Sybil Defenses in Mobile Social Networks

Wei Chang†, Jie Wu†, Chiu C. Tan†, and Feng Li‡
†Temple University, ‡Indiana University-Purdue University Indianapolis
Email: †{wei.chang, jiewu, cctan}@temple.edu, ‡fengli@iupui.edu

***Abstract*—Mobile social networks are vulnerable to Sybil attacks. By creating a large number of fake identities, malicious users can gain a disproportionately high benefit through a Byzantine fashion. Most social network-based Sybil defenses adopt the assumptions that the honest region is a fast-mixing network. However, more and more evidence shows that some real social networks are not fast-mixing, especially when only strong-trust relations are considered. Moreover, the accuracy of all existing solutions is related to the number of attack edges that the adversary can build. In this paper, for addressing these problems, we propose a local ranking system for estimating trust-level between users. Our scheme has three unique features. First, our system is based on both trust and distrust relations. Second, instead of storing the entire social graph, users carry limited information related to themselves. Last but not least, our system weakens the impacts of attack edges by removing several suspicious edges with high centrality. We validate the effectiveness of our solutions through comprehensive experiments.**

***Index Terms*—Community, distrust, local gateway, mobile social networks, sybil attacks, signed social networks.**

I. Introduction

In recent years, researchers have designed different frameworks for cooperative services of smartphones [1]. However, the open environment makes these systems vulnerable to *Sybil attacks*. In a Sybil attack, an adversary creates a large number of fake identities (Sybils), and since all Sybils are controlled by the adversary, she can subvert the system by making actions that benefit herself. Here is an example from a data sharing service. In order to reduce a cellphone's usage of cellular networks, when several nearby users are watching the same online content, they locally share partial data (which has been downloaded to each other's phone) via other networks; the amount of shared data a user obtained depends on the quantity he provided. The attacker can benefit from the service by only downloading one portion of data and obtaining other parts via different Sybil identities.

Traditional social network-based Sybil defenses are based on two common assumptions: (1) sybil-free social networks are fast-mixing; (2) Sybils can only fool a limited number of honest users. Intuitively, a fast-mixing network consists of a single well-connected core, even after the decomposition of the graph into its k cores [2]. However, these assumptions lead to two problems. First, the accuracy of a sybil defense algorithm is highly related with the number of attack edges. Second, it is unclear whether the assumption about the fast-mixing feature will hold in all social networks. Research by [2], [3] suggested that in some social networks, there are multiple cores with considerable sizes. Since we cannot be sure that fast-mixing will always be present, it is reasonable to make the following assumption: *honest users may cluster into one community, or several communities with similar sizes.* But, under the new assumption, using most existing sybil defenses will result in high false positive rates (many honest users from other honest communities will be labeled as sybils).

In order to solve the problems, we design a Sybil defense system in mobile social networks. Considering that trust is pairwise instead of global, our system explores both trust and distrust relations among users. For addressing the problem of multiple honest communities, our system uses a distributed sybil defense algorithm based on signed networks. For reducing the impacts of attack edges, we propose a gateway-breaking algorithm. Note that the connectivity between honest communities is different from that between communities of an honest and a Sybil. *If we cut off several high centrality edges from the social graph, the connectivity between honest nodes bears much less of an impact than that between sybil and honest nodes*. This algorithm *potentially* increases the accuracy of any social network-based sybil defense algorithm.

II. Related work

Neighborhood Monitoring-based Sybil Defenses: since attackers have a limited number of real devices, a group of Sybils are actually sharing one device. Based on this observation, Sybils can be detected by letting honest users monitor signals' features [4], [5] or moving patterns of other users [6]. Paper [7] proposed a strategy for detecting Sybils in mobile networks: each user locally stores a friendship graph and a foe graph. Whether a suspected node will be regarded as a Sybil is dependent on the similarity of the graphs between the involved users. But, their solution is not suitable for the late participants or frequent travelers. Moreover, none of the above schemes work well if the attackers only conduct malicious actions to a small set of honest users instead of all users.

Social Network-based Sybil Defenses: based on the small-cut structure of a social network where a Sybil node is more likely to be connected with Sybil nodes and a Sybil can only fool a limited number of honest nodes, graph theorems are usually adopted to detect Sybils [8]–[10]. Our proposed scheme is also based on the unique link structures of Sybil nodes, but unlike previous works, we use a more realistic model that the honest users present single or multiple communities. Under this model, most of the existing social network-based Sybil defenses suffer high false positive rates.

Signed social network [11] is a new type of social networks. Unlike traditional social networks, signed networks allow users to add both trust (positive) and distrust (negative) relations to

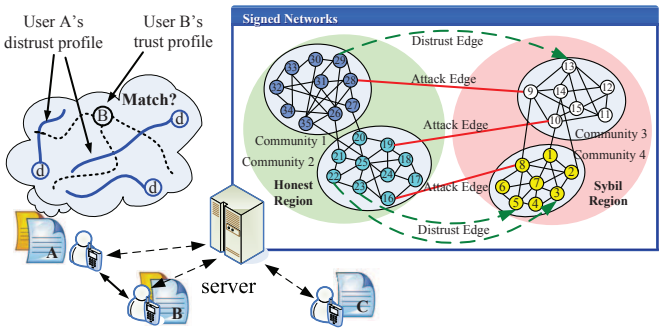


Fig. 1. System model.

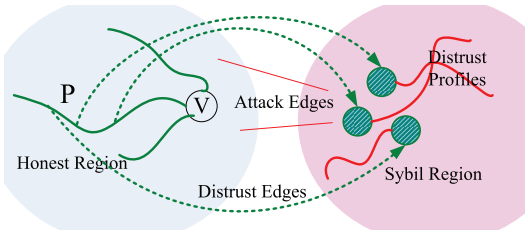


Fig. 2. The generation of distrust social profiles.

their social profiles [12]. In our system, we *create* a signed social network behind a given mobile social network to capture the mutual relations among users. The idea of our system is that, by using the created social graph, Sybil nodes can be detected through the propagation of both trust and distrust.

III. System Model

Our system consists of two parts: a remote server and users. The server is responsible for two jobs: (1) storing and periodically pruning the created signed network graph; (2) assigning *randomly sampled* social profiles to users for computing the trust-level between users.

We assume that each honest user has one mobile phone, which is associated with a single identity, while attackers may associate multiple fake identities with one device. For the remainder of the paper, we call the identities held by the attackers as *Sybil identities*, and we also refer to identities as nodes. Each identity is required to periodically send a special message to the server to keep valid, and the server will return updated social profiles. Unlike traditional models, we assume that the honest region of a social network may form several communities. Exactly how many honest communities may be formed is determined by the social networks being considered.

Our system works as follows. Each user locally stores two *randomly sampled* social profiles: a trust and a distrust profile. Whenever two strangers encounter and want to establish cooperative service, each user's phone will exchange the trust profile (together with signature and timestamp), and locally compute a trust and a distrust score to determine whether the other user is a Sybil. In order to increase the accuracy, a special pruning algorithm is running on the server.

In our system, to capture the trustworthiness between users, we *create* a signed network. If node u trusts (distrusts) node v, then $E^+_{uv} = 1$ ($E^-_{uv} = 1$). The trust relations come from the friendships of users, so E^+ are always bidirectional, while the distrust edges are unilateral. $N^+(v)$ indicates a list of directly trusted nodes of v, and $N^-(v)$ gives the directly distrusted list. For example, in Fig. 1, $N^+(v_6) = \{v_5, v_7, v_8\}$, and $N^-(v_{29}) = \{v_{13}\}$.

IV. Signed Network-based Sybil Defense

A. Generation of Signed Network

Our system detects Sybils by exploring both trust and distrust relations. But, how to find the distrust relations is an open question. Here, we propose several options. First, consider that multiple Sybils are sharing a single phone, and that each Sybil identity needs to periodically report some message in order to keep itself valid. For some honest users, they may catch the instant that an attacker switches her Sybil identities. If that is the case, then the honest users will report this misbehavior to our server, and a distrust edge will be added from the reporter to the accused. Second, when several honest users, who have been fooled by the same *attacker* via different identities, physically encounter the attacker at the same time, some of them may notice that the attacker is using a different identity; the honest one could report this event to our server. Besides these two options, any other neighbor monitoring techniques may also be adopted.

Our system does not require that each honest user must monitor other users. Instead, we just use the data provided by some cooperative users, who are willing to report the misbehavior events they witnessed. Since the attackers can use hit-and-run policy, sharing the knowledge about the abnormality increases the accuracy of our system. Traditional reputation system-based solutions only maintain a global trust value for each user while a signed social network preserves much more information, especially when adversaries have a target list while act normally to other nodes.

B. Scheme of Signed Network-based Sybil Defense

When user A needs to interact with a nearby stranger B, both of them will locally measure a trust-level between each other by using *Signed Network-based Sybil Defense algorithm* (SNSD), as shown by Algorithm 1. SNSD makes use of social profiles, which are stored in users' phones. SNSD first computes a trust score and a distrust score. Based on the scores, it further classifies nodes into: *trusted*, *neutral*, or *distrusted*. Because the Sybil-free social networks may contain several communities, we use the neutral tag to label the honest users from other honest communities. Similar to most social-based Sybil defense algorithms, SNSD also applies random walks to represent the propagation of trust and distrust. Since the random paths are generated by the server, the attackers cannot control how the random walks are conducted.

1) The Generation of Social Profiles: Each user locally records a trust and a distrust social profile. When a new user V joins our system and provides his trust-friend list, the server will generate the two profiles. The generating procedure for a trust-relation profile is as follows: the server first sends out K random walkers from V. Each walker will conduct an l-length

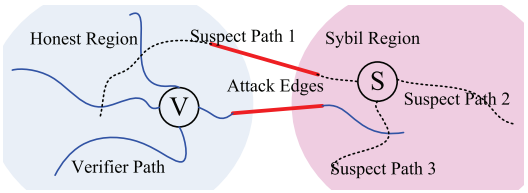


Fig. 3. The computation of trust score. The solid blue lines indicate V's trust social profile, and the dashed black lines represent the trust social profile of S. When V sets the verifier threshold $k_t = 2$, then only suspect path 1 will be fully verified. $Ver(V,S) = 1$, $|K| = 3$, and $Trust(V,S) = 1/3$.

Algorithm 1 SNSD Algorithm (Run on node V)

1: Exchange trust social profile with the other user S.
2: **Trusted degree calculation:**
3: Extract paths from the trust social profiles of V and S.
4: Set $Ver(V,S)$ to be 0.
5: **for** Each suspect path P_s **do**
6: **if** P_s intersects more than half of verifier paths **then**
7: $Ver(V,S) = Ver(V,S) + 1$.
8: Compute trusted degree by Equation 1.
9: **Distrusted degree calculation:**
10: Extract paths from the distrust social profiles of V.
11: **for** Each suspect path P_s **do**
12: **if** P_s comes across more than half of distrusted verifier paths **then**
13: $Dis(V,S) = Dis(V,S) + 1$.
14: Compute trusted degree by Equation 2.
15: According to Equation 3, label the suspect S.

random walk along trust edges. A generated path represents one possible way of trust propagation. These paths will be sent to V as a trust social profile. Obviously, the profile is a random sample of V's l-hops friendship.

In order to impersonate real users, attackers have to create a large friend set for each Sybil identity. Usually, such friendships are created by letting Sybil nodes friend each other. Therefore, friends of a distrusted node are likely to be distrustful. Our server creates V's distrust social profile by using the distrust relations of both V and his trusted friends. Take Fig. 2 as an example. First, a distrust seed set is generated: along trust edges, the server computes $K = 3$ short-length random paths from V (solid green lines). The ends (shadowed circles), which are distrusted by the nodes on these paths, form the seed set. From each seed, another l-length random walk will be conducted, and the paths (solid red lines) will be used as the distrust social profile of V.

2) Trust Level Estimation: The computation of the trust-level is based on the similarities between users' social profiles; SNSD gives a probability value to represent to what degree a node can be trusted. Whenever strangers V and S encounter, they will exchange their trust-social profiles. After obtaining the profiles, V will locally compute a trust score and a distrust score for S. For the ease of description, the paths in V's trust social profile are named as *verifier paths*, and the paths in the trust profile of S are called *suspect paths*, as shown by Fig. 3. If there is a common node on both a verifier path and a suspect path, the suspect path is verified once; when a suspect path has been verified more than k_t times, where k_t is a constant,

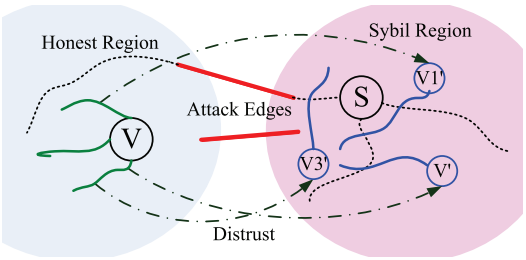


Fig. 4. The computation of distrust score. The distrust social profile of V contains 3 distrust paths (solid blue lines). If we define a verified distrust path as a suspect path that comes across at least half of the distrust verifier paths, then only suspect path 3 is verified. $Dis(V,S) = 1$, $K' = 3$ and $DisTru(V,S) = 1/3$.

we say that this suspect path is fully verified. Let $Ver(V,S)$ be the number of fully verified paths, and recall that there are totally K random paths in a trust social profile. In regard to V, the trust score of S is given by: $Trust(V,S) = Ver(V,S)/K$

For the computation of distrust score, SNSD considers V's distrust social profile and the trust social profile of S, as shown by Fig. 4. We name the paths from V's distrust social profile as *distrust verifier paths*, and use K' to represent the size of V's distrust social profile. When there are k_t distrust verifier paths having common nodes with a suspect path, this suspect path is a fully verified distrust path. Let $Dis(V,S)$ be the total number of fully verified distrust paths, and the distrust score of S in regard to V is given by: $DisTru(V,S) = Dis(V,S)/K'$

The final label of S, $L(V,S)$, is determined by the difference of the two scores: $z = Trust(V,S) - DisTru(V,S)$. Let α, β be two thresholds, $1 \geq \alpha > \beta \geq -1$. When $z \geq \alpha$, then $L(V,S) =$'Trusted'; if $\alpha \geq z \geq \beta$, $L(V,S) =$'Neutral'; otherwise, $L(V,S) =$'Distrusted'.

C. Security Analysis

Sybil attacks are hard to defend against, because Sybil nodes can impersonate honest nodes by supporting each other. However, SNSD forces the attacker into a dilemma. On the one hand, in order to boost the trust scores, it is better for Sybils to cluster into one community, such that the verifier paths are more likely to encounter a suspect path. As the attacker cannot predict how many and from which attack edges the trusted verifier paths may enter into the Sybil region, single community structure provides more chance for having contact with verifier paths. On the other hand, for reducing the distrust scores, the attacker should build Sybils into multiple communities for two reasons. (1) the probability of incurring the distrust verifier paths is proportional to the size of a Sybil community; having a single Sybil community may result in high distrust scores. (2) if a distrust verifier path enters Sybil region, having multiple communities can trap it in one of the communities instead of threatening all Sybils.

Although the attackers can adopt bad mouthing strategy, in our system, such strategy has very limited impacts on honest users. The main reason is that the distrust seeds come from an honest user's close friends. When the attackers create more distrust edges to honest users, the chance of having these fake distrust edges in other honest users' profiles is very small,

unless the attackers intensively add the fake distrust edges to a node. But if it is the case, our pruning algorithm will detect this high centrality structure. For attackers, the bad mouthing strategy brings many more problems than benefits.

V. Sybil gateway-breaking

In order to improve the accuracy, a special algorithm called Sybil gateway-breaking is proposed. Essentially, the algorithm prunes some suspicious edges of the signed social network.

A. Overview of our Sybil gateway-breaking algorithm

The accuracy of most existing social network-based Sybil defense algorithms is related to the number of attack edges. If a malicious user can establish more attack edges, the overall accuracy of a Sybil defense will decrease; if the attacker concentratedly establishes attack edges to a targeted group of users, there will be more victims. But, if we can remove some attack edges, the accuracy can definitely be increased.

Based on the above idea, we propose Sybil gateway-breaking algorithm (SGA), as shown by Algorithm 2. Consider that all of the paths connecting honest and Sybil nodes must go through the attack edges. The connectivity from a group of honest nodes (honest region) to a Sybil region is bounded by the quantity of attack edges. If each honest node is able to locally check whether it is fooled by others based on the connectivity, and deletes attack edges, the accuracy of *any* social network-based Sybil defense will be enhanced.

SGA consists of three parts: (1) Suspicious Edge Selection Algorithm; (2) Gateway Verification Algorithm (GVA); and (3) Attack Edge Detection Algorithm. GVA is the core part of SGA. It determines whether an edge is a gateway, which connects different communities. Since there may be multiple honest communities, only part of the gateways are attack edges while others are not. By exploring the distrust relations, the attack edge detection algorithm locates bad gateways. In order to reduce the computing time, a suspicious edges selection algorithm will be adopted at the beginning of SGA.

Algorithm 2 Sybil gateway-breaking algorithm (SGA)

```
Suspicious Edges Selection:
Select edges with high local betweenness as suspicious edges.
Use signed network-based Sybil defense algorithm (section V) to determine honest and Sybil.
Find shortest paths from the honest nodes to Sybil nodes.
Compute the visiting frequency of edges.
Take the edges with high visiting frequency as suspicious.
Gateway Verification:
Generate the initial neighbor set, {u}, {v}, and {w}.
Compute the number of unique paths from {v} to {u}, and from {v} to {w}
for Predefined times do
    Respectively add Δk disjoint neighbors into {u}, {v}, {w}.
    Compute the number of unique paths from {v} to {u}, and from {v} to {w}
    Compute the growing speeds of unique paths, S_vu and S_vw
    if |S_vu − S_vw| is greater than a threshold then
        E+_uw is a gateway;
E+_uw is not a gateway.
Attack Edge Detection:
Find attack edges from detected gateways by distrust relations; break them.
```

B. Gateway verification algorithm

Since GVA is the core of SGA, we will discuss it first. Whether two nodes located at the same community can be verified by their connectivity to other nodes. Intuitively, if one node's connectivity to the third node is much larger than that of the other node, it is very possible that the two nodes reside at different communities. Here, we use the number of unique paths to measure the connectivity feature.

***Definition** 1: Unique paths indicate a group of paths connecting two distinct nodes or regions without sharing a common edge.*

Because the amount of unique paths connecting two nodes is bounded by node degrees, we compute the unique paths from a region to another region. Let u and w be the ends of a given edge, v be the third node for checking the connectivity, and $UP(u, w)$ represent the number of unique paths from u to w. Based on the community structure of these three nodes, we may observe one of the three cases: (1) Either u or w shares the same community as v. (2) u, w and v locate in the same community. (3) None of them comes from v's community.

For the first case, assuming that nodes u and v reside in the same community, $UP(v, u)$ is much greater than $UP(v, w)$ since all the paths from v to w must go through the gateways between communities, which are limited. If we gradually increase the size of regions by Δk, the number of unique paths from w's region will stop growing much earlier than u's region. But, for the other two cases, we will not observe such differences. Based on this, we design a gateway verification algorithm (GVA) for checking whether a given edge is a gateway. The procedure is as follows. GVA gradually adds Δk disjoint neighbors into both regions, which respectively contain u and v (or w and v), and examines the amount of unique paths between the regions. Since we only care whether the edge is a gateway, GVA only checks the existence of the growing speeds' difference for a given node v.

The selection of the third node v is based on the fact that most random walks are trapped in their initial community. Before checking whether E^+_{uw} is a gateway, GVA first sends out several l-length random walks from node u or w, and lets the end nodes be the members of the verifier set. During the computation of the growing speed, GVA sequentially selects nodes from the set to be the third node v.

However, checking every edge of a social network by GVA is impractical. For fastening the process, we need to find out a set of suspicious edges first. Based on this idea, we propose a suspicious edge selection approach.

C. Suspicious edge selection

There are two options for finding the suspicious edges: one is based on node centrality and the other focuses on the edge centrality. The reason for caring about the node centrality is that under target attack, the target nodes will have

high centrality values. Hence, the server can first generate a local map (G) of each node, and computes each node's centrality on G. The GVA is only applied to the edges, which are connected with the high centrality nodes. The criteria of centrality we used is called the betweenness centrality [13], which is defined as the number of shortest paths passing a node out of the total number of shortest paths within a given network. $B(w) = \sum_{u,v \in G, u \neq v} g_{uv}(w)/g_{uv}$, where g_{uv} is the total number of shortest indirect paths linking nodes u and v, and $g_{uv}(w)$ is the number of those indirect paths that include node w.

Since all of the paths between a Sybil node and an honest node must traverse the same set of attack edges, the suspicious edges could be the edges passed by the majority of random paths, which connect the nodes with antagonistic relations (at least one directly distrust edge between the nodes). Therefore, the server randomly selects several pairs of antagonistic nodes, creates random paths between each pair of antagonistic nodes, and counts the visiting frequency of the transited edges. For the edges with high frequency, the gateway verification algorithm will be adopted. The above procedure substantially examines the centrality of edges based on partial nodes' relationships.

D. Attack edge detection

A gateway connects two communities together. Whether a gateway is an attack edge is determined by the distrust relationships between the communities. If either one of them, or both of them highly distrust the other, it is very likely that the gateway is an attack edge. Consider that the majority of random paths are trapped inside their own communities; instead of counting the number of distrust relations between two communities, we adopt random sampling to estimate it.

Attack edge detection algorithm works as follows: for each gateway, the server temporarily breaks it, and then, from its both ends, the server sends out k random walkers along the trust edges, respectively. The length of the random walks is a small fixed number, and all of the visited nodes form a sampling set. The server also creates another set, called a distrust sampling set, which consists of nodes distrusted by the sampling set's members. The intersection of these two sets indicates the intensity of distrust of the communities. The larger the intersection set is, the more likely the gateway is an attack edge. The server permanently remove the gateways with high intensity.

VI. Performance Analysis and Evaluation

Simulation setup: to generate a social network with different communities, we create 2 to 6 communities with 256 nodes in each; edges inside a community are randomly deployed according to the power law distribution. Links between different communities are also randomly generated, and 5% of the nodes from each community are connected to others. We compare our methods with the SybilGuard [9]. For the ease of description, we call our Signed Network-based Sybil Defense "SS", name our Sybil Gateway-breaking Algorithm "GB", and use "SG" to represent SybilGuard.

A. Simulation metric

AFNR and AFPR: the first metric we applied is called Average False Positive Rate (AFPR) and Average False Negative Rate (AFNR). If an honest node falsely regards another honest node as Sybil, we call the event a false positive (FP). A false negative (FN) means a Sybil being regarded as an honest node. Since each honest node locally determines whether to accept other nodes, we compute the False Positive Rate (FPR) and False Negative Rate (FNR) at each node, and then, compute the averages of them. $FPR = \text{FP}/(\text{FP+TN})$, $FNR = \text{FN}/(\text{TP+FN})$, where TP represents the total amount of true positives, and TN represents that of true negatives.

SCR: A distributed network system can tolerate some faults made by Sybils. We term such robustness Sybil Conquered Rate (SCR). We assume that if more than $1/3$ of the accepted nodes of an honest user are sybils, the honest node is conquered by the attacker. SCR is defined as the number of conquered nodes out of the total amount of honest nodes. The smaller the SCR is, the better a Sybil defense algorithm is.

B. Simulation results

Since it is possible that an honest region may consist of multiple communities, the number of honest communities may affect the accuracy of the Sybil gateway detection algorithm. We made one Sybil community and one (up to five) honest communities. The result is given by Fig. 5: with the number of honest community increasing, the accuracy of our Sybil gateway detection algorithm also increases.

In order to reduce the computing time, we filter out a group of suspicious edges by their betweenness. From the light-colored bars of Fig. 6, we can see that the number of suspicion dramatically drops with the increase of betweenness threshold. But the majority of attack edges (Sybil edges) are still captured by the suspicion group, as shown by the dark-colored bars of Fig. 6.

Based on the results of the betweenness experiment, we produce a derived experiment: what could happen if we directly delete all of the suspicion edges without using our gateway-verification algorithm? We created an honest community and a Sybil community, and then, we remove the suspicion edges. SybilGuard is used to detect Sybils. The results show that directly deleting suspicions has a negative effect on accuracy, which also indicates the necessity of using our gateway-verification algorithm. One possible explanation is that the suspect edge group contains too many regular edges (about 90% in Fig. 6); removing these edges fundamentally changes the structure of the network, and it causes some honest nodes to become closer to the Sybil community.

Next, we focus on the accuracy of our proposed algorithms. We examined the impacts of the number of attack edges on SCR, as shown by Fig. 7. The more attack edges are created, the more nodes are conquered. The result of SS+GB is better than that of SG, which means the performance of SG has increased after using our Sybil gateway-breaking algorithm. The SCR value of our accepted nodes (SS+GB) is about 4% less than that of SG. By further checking identities of rejected

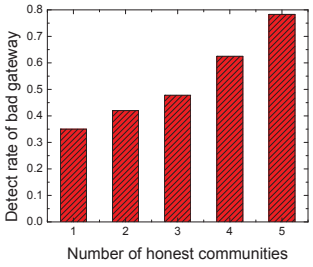


Fig. 5. The impacts of the number of honest communities

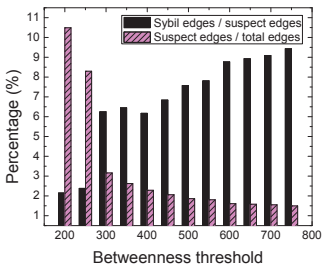


Fig. 6. The impacts of the betweenness threshold

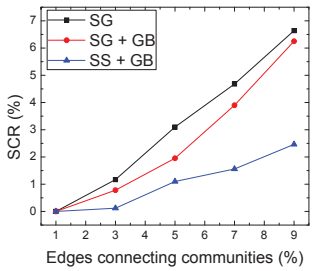


Fig. 7. The impacts of the number of attack edges.

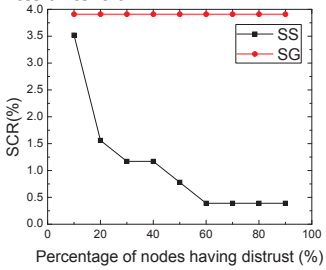
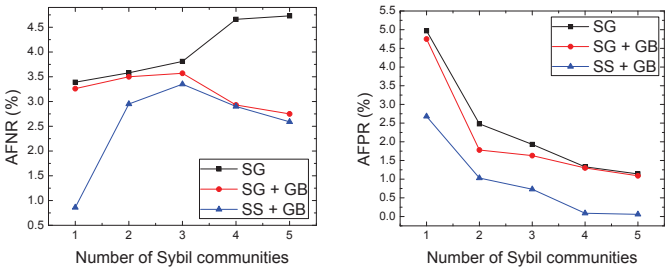
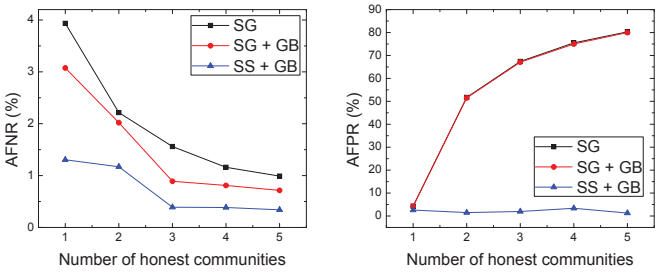


Fig. 8. The impacts of the number of distrust relations.



(a) avg. false negative rate. (b) avg. false positive rate.

Fig. 9. The impacts of the number of Sybil communities.



(a) avg. false negative rate. (b) avg. false positive rate.

Fig. 10. The impacts of the number of honest communities.

nodes, we find that our algorithm does eliminate some Sybil nodes near attack edges, which are accepted by SG.

Since SS is based on both trust and distrust relations, our next tested factor is the number of distrust edges, as shown in Fig. 8. Clearly, with the growing number of distrust edges, the accuracy of our algorithm will be increased. Moreover, by letting 20% nodes having one distrust edge, the SCR of SS is about 3.5% less than that of SG.

Our next consideration is the number of Sybil communities shown in Fig. 9. We kept one honest community, and gradually add more Sybil communities. For SS, there are approximately 4.75% distrust relations among all the relations. With the growth of Sybil communities, the AFNR of both algorithms increases in the beginning, and AFPR drops. However, when the number of Sybil communities becomes four and five, the AFNR values of SG+GB and SS+GB decrease. The reason is that with the growing number of Sybil communities, the AFNR values of both SG and SS increase, while the result of GB becomes more accurate. The integrated effects of them cause the AFNR to increase firstly and then decrease.

Our last consideration is the impacts of the number of honest communities on the accuracy of the three algorithms. In this part of simulation, we have one Sybil community and one (up to five) honest communities, with 256 nodes in each one. The simulation results support our viewpoint that the existing Sybil defense algorithms have high false positive rates when honest users cluster into multiple communities. From Fig. 10, we can see that both AFPR and AFNR of SS+GB is better than that of SG or SG+GB. Moreover, after adopting the gateway-breaking algorithm, the AFNR value of SS is decreased.

VII. Conclusion

We consider the problem of Sybil attacks in mobile social networks. Unlike traditional Sybil-defenses, we use a more realistic model that the social network of honest users may not be fast-mixing. Based on this model, we propose a new Sybil defense algorithm by exploring both trust and distrust relations. Our solution is lightweight: users only need to carry two small social profiles instead of the whole graph. Our algorithm solves the problem of high false positive in the existing Sybil defense algorithms. For increasing the accuracy, we propose a gateway-breaking algorithm, which can be used by any social networks-based Sybil defenses. Extensive simulations prove the significant performance of our algorithms.

References

[1] I. Constandache, X. Bao, M. Azizyan, and R. Choudhury, "Did you see Bob?: human localization using mobile phones," in *ACM MobiCom*, 2010.

[2] A. Mohaisen, A. Yun, and Y. Kim, "Measuring the mixing time of social graphs," in *ACM IMC*, 2010.

[3] B. Viswanath, A. Post, K. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *ACM SIGCOMM*, 2010.

[4] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *ACM IPSN*, 2004.

[5] M. Demirbas and Y. Song, "An RSSI-based scheme for sybil attack detection in wireless sensor networks," in *IEEE WOWMOM*, 2006.

[6] C. Piro, C. Shields, and B. Levine, "Detecting the sybil attack in mobile ad hoc networks," in *IEEE Securecomm*, 2006.

[7] D. Quercia and S. Hailes, "Sybil attacks against mobile users: friends and foes to the rescue," in *IEEE INFOCOM*, 2010.

[8] G. Danezis and P. Mittal, "Sybilinfer: detecting sybil nodes using social networks," in *NDSS*, 2009.

[9] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *ACM SIGCOMM*, 2006.

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

[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *ACM CHI*, 2010.

[12] N. Chiluka, N. Andrade, J. Pouwelse, and H. Sips, "Leveraging trust and distrust for sybil-tolerant voting in online social media," in *ACM PSOSM*, 2012.

[13] P. Marsden, "Egocentric and sociocentric measures of network centrality," *Social networks*, 2002.