VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils

Jilong Xue∗, Zhi Yang∗, Xiaoyong Yang∗, Xiao Wang†, Lijiang Chen† and Yafei Dai∗
∗Computer Science Department, Peking University, Beijing, China
†HP labs, China
† Renren Inc.
{xjl, yangzhi, yxy, dyf}@net.pku.edu.cn, {lijiang.chen}@hp.com, xiao.wang@renren-inc.com

Abstract—Online social networks (OSNs) currently face a significant challenge by the existence and continuous creation of fake user accounts (Sybils), which can undermine the quality of social network service by introducing spam and manipulating online rating. Recently, there has been much excitement in the research community over exploiting social network structure to detect Sybils. However, they rely on the assumption that Sybils form a tight-knit community, which may not hold in real OSNs. In this paper, we present VoteTrust, a Sybil detection system that further leverages user interactions of initiating and accepting links. VoteTrust uses the techniques of trust-based vote assignment and global vote aggregation to evaluate the probability that the user is a Sybil. Using detailed evaluation on real social network (Renren), we show VoteTrust’s ability to prevent Sybils gathering victims (e.g., spam audience) by sending a large amount of unsolicited friend requests and befriending many normal users, and demonstrate it can significantly outperform traditional ranking systems (such as TrustRank or BadRank) in Sybil detection.

I. INTRODUCTION

Sybil attacks [1] are one of the most prevalent and practical attacks against distributed systems. In this attack, a malicious user creates multiple fake identities, known as Sybils, to unfairly increase their power and influence within a target community. To date, researchers have demonstrated the efficacy of Sybil attacks against P2P systems [2], anonymous communication networks [3], and sensor networks [4].

Recently, online social networks (OSNs) have also come under attack from Sybils. Researchers have observed Sybils forwarding spam and malware on Facebook [5] and Twitter [6]. Looking forward, Sybil attacks on OSNs are poised to become increasingly widespread and dangerous as more people come to rely on OSNs for basic online communication [7] and as replacements of news outlets [8].

To address the problem of Sybils on OSNs, researchers have developed social-graph-based algorithms such as SybilGuard [9], SybilLimit [10], SybilInfer [11], and SumUp [12] to perform detection of Sybils on social graphs. These systems detect Sybils by identifying tightly connected communities of Sybil nodes [13]. However, our recent measurement [14] in the Renren OSN show that Sybils do not form tight-knit communities, since they could befriend many non-Sybils by sending a large amount of unsolicited friend invitations.

Notice that these social-graph-based systems is insufficient and sometimes misleading in Sybil detection, we extend social graph by including user interactions of initiating and accepting links. We represent the friend request data as a directed graph, with an edge directed from the sender to the receiver and a weight (1/0) indicates whether this initiation is accepted. This graph is referred to as the friend invitation graph, as illustrated in Fig. 1.

This new graph model incorporates two information to perform an accurate detection: The first is requests to befriend, which are rarely sent to Sybils. Relying on this implicit user opinions reflected in the link initiations, we could adopt traditional ranking system (such as TrustRank [19] and BadRank [20]) to rank users according to their perceived likelihood of being fake (Sybils). However, only using such information is prone to low precision, as Sybils and non-popular users would be mixed together due to both of them received few requests(see Fig. 1). Interestingly, the friend invitation graph further contains the information of the accepting/rejecting friend request. Although Sybil and non-popular user rarely receive friend requests from others, both of them send friend requests to gain friends. However, Sybils’ requests are more like to be rejected (a negative feedback).

In this paper, we present VoteTrust, a global voting-based system that nicely combine link structure and users feedback (accept or reject friend requests) to detect Sybils. In VoteTrust, if a node A sends a friend request to node B, we say that B casts a (positive/negative) vote on A if B accepts/rejects the request. VoteTrust first uses a PageRank-style algorithm to appropriately assign the number of votes each user can
cast (referred to as vote capacity). This process assigns little vote capacity for each Sybil and thus prevents them from significantly vouching each other through collusion. Then, VoteTrust evaluates a global rating (i.e., the probability of being a Sybil) for each node through aggregating the votes all over the network. Notice that the rating of a node actually represents the value of its votes, as the votes of low-rating nodes are also unlikely to be reliable. Hence, VoteTrust bias towards the voters of more number of votes and high rating when aggregating the votes for a node.

In summary, our contributions include the following:

- First, we present VoteTrust, a voting system that leverages user activities of initiating and accepting links to defend against Sybils;
- Second, we give an analysis on security properties of VoteTrust, proving that it could provide bounds on the number of friend requests individual Sybils could send to normal users, and the size of Sybil community.
- Finally, Our evaluation over real network shows that VoteTrust can detect real Sybils with high precision, and significantly outperform traditional ranking systems (including TrustRank and BadRank).

In the rest of the paper, we give a detailed description of our system in Section II. In Section III, we provide a formal analysis of the security properties of VoteTrust. Then we evaluate and validate its performance in Section IV. Finally we discuss related work in Section V and conclude in Section VI.

II. VOTING-BASED SYBIL DETECTION SYSTEM

In this section, we detail the implementation of VoteTrust. We first describe the basic idea and framework of our model, then propose two key techniques, trust-based votes assignment and global vote aggregating, which leverage the friend invitation graph to detect social network Sybils.

A. Overview

In our prior work [14], we have shown normal users are not likely to accept Sybil’s friend requests, leading to low acceptance percentages (i.e., the fraction of accepted friend requests) for Sybils. Although detectors could use this feature, an attacker could inflate the acceptance percentage of their Sybils by sending friend requests to other Sybils, who are guaranteed to accept (i.e., collusive voting). Therefore, we propose VoteTrust to effectively identify collusive voting and resist collusion attack. VoteTrust system employs the following two key techniques to prevent collusive rating:

Trust-based vote assignment: We assume that Sybils receive a limited number of friend requests from normal users due to the difficulty of enticing users to invite them. VoteTrust adopts a PageRank-like mechanism to propagate the voting capacity to each user. This process limits the number of votes each Sybil could cast (i.e., vote capacity). For example, in Fig.1, the vote capacity of Sybil community is limited by a few incoming links from normal region.

Global vote aggregating: Next, the system collects the votes casted to each node and aggregates these votes to estimate its global rating (i.e., the probability that the node is a normal user). The estimation is based on the amount of votes and the rating of corresponding voters. The idea behind is that low-rating nodes are more likely to be Sybils, and thus their votes should have lower weight. In order to prevent Sybils creating more vouching votes through enlarging the their community, the system ignores the votes collected from nodes with very low capacity.

B. Graph Models

To facilitate the design of VoteTrust, we separate the invitation graph (illustrated in Fig.1) into two graphs:

Link initiation graph: We model the link initiation interactions as a directed graph $G_I = (V_I, E_I)$, where nodes represent network users and directed links $(u, v)$ represent node $u$ send a friend request to node $v$.

Link acceptance graph: We model the link acceptance interactions as a weighted-directed graph $G_E = (V_E, E_E, W_E)$ where directed links $(u, v)$ represents $u$ receives a request from $v$ (inverse direction of link initiation edge). The weight value $x_{uv} \in W_E$, represents whether $u$ accepts ($x_{uv} = 1$) or rejects ($x_{uv} = 0$) the request.

Notice in this paper, we use the term In-link to represent the friend request going into Sybil community from normal users, and term Out-link to represent the invitation going out of Sybil community to normal users, also called as attack link. Clearly, Sybils (e.g., Spammers) would like to create out-links as many as possible to gather a large number of victims (e.g., spam audience) and to propagate spams via viral marketing. Hence, the goal of VoteTrust is being able to limit the number of out-links each Sybil could create, or in other words, to find out Sybils even they only create few number of out-links.

C. Trust-based Votes Assignment

We first address the problem of how to distribute the voting capacity (i.e., the number of votes a node can cast) across users? Clearly, we should assign most votes to normal users, even though an attacker can create many Sybil accounts. To do so, we first select some high trust users as seeds, then propagate to others through link initiation graph. As Sybil region has a fixed number of in-links (irrespective of its size), we could limit the voting capacity of Sybils.

1) Selecting Trust Seeds: Ranking system, like PageRank, is effective in identifying popular user in a network. We combine the PageRank score and other features (e.g., clustering coefficient and acceptance percentage of friend requests given in [14]) to find trust users. Finally through human scrutinizing, we can easily obtain certain number of trust seeds.

Suppose the system has $M$ vote capacity and $N$ nodes in total. Given a set of trust seeds (denoted as $S$), we first equally assign the $M$ vote capacity over $S$. For simplicity, we let $M = N$ by assuming that each node has exact one vote capacity on
average. Thus the initial vote capacity for a user \( u \) is,
\[
I(u) = \begin{cases} 
\frac{N}{|S|}, & \text{if } u \in S; \\
0, & \text{otherwise}
\end{cases}
\]

2) Votes Propagation: Vote propagation is based on link initiation graph \( G_I \). Notice the vote capacity of a user \( u \) is the sum of the vote capacity received from all the users who invite him. Suppose node \( v \) sends a friend request to \( u \), and it has \( \theta(v) \) votes and a out-degree of \( \omega(v) \). Thus node \( u \) will receive \( \frac{\theta(u)}{\omega(u)} \) votes from this neighbor, and its vote capacity can be computed as,
\[
\theta(u) = d \cdot \sum_{v:(v,u) \in E_I} \frac{\theta(v)}{\omega(v)} + (1 - d) \cdot I(u)
\]
where \( d \) is a decay factor. For formalized expression, we define the transition matrix \( T \) of invitation graph \( G_I \), where,
\[
T(u,v) = \begin{cases} 
0, & \text{if } (v,u) \notin E_I; \\
\frac{1}{\omega(v)}, & \text{if } (v,u) \in E_I.
\end{cases}
\]
Thus, we can rewrite the equation (1) as,
\[
\theta = d \cdot T \cdot \theta + (1 - d) \cdot I
\]

As other similar algorithm, the vote capacity can be iteratively computed. In practice, the computation will converge in only a fixed number of iterations.

D. Global Vote Aggregating

We next address the problem of how to aggregate the votes to compute the node rating? We define the trust rating \( p(u) (0 \leq p(u) \leq 1) \) for node \( u \) as the probability that \( u \) is a legitimate user. Such rating could help the social network detect Sybils. For example, we can consider \( u \) as Sybil if its trusting rating falls below a threshold.

For a node \( u \), the nodes whom \( u \) sends friend requests to compose his voter set. VoteTrust computes the rating of \( u \) by combining all the vote values from his voters. As voter of high rating is more likely to be normal user, we should assign higher value to his votes. Considering the example in Fig. 2, user A with rating 0.2 casts a negative vote to C, and user B with rating 0.9 casts a positive vote to C. If we treat all the votes equally, user C’s trust score will be \( \frac{1}{1+1} = 0.5 \). However, this computation ignores the fact that the vote of user B is more trust. Hence, we set the value of one’s vote as the voter’s trust rating, so the rating of user C becomes as \( \frac{1 \times 0.9}{1 \times 0.9 + 1 \times 0.2} = 0.82 \).

\[
\text{vote} = 1, \text{rating} = 0.2 \quad \text{vote} = 1, \text{rating} = 0.9
\]

We now formalize the vote aggregating mechanism based on link acceptance graph \( G_E \). Let \( \theta(v) \) and \( p(v) \) denote the vote capacity and current rating of node \( v \), respectively. Then, user \( u \)'s rating \( p(u) \) can be computed as,
\[
\hat{p}(u) = \frac{\sum_{v:(v,u) \in E_E} \theta(v) \cdot p(v) \cdot x_{v,u}}{\sum_{v:(v,u) \in E_E} \theta(v) \cdot p(v)}
\]

Initially, we assign the \( p(u) = 0.5 \) for each user. Similar with the computation of vote capacity, the node rating can be computed iteratively and stabilize quickly.

Notice that rating computation can be regarded as an binomial proportion estimation, the result may be unreliable if there are only a small number of votes casted for a node (i.e., small-sample problem). Hence, we combine Wilson score to increase the confidence. Suppose the number of trials (samples) is \( N(u) \), which is the number of total votes cast to \( u \). The Wilson score can be shown to be weighted average of \( \hat{p} \) and 0.5, with \( \hat{p} \) receiving greater weight as the sample size (votes number) increases. It can be derived as,
\[
p(u) = \frac{\hat{p}(u) + 1 + z_{1-\alpha/2}^2}{2 + \frac{z_{1-\alpha/2}^2}{N(u) - 1}}
\]

where the \( z_{1-\alpha/2} \) is the \( 1-\alpha/2 \) percentile of a standard normal distribution. For 95% confidence level, \( z_{1-\alpha/2} = 1.96 \).

\textbf{Limiting the vouching votes:} An important problem we should address is how to prevent Sybils enlarging the number of total voting votes by increasing the community size. Considering the case shown in Fig. 3, Sybil community initially has 3 nodes sharing a total of 1 vote capacity (each with 1/3). If they vouch each other by accepting the friend requests within the community, each Sybil can collect 1/3 vouching votes from his voucher. When the Sybil community size increases to 5, the vote capacity of individuals drops to 1/5. However, each of them can send 2 requests to other Sybils and thus collect 2/5 vouching votes (> 1/3) from their vouchers. This means Sybils can maintain vouching votes for each individual as enlarging their community. In fact, a complete-connected community with \( N \) nodes and \( c \) capacity could create \( \frac{(N-1)}{2} \) vouching votes, and thus increasing the total number of friend requests they could send to normals.

![Fig. 3. Example of vote collection](image)

However, notice that individual vote capacity will decrease as Sybil community grows due to sharing a fixed number of in-links, VoteTrust limits the size of Sybil community by
ignoring the votes from nodes of very low capacity, i.e., below the threshold $\delta$. For example, in Fig. 3, the individual capacity reduces to $1/5$ from $1/3$ as the community adding two nodes, so all the vouching votes would be ignored in vote aggregation if we set $\delta = 1/3$. Choosing the threshold $\delta$ should make a balance between ignoring vouch votes within Sybil community and losing some normal votes. We shall show how to make such tradeoff in Section IV.

**Summary:** we sketch the algorithm in Fig. 4. Trust-based vote propagating guarantees Sybil communities get few vote capacity irrespective of their size. Global vote aggregating limits the number of friend requests Sybils could send to normals.

**Proof:** Suppose that the community forms a complete-connected graph. Each Sybil within it sends and accepts friend requests to each other. According to the work [15], the total votes flow into Sybil community, say $E_S$, is given by,

$$E_S = E_S^{in} - E_S^{out}$$  \hspace{1cm} (6)

This equation implies that the total vote capacity of Sybil community depends on the difference between incoming capacity $E_S^{in}$ and outgoing vote capacity $E_S^{out}$. And $E_S^{in}$ and $E_S^{out}$ can be calculated using the following expressions,

$$E_S^{in} = \frac{d}{1-d} \sum_{i \in V_{in}} f_i x_i$$  \hspace{1cm} (7)

$$E_S^{out} = \frac{d}{1-d} \sum_{i \in V_{out}} (1-f_i) x_i$$  \hspace{1cm} (8)

where $V_{in}$ represents the set of normal users that link to Sybils and $V_{out}$ is the set of Sybils that link to normals. Define $f_i$ as node $i$'s fraction of edges that link to Sybils. $d$ is the decay factor in Eq.(1), and we use $\alpha$ to denote $d/(1-d)$.

Let $\bar{x}_l$ and $\bar{x}_s$ represent the average vote capacity for each node in normal and Sybil community, respectively. To be fully connected, each Sybil sends $N_s/2$ invitations to others within the community. Here, $N_s$ is the size of Sybil community. Thus, for $i \in V_{out}$, $1-f_i = \frac{N_{out} + N_s/2}{N_{out} + N_in/2}$. We denote $f_{in}$ as the average of $f_i$ for users in $V_{in}$. Substitute to Eq.(6), we get,

$$E_S = \alpha \left( N_{in} f_{in} \bar{x}_l - E_S \frac{N_{out} + N_s/2}{N_{out}} \right)$$  \hspace{1cm} (9)

Solving the above equation, the total vote capacity of Sybil community is,

$$E_S = \frac{\alpha N_{in} f_{in} \bar{x}_l}{\alpha N_{out} + N_s/2 + 1}.$$  \hspace{1cm} (10)

Due to that Sybils have similar structure, we assume they have equal trust rating $\hat{p}_s$. According to the rating model in Eq.(3), the average rating of Sybil can be calculated as,

$$\hat{p}_s = \frac{N_s \bar{x}_s \hat{p}_s + \gamma N_{out} \bar{x}_l \hat{p}_l}{N_s \bar{x}_s \hat{p}_s + N_{out} \bar{x}_l \hat{p}_l}$$  \hspace{1cm} (11)

where $\gamma$ is the acceptance percentage of Sybil out-links. Based on Eq.(11), we can get the Sybil's total out-links,

$$N_{out} = \frac{N_s \bar{x}_s (\hat{p}_s - \gamma)}{\bar{x}_l \hat{p}_l (\hat{p}_s - \gamma)}$$  \hspace{1cm} (12)

---

**III. SECURITY PROPERTIES**

This section provides a formal analysis of the security properties of VoteTrust. Our analysis provides bounds on the number of friend requests that individual Sybils could send to normal users, and the size of Sybil community.

A typical attack mode is illustrated in Fig. 5. Attacker would connect Sybils into a complete-connected graph, so that each Sybil could receive the maximum vouching votes from other Sybils. We call normal users (either intentionally or accidentally) sending invitations to Sybils as promoters.

**Theorem I:** If a Sybil wants to evade detection, the average number of his out-links $N_{out}$ (i.e., the number of friend requests sent to normal users) needs to satisfy the following upper bound,

$$N_{out} \leq \rho N_{in} \cdot \frac{\delta_f - \delta_f^2}{\delta_f - \gamma}$$  \hspace{1cm} (5)

where $N_{in}$ is number of In-links from normal to Sybil community, and $\rho$ is a constant factor. $\delta_f$ is the detection threshold and $\gamma$ is acceptance percentage of Sybil out-links.
Notice that \( \bar{p}_s \in [0, 1] \), we get \( \frac{\partial N_{out}}{\partial \bar{p}_s} < 0 \). This indicates that \( N_{out} \) decrease as \( \bar{p}_s \) grows.

To evade detection, Sybils should maintain their rating \( \bar{p}_s \) above the detection threshold \( \delta_f \). Therefore, \( N_{out} \) has the least upper bound when \( \bar{p}_s = \delta_f \). Note that \( E_S = N_s \bar{x}_s \), substituting Eq.(10) into Eq.(12), and letting \( \bar{p}_s = \delta_f \) yields the inequality,

\[
(4\alpha + 4)\alpha N_s^2 + (2\alpha N_s - 2bc)N_{out} - bcN_s \leq 0 \tag{13}
\]

where \( a = \delta_f - \gamma \), \( b = \delta_f - \delta_f^2 \) and \( c = \alpha N_{in}/\rho \).

According to the property of quadratic function, we know the condition that the above inequality holds is \( N_{out} \geq 0 \) and,

\[
N_{out} \leq \frac{2bc - 2\alpha N_s + \sqrt{(2\alpha N_s - 2bc)^2 + (16\alpha + 16)abcN_s}}{8\alpha + 8a} = \frac{2bc - 2\alpha N_s + 2\sqrt{(2\alpha N_s - (2a + 1)bc)^2 - (4\alpha^2 + 4a)b^2c^2}}{8\alpha\alpha + 8a} \leq \frac{2bc - 2\alpha N_s + 2\alpha N_s + (4a + 2)bc}{8\alpha\alpha + 8a} = \frac{bc}{2a}
\]

Substitute the \( a, b \) and \( c \) into the above inequality, and let \( \rho = \frac{\alpha_{fin}}{2p_l} \), we get the upper bound (5). Theorem 1 is proved.

From theorem 1, we learn that the number of out-links \( N_{out} \) for individual Sybils is mainly constrained by the number of in-links \( N_{in} \), exhibiting a linear relationship. In theorem 1, the constant factor \( \rho = \alpha_{fin}/2p_l \), where \( \alpha = d/(1 - d) \), \( d \) is the decay factor in Eq.(1). \( f_{in} \) represents the ratio of the number of in-links of Sybils to total out-degree of promotors. \( p_l \) indicate average rating in \( G_N \).

Theorem 1 shows the bound of individual Sybils, now we show VoteTrust could also impose a bound on the Sybil community size.

\textbf{Theorem 2:} The vouching votes within Sybil community can be collected by system if the community size \( N_s \) satisfies the upper bound,

\[
N_s \leq \sigma \frac{N_{in}}{\delta_v} \tag{14}
\]

where \( N_{in} \) is number of In-links from normal to Sybil community, and \( \sigma \) is a constant factor. \( \delta_v \) is the threshold for vote collection.

\textbf{Proof:} In a complete-connected Sybil community with size \( N_s \), each Sybil averagely gets \( \bar{x}_l \) votes. Thus the total votes in this community is \( E_S = N_s \bar{x}_s \). Substituting it into Eq.(10) we get,

\[
N_s \bar{x}_s = \frac{\alpha N_{in} f_{in} \bar{x}_l}{\alpha N_{out} N_{out} + N_s/2} + 1 \tag{15}
\]

Note that the votes will be ignored by VoteTrust when \( \bar{x}_s < \delta_v \), so \( N_s \) has the lowest upper bound when \( \bar{x}_s = \delta_v \). Substituting this equation into Eq.(15), we get

\[
\frac{\delta_v}{2} N_s^2 + \left( \frac{a \delta_v - b N_{in}}{2} \right) N_s - b N_{in} N_{out} \leq 0 \tag{16}
\]

where \( a = (\alpha + 1)N_{out} \) and \( b = \alpha f_{in} \bar{x}_l \) both defined in Theorem 1. Solving the inequality like that of inequality (13) yields the upper bound of community size in Eq.(14). Theorem 2 is proved.

From theorem 2, we learn that the community size is constrained by both in-links number \( N_{in} \) and threshold \( \delta_v \). The constant factor \( \sigma = \alpha f_{in} \bar{x}_l \), and the definition of these constants is the same as those in Theorem 1.

We use parameters obtained from Renren dataset to provide a concrete examples of the two upper bounds. Commonly, the decay factor \( d = 0.85 \), thus \( \alpha = 5.67 \). Through analyzing Renren social network, we get the \( f_{in} \approx 0.024 \) and \( p_l \approx 0.63 \), thus \( \rho = 0.107 \). The threshold \( \delta_f \) and \( \delta_v \) are set as 0.4 and 0.12. The acceptance rate \( \gamma \) is 0.2. Suppose Sybil community has a total of \( N_{in} = 20 \) in-links, each Sybil can only send 2.6 friend requests to normal users in average, and the Sybil community size should not exceed 60.5.

\textbf{IV. Evaluation}

In this section, we demonstrate the performance and security properties of VoteTrust through conducting simulations and detection over real social network.

\textbf{A. Data Set and Methodology}

Our experiment data sets are collected from Renren, which is the largest online social network in China, with more than 160 million registered users [10]. Renren provides function-ality and features similar to Facebook. In our experiment, we use the PKU network containing more than 200K users. We use this regional network because: First, we have complete friend request interaction records (a total of 5.01 million) for anonymous users in PKU network. Second, PKU network is the most well-known and popular regional network in Renren, which makes it an enticing target for attackers (e.g., Sybils are disguised as PKU users to increase their popularity). As a result, PKU network suffers more Sybils than other networks.

We evaluate the performance of VoteTrust through adding artificial Sybil accounts over PKU network and directly detecting the real Sybils in the network. The two sets of simulations serve to validate different aspects of the advantages of VoteTrust. The simulations based on artificially added Sybils validate whether our theoretical bounds hold in different attacking cases (e.g., different number of in-links or Sybil community size). The detection performed on real network further demonstrates to what extent VoteTrust could outperform other traditional algorithms (including Trustrank and Badrank) in finding out real social Sybils.

To measure the detection performance, we use the metrics \textit{Precision} and \textit{Recall}. In particular, suppose that \textit{oracle function} \( O \) represents the ground truth of users,

\[
O(u) = \begin{cases} 
0, & \text{if } u \text{ is Sybil;} \\
1, & \text{if } u \text{ is legitimate user.}
\end{cases}
\]
Given detection threshold $\delta$ and rating $p$ of users, precision and recall are defined as,
\[
\text{Precision}(\delta) = \frac{|\{u \in V | p(u) < \delta \text{ and } O(u) = 0\}|}{|\{v \in V | p(v) < \delta\}|} \tag{17}
\]
\[
\text{Recall}(\delta) = \frac{|\{u \in V | p(u) < \delta \text{ and } O(u) = 0\}|}{|\{v \in V | O(u) = 0\}|} \tag{18}
\]

B. Limiting the Ability of Sybil Community

We evaluate the resilience of VoteTrust through artificially injecting Sybil communities into PKU network. We assume that Sybils form a tight complete-graph to maximize the vouching votes within the community (the worst case).

Limiting attack number: To verify the bound on the number of out-links individual Sybils can create, we inject $N_s$ artificial Sybils to PKU network. To vouch each other, Sybils form a tight complete-graph by sending $N_s/2$ friend requests to each other. Then, Sybils accept the friend requests within their community. We set the probability that a normal user accept Sybil friend request as 0.2 according to the measurement of [14].

We create a complete-connected Sybil community with $N_s = 100$ Sybil identities, and create 10 in-links by randomly selecting source and target from normal and Sybil nodes, respectively. We then allow each Sybil to create out-links by send friend requests to nodes randomly selected from PKU network. Fig. 6 shows the recall of Sybils as the number of out-links varies from 1 to 20, under detection thresholds of 0.4 and 0.5, respectively. From the figure we see most Sybils could be detected when Sybils only create 4 out-links, e.g., 88.7% and 96.6% when using thresholds of 0.4 and 0.5, respectively. On average, each Sybil could only create 1.63 out-links before they are detected using $\delta_f = 0.4$, which is much smaller than the number of in-links.

To validate the bound given by inequality (5), we vary in-link number from 10 to 50, and compute the average number of out-links that Sybil can create before being detected. Fig. 7 plot the theoretical upper bound and experiment value, which are very close. The result verifies the security property of VoteTrust, demonstrating its ability in limiting Sybil attack capacity in sending unsolicited friend requests to normal users.

Limiting Sybil community size: Given a fixed number of in-links, the vote capacity of individual Sybils would decreases as the community size grows. To limit Sybil community, VoteTrust ignores the votes from users whose capacity is below the threshold $\delta$. From inequality (16) we know that increasing the threshold could gain the benefit of reducing the community size $N_s$ (benefit) since $N_s$ is proportional to $1/\delta$, but at a cost of losing votes for some normal users. Fig. 8 shows the trade-off curve when selecting different thresholds, where $Y$-axis is the community size factor $1/\delta$ and X-axis is the fraction of total lost votes (computed based on user vote capacity distribution). To make a balance between collecting most votes and limit Sybil community size, we select the turning point of $(0.01, 8)$, and get the corresponding threshold $\delta = 0.12$.

To examine how normal users and Sybils are affected under this threshold, we measure the the ratio of ignored votes to their total received votes for normal user, and ratio of ignored vouching votes to the total received vouching votes for Sybil.
community of different sizes ($N_s = 100$ and 500). Experiment assumes that Sybil community has 50 in-links and each Sybil sends 10 out-links.

Fig. 9 plots the above rates for normal users and Sybils in community respectively. First, the normal users lose only 0.45% votes casted for them, and 86.4% of normal users are unchanged. However, Sybils lose majority of their vouching votes, e.g., losing 57.1% when community size is 100 and 82.8% when community size is 500. This indicates that VoteTrust effectively eliminates vouching votes from Sybil community when the scale grows, meanwhile leading to little affects on normal users.

C. Detecting Real Sybils

In this section, we detect real Sybils existing in PKU network, and compare VoteTrust against other two alternatives, i.e., TrustRank and BadRank. We have two ground-truth datasets: The first one contains 500 randomly selected PKU users. An expert team carefully scrutinized all accounts to classify them as normal users or Sybils by looking over detailed profile data, including uploaded photos, messages sent and received, email addresses, and shared content (blogs and web links). This manual checking finds 73 Sybil accounts in the dataset. We use this dataset to evaluate the overall performance of VoteTrust in detecting Sybils over the PKU network. The other contains 2502 PKU Sybil accounts that already detected and banned by Renren technical team using prior techniques. Since it is a more large ground truth data about Sybils, we use this dataset to confirm the advantage of VoteTrust in detecting Sybils.

TrustRank versus VoteTrust. TrustRank only leverages the heuristic that Sybils have few in-links, and propagates the trust score from trust user to others. However, Trustrank may mix Sybils with many non-popular users that also have few in-links. We first evaluate the performance of TrustRank and VoteTrust over the first ground-truth dataset checked by human effort.

Fig. 10(a) and Figure 10(b) provide the precision and recall of TrustRank and VoteTrust varying with the detection threshold. For TrustRank, we use user rank as the detection threshold, instead of the uneven user score, to make the figure clear. From Fig. 10(a), we see that Trustrank cannot achieve a very high precision, even for nodes with very low rank. For example, the average precision for 100 user of lowest rank is only 67.4%. This demonstrates that Trustrank mixes Sybils with many non-popular users. In contrast, the precision of VoteTrust could achieve high precision compared with Trustrank, because it further exploits implicit user feedbacks reflected in the friend request acceptance.

![Fig. 10. Recall and precision of (a) TrustRank and (b) VoteTrust](image)

To demonstrate the above advantage, Fig. 11(a) plots the precision-recall curve (PRC) of both TrustRank and VoteTrust. The corresponding precision is eleven-point interpolated average precision. We see that given the same recall, VoteTrust outperforms TrustRank in detection precision. For example, when recall is 70%, VoteTrust can get the precision of 71.5%, however, TrustRank only get the precision of 51.0%. In average, VoteTrust can improve the precision of Sybil detection of 32.9% than TrustRank in manually checked dataset.

Fig. 11(b) plots the RPC of TrustRank and VoteTrust for 2502 banned accounts provided by Renren. Here, we assume that only these banned accounts are Sybils, and other nodes in PKU network are normal users. Averagely, VoteTrust improves the precision of 75.6% than TrustRank for these banned accounts, and the improvement is 50.1% when limiting the Recall $\geq 80\%$. The horizontal line in Figure 11(b) is because 78.7% of the banned accounts have the same lowest TrustRank score. Notice here both systems have low precision because the assumption of taking Sybils out of our dataset as normal users, but this does not affect the gain of VoteTrust over Trustrank.

BadRank versus VoteTrust. BadRank attempts to further reduce the rank of Sybils by punishing their vouchers. It propagates the bad score from Sybil seeds to users who link to Sybils. However, the performance of BadRank is significantly depend on Sybil seeds and may punish innocent users that are enticed to send requests to Sybils. We compare the performance of BadRank and VoteTrust on both human checked sample and real banned accounts. We select 100 highest in-degree nodes from the 2502 banned accounts as the bad seeds of BadRank.

Fig. 12(a) plots the eleven-point interpolated average precision of BadRank and VoteTrust for our human checked ground-truth dataset. We find that BadRank cannot efficiently detect Sybils in our dataset. This is because the Sybils cannot be detected if they are out of the Sybil seed’s community.
We next focus on the top 1000 users with highest badrank score to examine whether the VoteTrust still have better performance. We randomly pick 200 nodes from 1000 top badrank users, and classify them as normal users or Sybils through manual checking. Fig. 12(b) shows the RPC curves BadRank and VoteTrust for this set of users. We find that there only have 48.2% Sybil accounts in the top BadRank users, demonstrating the low accuracy of BadRank in Sybil detection. In comparison, VoteTrust improves the precision of 44.5% at same recall level on average.

Fig. 12(c) plots the the RPC of BadRank and VoteTrust for 2502 banned accounts. Even seeds are selected from these accounts, we see that VoteTrust still outperforms BadRank. On average, VoteTrust improves the precision by 41.6%, as compared with BadRank. Further, VoteTrust could recall more Sybils. For instance, given the same precision level of 0.035, VoteTrust can recall 60% banned Sybil accounts, however BadRank only recalls 5%. All the result show that VoteTrust is more effective in OSNs.

D. Separating Normal User from Sybils

A key reason that the VoteTrust achieves high precision is that it could distinguish Sybils and non-popular users based on the acceptance of their friend requests. However, traditional ranking systems just mix them together. In order to better understand the underlying reason, we focus on non-popular users in the low-rank range.

Fig. 13 depicts the distribution of friend requests of each PKU user sends and receives, where we sort users in descending order according to their number of received requests. We see that the distribution of received requests is highly skewed, with top 20% popular users receive about 80% friend requests. Hence, when using traditional ranking systems, most users would have low rank and are mixed with Sybils. In contrast, the requests user send distribute more evenly, e.g., 80% requests distribute across nearly 80% users. Hence, even majority nodes receive few friend requests, we could still use the feedbacks on the friend requests they send.

Focus on the non-popular user and Sybils, we compare the efficiency of TrustRank and our VoteTrust. Fig. 14(a) plots the cumulative distribution of TrustRank Score for all users. We see that there are 31.1% users got the same lowest TrustRank score and 80% users’ score is less than $10^{-6}$. Here, we define the these 80% users as non-popular user due to their low ranks and few incoming requests. However, VoteTrust could separate the Sybils from non-popular users. Fig. 14(b) depicts the distribution of their VoteTrust ratings, which are more evenly distributed. About 20% percent non-popular users have the rating of 0.5 due to sending few friend requests, however, we would be able to distinguish them if they continue to send request to befriend with others.

Finally we validate the accuracy of VoteTrust to classify Sybils and non-popular users based on our manually checked dataset (about 400 non-popular users). The final aggregated result is plotted in Fig. 15. When the classify threshold $\delta$ ranges from 0.2 to 0.6, VoteTrust can get the accuracy of 84.7% on average, with the maximum accuracy (85.7%) achieved at $\delta = 0.4$. Therefore, VoteTrust is a highly accurate classifier.
V. RELATED WORK

Recently, there has been a great effort in defending against Sybils (e.g., spammers) in online social networks. This section summarizes these studies on Sybil detection in OSNs.

Feature-based approaches: Fake accounts are created for profitable malicious activities, such as spamming, click-fraud, malware distribution, and identity fraud. Some fakes are created to increase the visibility of spam content, forum posts, and fan pages by manipulating votes or view counts. Hence, many works analyze aberrant behavior or spam content to detect Sybil accounts. By this way, Researchers have found a number of Sybils accounts in OSNs, such as Facebook [5], Renren [14] and Twitter [16]. On the other hand, there are Sybil detection systems based on the Bayesian filters and SVMs in Twitter [17] and Facebook [18]. However, automated feature-based Sybil detection suffers from high false negative and positive rates due to the large variety and unpredictability of legitimate and malicious OSN users’ behaviors.

Social network-based approaches: Some decentralized protocols OSNs [9], [11], [13] leverage the graph structure to detect Sybil community. These techniques rely on the assumption that Sybils form tight-knit communities. However,Yang et al. [14] recently analyzed a sample of 660K Sybils in the RenRen OSN, and found that they do not form tight-knit communities. Hence, existing network-based Sybil defenses are unlikely to succeed in today’s OSNs.

Link-based Ranking System: Recently, lots of research communities adopt the traditional ranking system to rank users according to their perceived likelihood of being Sybils. Trustrank [19] and Badrank [20] leverage the idea that prorogate good score or bad score from selected seeds according the graph links. Saptarshi et al [21] and Chao et al [22] employ Badrank-like algorithm to combat link farm in Twitter. However, all these link-based algorithms cannot be used directly in Sybil detection due to their low accuracy.

VI. CONCLUSION

This paper presented VoteTrust, a rating system that leverages user interactions of initiating and accepting links to defend against Sybil attacks. By using the technique of trust-based vote assignment and global vote flow aggregation, VoteTrust estimates the likelihood that the user is a Sybil with high accuracy. We demonstrate the benefits of VoteTrust in restricting the attack power of adversaries: the number of friend requests Sybils could send to normal users is limited by the number of friend requests they receive from normal users. We demonstrate the VoteTrust can significantly outperform traditional ranking systems by evaluating it on Renren network.

Acknowledgments

We thank the anonymous reviewers for their comments. This work was supported by the National Basic Research Program of China (Grant No. 2011CB302305), the National Science Foundation for Young Scholars of China (Grant No.61202423) and the State Key Program of National Natural Science of China (Grant No. 61232004).

REFERENCES