

Algorithm 5. Combo($G, u, tuples$ *from Alg.*1).

$l = l_0$
while $l <= l_{max}$ **do**

```
3:    Perform R random walks with length l originating from u
4:    m = the number of nodes whose frequency is no smaller than t
5:    Let the tuple corresponding to length l in the outputs of Algorithm 1 be ⟨l, mean, stdDeviation⟩
6:    if mean − m > stdDeviation ∗ α then
7:       output u is sybil
8:       traversedList = Sort the traversed nodes by their frequency in decreasing order
9:       counter = 0
10:      S = ∅
11:      do
12:         counter = conductance(S)
13:         for i = traversedList.first() to traversedList.last() do
14:            if node i ∈ S then
15:               continue
16:            if conductance({i} ∪ S) <= conductance(S) then
17:               S = {i} ∪ S
18:      while (counter > conductance(S))
19:      output S
20:      end the algorithm
21:   end if
22:   l = l ∗ 2
23: end while
24: output u is honest
```

It is easy to see that the Combo algorithm behaves the same as the sybil identification algorithm (Section 4.1) when identifying sybil nodes, while it diverges from the sybil community detection algorithm (Section 4.2) in that it reuses the simple random walks performed in the identification phase to search for the sybil community. The advantage is that it avoids estimating the partial random walk length (Algorithm 3) and performing partial random walks (Algorithm 4), and thus incurs much smaller computation overhead. However, compared with partial random walks, the simple random walks originating from a sybil node are more likely to escape the sybil region. Our evaluation in Section 5 shows that replacing partial random walks with simple random walks slightly impact detection accuracy, while it significantly reduces running time of the algorithm. Therefore, the Combo algorithm provides a tradeoff between efficiency and accuracy: to detect sybil nodes, users can use the Combo algorithm if running time is a concern. Otherwise, they can use the stand-alone sybil identification and sybil community detection algorithms if detection accuracy is more important.

4.4 Limiting the Number of Attack Edges

Our algorithms rely on the assumption that the number of attack edges is limited. However, it has been shown that not all the relationships in online social networks are trusted [8]. In this section, we propose two approaches to limiting the number of attack edges in online social networks.

4.4.1 Relationship Rating

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*. 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. In this way, the user's relations are classified into two categories. 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 Appendix F, available in the online supplemental material.

4.4.2 Activity Network

We can also use the concept of *activity network* [13], [14] 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. It contains all the nodes from its social network counterpart, but only a subset of edges. 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. In other words, a social network is transformed into an activity network by removing the weak connections with no user activity. 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. Notice that using activity network only raises the difficulty for the attackers to create many attack edges, but it cannot fundamentally prevent attackers from generating activity links between sybil nodes and honest nodes, as shown by Boshmaf et al. [15] through the use of socialbot networks.

5 Evaluation

5.1 Data Sets and Experiment Setup

In this section, we evaluate the effectiveness of SybilDefender using two data sets [14], [16] 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. The reason why the average degree of the Facebook data set is much smaller than the Orkut data set is that the Orkut data set is a breadth-first sample of the Orkut social graph, which maintains the topological properties of Orkut like average degree; on the other hand, the Facebook data set is a regional network in Facebook. Two nodes share an edge in this data set if and only if both of them are members of the same regional network, and they are Facebook friends with each other. By evaluating the performance of SybilDefender on these two data sets, we show that SybilDefender applies to social networks with different topological properties.

TABLE 1
False Positive and Negative Rates of the Sybil Identification Algorithm (1,000 Attack Edges)

	10 sybil nodes per attack edge (10000 sybils)								5 sybil nodes per attack edge (5000 sybils)							
	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%

In the experiments, we used the preferential attachment (PA) model [17] and the Erdös-Rényi (ER) model [18] to construct sybil regions. 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 [14], [16], [19]. The topologies built through the ER model, on the other hand, are random networks with no particular bias, which emulate the arbitrary structures of sybil regions that may be created by attackers. Both the PA and the ER models have been used in previous research to build sybil regions [3], [5], [20]. 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. [7] 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, i.e., we do not consider other types of malicious users in the real data sets. 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 realistic social networks. Note that in the evaluation of some previous schemes, the social network samples are first preprocessed by removing nodes with small degrees [3], [5], [6], to prevent such nodes from degrading the effectiveness of these schemes. Instead, we did not make any modification on the published data sets.

Besides the PA and ER models, in our experiments we also tried other graph construction models to build sybil regions, including Kleinberg's synthetic social network model [21], the Forest Fire model [19], the Random Walk model [19], the Nearest Neighbor model [19], and the transitive linking model [22]. The evaluation results obtained by using these models are very similar to those obtained by using the PA and ER models. We only present the results of the PA and ER models for brevity. We admit that to the best of our knowledge, there is no real-world evidence that the sybil regions are created following a particular model, as the adversary can be arbitrarily malicious. Thus, to address this limitation, we evaluated our algorithms with a variety of popular graph models, and present the representative results.

5.2 Evaluation of the Sybil Identification Algorithm

Yu et al. [6] 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 $\Omega(1)$. The rationale is that the fasting mixing property of the network is not disrupted if each attack edge introduces few sybil nodes. In this section, we will show that the performance of our sybil identification algorithm approaches this theoretical bound, and our algorithm outperforms SybilLimit by more than 10 times in both accuracy and running time. An experimental verification of the intuition of our sybil identification algorithm is presented in Appendix B, available in the online supplemental material.

To evaluate our sybil identification algorithm, the parameters we used in the experiments are as follows: $l_{min} = 100$, $l_{max} = 10{,}000$, $l_0 = 1{,}000$, $t = 5$, $\alpha = 20$, $l_s = 20$, $f = 100$, $R \in \{1{,}000, 1{,}500, 2{,}000\}$. When building the sybil regions, we set the number of attack edges to be 1,000. 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 our algorithm, in each experiment we randomly select 1,000 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 1 shows the results when each attack edge introduces 10 sybil nodes. 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 1 also shows the results when each attack edge introduces five sybil nodes. We observe that with the increase of the number of random walks performed in the algorithm, the false positive rate raises, while the false negative rate decreases. The reason is

TABLE 2
False Positive and Negative Rates (1,000 Attack Edges, One Sybil Node per Attack Edge)

	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%

that the more random walks are performed, the smaller the standard deviation of the number of nodes whose frequency, the number of times being traversed, is no smaller than t is. As a result, the threshold τ increases when more random walks are performed, which makes it less likely for a sybil node to be identified as honest, and vice versa for honest nodes.

Table 2 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 1. 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 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 five sybil nodes, and an overwhelming majority of sybil nodes when each attack edge introduces one sybil node, both with very low false positive rate. More evaluation results of our algorithm are provided in Appendix C, available in the online supplemental material, and in Appendix D, available in the online supplemental material, we compare the performance of our algorithm with other sybil defense schemes.

5.2.1 Evaluation of the Sybil Identification Algorithm on a Weighted Social Network

As mentioned in Section 4.4, activity network is a social network model that takes the activities between users into account. Similarly, given the user activity information, a social network can be transformed into a weighted graph, by assigning a weight to each edge based on the number of interactions between the two end nodes. There are also other forms of weighted social networks, for example, the trust networks whose edge weights denote the level of trust [23], and the networks of coauthorship, where an edge weight is the number of papers coauthored by the two end users [24].

To investigate the performance of our sybil identification algorithm on such weighted social networks, we use the data set provided by Wilson et al. [14]. The data set is an undirected, weighted graph. It has the same set of nodes and the same topology as the Facebook data set used in our previous evaluations. However, each edge in the graph is assigned a weight, based on the number of interactions (wall posts and photo comments) between the two end nodes. For example, assuming user i and user j are friends in the data set, if the number of interactions between i and j is 10, then the weight of edge $\overline{ij}$ is 10. Otherwise, if there is no interaction between i and j ever, then the weight is 0. In total, this weighted graph records 17,644,327 interactions between 3,097,165 users, which means that on average each user takes part in 11.4 interactions.

TABLE 3
False Rates of the Sybil Identification Algorithm on a Weighted Social Network

10000 sybils				5000 sybils			
PA Model		ER Model		PA Model		ER Model	
F^+	F^-	F^+	F^-	F^+	F^-	F^+	F^-
0.7%	0.15%	0.5%	0.21%	0.7%	0.20%	0.7%	0.56%

In our sybil identification algorithm, at each intermediate node of a random walk, all its neighbors are treated equally when selecting the next hop. To extend this algorithm to weighted social networks, we take the edge weights into account when choosing the next hop, and define *weighted* random walks. At each step of a weighted random walk, the transition probability from node i to j is $P_{ij} = \frac{A_{ij}(1+w_{ij})}{d_i+w_i}$, where w_{ij} is the weight of edge $\overline{ij}$, d_i is the degree of node i, and w_i is the sum of weights of all the edges connecting node i. $A_{ij} = 1$ if i and j are connected, otherwise $A_{ij} = 0$. As a result, the higher weight an edge has, with higher possibility the weighted random walk will traverse that edge. The modified sybil identification algorithm runs by performing weighted random walks.

Table 3 shows the false rates of the modified sybil identification algorithm on the weighted social network sample. The number of attack edges is 1,000, $R = 2{,}000$. The other parameters used in the algorithm are equal to the ones used in previous evaluations. When constructing the sybil regions, we randomly assign weights to the edges connecting sybil nodes, such that the average weight is equal to the average weight of our data set. The results illustrate that our algorithm achieves similar performance on the weighted graph: both false positive and negative rates remain low.

5.2.2 Evaluation of Trust-Driven Random Walks

Mohaisen et al. [25] proposed several designs of random walks that incorporate trust between nodes in social graphs, and studied their impact on the performance of SybilLimit. Their findings suggest that these modified and biased random walks model trust and influence SybilLimit's performance differently. In this section, we evaluate the performance of our sybil identification algorithm operating with these trust-driven random walks on our Facebook data set. In the experiments, the number of attack edges is fixed at 1,000, $R = 2{,}000$, and the sybil region is built with the PA model.

For lazy random walks, the transition probability form node i to node j is $\frac{1-\alpha}{d_i}$ if j is a neighbor of i, α if $i = j$, and 0 otherwise, where α is a parameter characterizing the trust level, and d_i is the degree of node i. The false rates of the sybil identification algorithm over lazy random walks ($\alpha = 0.2$) is shown in Table 4, which are similar to the results of simple random walks presented in Table 1. The reason is that by capturing the random walk in the current node with probability α, the laziness actually reduces the effective length of random walks by α, which does not degrade performance given a small α.

TABLE 4
False Rates of the Sybil Identification Algorithm Operating with Trust-Driven Random Walks

	lazy random walk			similarity-based random walk		
# of sybils	10000	5000	1000	10000	5000	1000
F^+	0	0.1%	0.3%	0.4%	0.4%	0.5%
F^-	0.13%	0.34%	2.6%	0.17%	0.42%	1.7%

At each step of an originator-biased random walk, the transition probability from the current node to the originator of the random walk is α, and with probability $1-\alpha$ the next hop is chosen uniformly among the neighbors of the current node. In the experiments, we found that when operating with originator-biased random walks, our sybil identification algorithm cannot effectively identify sybil nodes. The coverage of the originator-biased random walks are severely limited by their inherent "discontinuity": At each step the random walk is moving back to the originator with probability α. Since our sybil identification algorithm is built upon the intuition that the coverage of random walks starting from an honest node is larger than the random walks starting from a sybil node, the originator-biased random walks do not fit our algorithm.

For two nodes i and j and their sets of neighbors N_i and N_j, the similarity between i and j is defined as $S_{ij} = \frac{N_i \cap N_j}{N_i \cup N_i}$. At each step of a similarity-based random walk, the transition probability from node i to one of its neighbors j is

$$\frac{S_{ij}+1}{\sum_{k \in N_i}(S_{ik}+1)},$$

i.e., the more similar two adjacent nodes are, with higher possibility a similarity-based random walk will traverse the link connecting the two nodes. Table 4 shows the false rates of the sybil identification algorithm operating with similarity-based random walks. The results are comparable to those obtained over the weighted social network (Table 3). This is because like the weighted random walks are biased based on link weights, the similarity-based random walks are biased according to the similarity between each pair of adjacent nodes.

5.3 Evaluation of the Sybil Community Detection Algorithm

To evaluate our sybil community detection algorithm, the parameters we used in the experiments are as follows: $l_0 = 100$, $\beta = 0.95$, $R = 2{,}000$. We test the algorithm on two social topologies, with the sybil region built through two models, respectively. The number of attack edges is 1,000, 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 5
Accuracy of the Sybil Community Detection Algorithm ($1{,}000$ Attack Edges)

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
5 sybil nodes per attack edge (5000 sybils)				
	Percentage of found sybil nodes		Number of falsely detected honest nodes	
	Orkut	Facebook	Orkut	Facebook
PA model	99.86%	99.66%	0.1	0.8
ER model	99.74%	99.66%	0.1	0.2
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

Table 5 shows the results when each attack edge introduces 10 sybil nodes, five sybil nodes, and one 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 the numbers of falsely detected honest nodes are very small: on average less than one 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. More evaluation results of our algorithm, including its accuracy over much larger regions and the accuracy of a weighted version of the algorithm, are provided in Appendix E, available in the online supplemental material.

5.4 Evaluation of the Combo Algorithm

In this section, we evaluate the performance of the Combo algorithm presented in Section 4.3. Since the Combo algorithm behaves the same as the sybil identification algorithm when identifying sybil nodes, we measure the running time and accuracy of the Combo algorithm after a sybil node has been found and the algorithm is used to detect the sybil community surrounding the sybil node, and compare it with our sybil community detection algorithm.

In the experiments, we fixed the number of attack edges at 1,000, and evaluated the Combo algorithm when the number of sybil nodes in the sybil region is 10,000, 5,000, and 1,000, respectively. The sybil region is constructed with the PA model. As mentioned in Section 4.3, the advantage of the Combo algorithm over the stand-alone sybil community detection algorithm is that the former reuses the simple random walks performed in the identification method, and thus avoids estimating partial random walk length and performing partial random walks, which significantly reduces computation overhead. For instance, on a single core of an Intel Xeon 2.93-GHz processor, the running time for the Combo algorithm to detect a 1,00,000-node sybil region constructed with the PA model and connecting to the Orkut graph is 91 seconds, while for the stand-alone sybil community detection algorithm it is 257 seconds, almost three times longer. Table 6 shows the accuracy of the Combo algorithm for

TABLE 6
Accuracy of the Combo Algorithm

	Percentage of found sybil nodes		Number of falsely detected honest nodes	
	Orkut	Facebook	Orkut	Facebook
10000 sybils	99.82%	99.64%	0	0.4
5000 sybils	99.68%	99.40%	0.2	1.0
1000 sybils	98.4%	96.5%	0.2	1.2

sybil community detection. All the results are averaged over 20 runs. Compared with the accuracy of the sybil community detection algorithm in Table 5, the detection rate is slightly lower. This is because the Combo algorithm detects the sybil community by analyzing simple random walks originating from a sybil node, instead of partial random walks that are more likely to be trapped within the sybil region.

6 Conclusion

In this paper, we present SybilDefender, a scheme that leverages network topologies to defend against sybil attacks in large social networks. SybilDefender consists of a sybil identification algorithm, a sybil community detection algorithm, and two approaches to limiting the number of attack edges in online social networks. Our evaluation on two large-scale real-world social network samples shows that SybilDefender can correctly identify sybil nodes, even when the number of sybil nodes introduced by each attack edge approaches the theoretically detectable lower bound, and that it can effectively detect the sybil community surrounding a sybil node with different sizes and structures.

References

[1] J.R. Douceur, "The Sybil Attack," *Proc. Revised Papers First Int'l Workshop Peer-to-Peer Systems (IPTPS '01)*, 2002.

[2] E. Novak and Q. Li, "A Survey of Security and Privacy Research in Online Social Networks," Technical Report WM-CS-2012-2, College of William and Mary, 2012.

[3] G. Danezis and P. Mit, "Sybilinfer: Detecting Sybil Nodes Using Social Networks," *Proc. Network and Distributed System Security Symp. (NDSS)*, 2009.

[4] N. Tran, J. Li, L. Subramanian, and S.S. Chow, "Optimal Sybil-Resilient Node Admission Control," *Proc. IEEE INFOCOM*, 2011.

[5] L. Xu, S. Chainan, H. Takizawa, and H. Kobayashi, "Resisting Sybil Attack by Social Network and Network Clustering," *Proc. IEEE/IPSJ 10th Int'l Symp. Applications and Internet (SAINT)*, 2010.

[6] H. Yu, P.B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," *Proc. IEEE Symp. Security and Privacy*, 2008.

[7] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman, "SybilGuard: Defending against Sybil Attacks via Social Networks," *Proc. ACM SIGCOMM*, 2006.

[8] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks," *Proc. 18th Int'l Conf. World Wide Web (WWW '09)*, 2009.

[9] W. Wei, F. Xu, C.C. Tan, and Q. Li, "SybilDefender: Defend against Sybil Attacks in Large Social Networks," *Proc. IEEE INFOCOM*, 2012.

[10] R.S. Peterson and E.G. Sirer, "AntFarm: Efficient Content Distribution with Managed Swarms," *Proc. Networked Systems Design and Implementation (NSDI)*, 2009.

[11] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman, "Sybilguard: Defending against Sybil Attacks via Social Networks," Technical Report IRP-TR-06-01, Intel Research Pittsburgh, 2006.

[12] R. Kannan, S. Vempala, and A. Vetta, "On Clusterings: Good, Bad and Spectral," *Proc. 41st Ann. Symp. Foundations Computer Science (FOCS)*, 2000.

[13] B. Viswanath, A. Mislove, M. Cha, and K.P. Gummadi, "On the Evolution of User Interaction in Facebook," *Proc. Second ACM Workshop Online Social Networks (WOSN)*, 2009.

[14] C. Wilson, B. Boe, A. Sala, K.P.N. Puttaswamy, and B.Y. Zhao, "User Interactions in Social Networks and Their Implications," *Proc. Fourth ACM European Conf. Computer Systems (EuroSys)*, 2009.

[15] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The Socialbot Network: When Bots Socialize for Fame and Money," *Proc. 27th Ann. Computer Security Applications Conf. (ACSAC)*, 2011.

[16] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," *Proc. Seventh ACM SIGCOMM Conf. Internet Measurement (ACM/USENIX IMC)*, 2007.

[17] R. Albert and A. Barabási, "Statistical Mechanics of Complex Networks," *Rev. Modern Physics*, vol. 74, pp. 47-97, 2002.

[18] P. Erdös and A. Rényi, "On Random Graphs," *Publicationes Mathemticae (Debrecen)*, vol. 6, pp. 290-297, 1959.

[19] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B.Y. Zhao, "Measurement-Calibrated Graph Models for Social Network Experiments," *Proc. 19th Int'l Conf. World Wide Web (WWW '10)*, 2010.

[20] B. Viswanath, A. Post, K.P. Gummadi, and A. Mislove, "An Analysis of Social Network-Based Sybil Defenses," *Proc. ACM SIGCOMM*, 2010.

[21] J. Kleinberg, "The Small-World Phenomenon: An Algorithm Perspective," *Proc. 32nd Ann. ACM Symp. Theory Computing (STOC '00)*, 2000.

[22] J. Davidsen, H. Ebel, and S. Bornholdt, "Emergence of a Small World from Local Interactions: Modeling Acquaintance Networks," *Physical Rev. Letters*, vol. 88, 2002.

[23] "Trust Network Data Sets," http://www.trustlet.org/wiki/-Trust_network_datasets, 2013.

[24] M.E.J. Newman, "The Structure of Scientific Collaboration Networks," *Proc. Nat'l Academy Sciences USA*, vol. 98, no. 2, pp. 404-409, 2001.

[25] A. Mohaisen, N. Hopper, and Y. Kim, "Keep Your Friends Close: Incorporating Trust into Social Network-Based Sybil Defenses," *Proc. IEEE INFOCOM*, 2011.

Wei Wei received the BS and MS degrees in computer science from the Beihang University in 2005 and 2008, respectively. He is currently a graduate research assistant at the Department of Computer Science, College of William and Mary. His research interests include social networks and distributed systems.

Fengyuan Xu received the BS degree in computer science from the Jilin University in 2007. He is currently a graduate research assistant at the Department of Computer Science, College of William and Mary. His research interests include mobile computing, wireless networks, and cyber-physical systems.

Chiu C. Tan received the BS degree in computer science and the BA degree in economics (honors) from the University of Texas at Austin in 2004. He is currently an assistant professor at the Department of Computer and Information Sciences, Temple University. His research interests include ubiquitous computing, embedded systems, large-scale RFID systems, cyber-physical systems (CPS), security and privacy, and wireless networks. He is a member of the IEEE.

Qun Li received the PhD degree in computer science from Dartmouth College. He is an associate professor at the Department of Computer Science, College of William and Mary. His research interests include wireless networks, sensor networks, RFID, pervasive computing systems, smart grid, wireless health, and social networks. He received the US National Science Foundation (NSF) Career award in 2008. He is a senior member of the IEEE.