Keep the Fakes Out: Defending Against Sybil Attack in P2P Systems

Kan Chen(✉), Peidong Zhu, and Yueshan Xiong

National University of Defense Technology, Changsha, China
{jeffee,pdzhu,ysxiong}@nudt.edu.cn

Abstract. Sybil attack is one of the major threats in distributed systems. A number of colluded Sybil peers can pollute and disrupt the system's key functions. The main idea of defense against Sybil attack is to distinguish the Sybils according to specific rules. Prior works are all limited by attack edges, the connections between normal and Sybil peers. The problem is that the number of attack edges could be huge, resulting in low accuracies. Besides, Sybil peers always present in groups and bring about the bridge problem, which is always ignored. In this paper, we propose KFOut, a light weighted framework for Sybil detection. At the heart of KFOut lie a trust model of social relations and a security mechanism of path notification of K-different paths, which can conquer the bridge problem effectively. We prove through experiments that KFOut can accept normal peers and reject Sybil peers both with high accuracies.

Keywords: Sybil attack · Detection · Social relation · P2P system

1 Introduction

Due to the nature of P2P systems, such as anonymity [1] and self-organization [2, 3], many applications are vulnerable to Sybil attack, which refers to the threat resulting from the arbitrary use of fake identities.

In P2P systems, every user is identified as a peer. Generally a single user creates only one peer, which makes a fair environment for everyone. However, in some cases if a malicious user creates a number of fake peers, he may break down the fairness and take advantages in system functions, such as voting [4, 5] and rating [6]. In using of these fake peers, the adversary may disrupt the key functions of the system. And even worse, if he controls enough fake peers, the trust relationship will be manipulated and the whole system may be in charge.

It has been proven that the only way to eliminate Sybil attack is to build a trusted identify authority [7], in which every user's real life identity is kept and identified. However, it's unpractical lying in some implementation problems and information leaking concerns. As a result, researchers refer to defense mechanisms to restrict the corruptive influences. Leveraging social network turns out to be the most effective approach [8, 9].

In this paper, we present KFOut, a decentralized Sybil-resilient protocol. We aim to detect Sybil peers with social relations. Honest peers are accepted, while the Sybil peers are rejected. The contribution of this paper is three fold. First, KFOut presents

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 183–191, 2015.
DOI: 10.1007/978-3-319-23829-6_14

high accuracies both in accepting honest peers and rejecting Sybil peers. Second, we efficiently solve the bridge problem, which refers to the problem that some Sybil peers act as bridges to make the other Sybil peers to be accepted. Third, KFOut is lightweight, which is essential in networked systems.

The rest of this paper is organized as follows. In Sect. 2, we introduce the system models. Key thoughts and the details of KFOut are described in Sect. 3. The performance evaluation results are presented in Sect. 4. Related works are reviewed in Sect. 5. And finally, discussion and conclusion are in Sect. 6.

2 System Model

In P2P networks, users are represented as peers. Every peer is a digital identity of a user. However, it's not necessary that every user has only one peer. Our system includes N peers, manipulated by M users ($N \geqslant M$). For the rest of this paper, we use peer and user interchangeably unless explicitly mentioned.

We're motivated to reduce the power of Sybil attack by rejecting Sybil peers. This is fulfilled with the use of social relations. Through communication and participation in system affairs, peers build trust relations with each other. We believe that every peer has his experiences to distinguish Sybil peers, and an honest peer would not like to trust and interact with a Sybil one. If a peer trusts another, a relationship is built. In our system, every peer defines a list of trust relations according to his historical interactions and local experiences. The peers in the list are named as the neighbors or friends.

The relations in our model are built on daily interactions. Different from the traditional interactions of sending and receiving service, the socialized interactions can't be fulfilled only by machines or agents. Instead, it takes human efforts. Thus although a Sybil user can create many Sybil peers, limited by time, energy and other resources, he cannot maintain social relations for all of them. In fact, in practice, a Sybil user only focuses on one or two certain peers, and uses them to interact with others. These peers are known as the pretended peers. As for the rest, they are poorly connected and named as the fake peers. Since it takes human efforts to maintain a pretended peer, the count of pretended peers would be small. By contrast, the count of fake peers can be huge since it doesn't need many efforts to register a peer. This assumption has been exploited and examined in many other works [8–11]. In this paper, we use it as the basic hypothesis.

In our model, a peer is chosen to be the verifier. Once the verifier has decided to believe that a peer is honest, we say that the verifier accepts that peer. Otherwise, we say that the verifier rejects that peer. A good protocol aims to accept most honest peers and reject most Sybil peers. In a centralized setting, the server can perform as the verifier. However, in a decentralized setting such as P2P networks, every peer could be his own verifier.

It's noticed that we don't aim to figure out all Sybil peers. Since the power of Sybil attack is determined by the number of Sybil peers, and the majority of them are fake peers. If we detect and reject the fake ones effectively, the rest pretended peers are powerless to launch an attack. So in this paper, we mainly focus on the detection of fake peers.

3 Protocol Design

In this section we first give the definition of K-similar paths, and then describe our protocol in details.

3.1 K-Similar Paths

We detect Sybil peers on the basis of social relations. All the relations construct a social network, which is described as a social graph G. We use the trust paths as clues to prove honesty. If a peer has many paths linking from others, it indicates that he's trustable and would not like to be a Sybil peer.

However, the problem is that the Sybil peers never present along, but in groups. The pretended peers may get enough trusts and share with the fake ones. Here the pretended peers act as bridges, so we name the problem as the bridge problem.

The bridge problem is crucial because it disrupts the effectiveness of detection methods. However, in many prior works it's often ignored. We conquer the bridge problem through extra restrictions of trust paths.

For two paths P_1 and P_2, $P_1 = \{v_1, v_2 \ldots v_n\}$ and $P_2 = \{u_1, u_2 \ldots u_n\}$, if $v_1 = u_1$, $v_2 = u_2$, … $v_i = u_i$, we say that P_1 and P_2 satisfy i-similar. If K is the max value of i, we define the coefficient of similarity (*cos*) of P_1 and P_2 is K.

For example, if $p_1 = \{q, \mathrm{s}, t\}$, $p_1 = \{q, \mathrm{s}, p\}$, then P_1 and P_2 satisfy 1-like and 2-like, and the *cos* is 2.

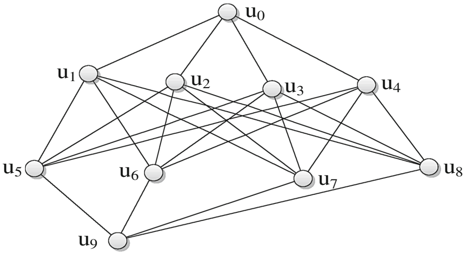

Fig. 1. An example topology of Sybil group

Let's explain how to use the *K*-similar paths to solve the bridge problem with the topology of Fig. 1. We assume that u_0 has got a path P with a length of n. The other peers can share paths from u_0. For example, u_1 can get $\{P, u_0\}$ from u_0. Then u_5 can get four paths $\{P, u_0, u_1\}$, $\{P, u_0, u_2\}$, $\{P, u_0, u_3\}$ and $\{P, u_0, u_4\}$ from u_1, u_2, u_3 and u_4. Finally u_9 gets sixteen paths totally. And this explains how the Sybil peers share paths and how the bridge problem happens.

There're four paths of u_9 originating from u_1, $\{P, u_0, u_1, u_5\}$, $\{P, u_0, u_1, u_6\}$, $\{P, u_0, u_1, u_7\}$ and $\{P, u_0, u_1, u_8\}$. Now let's assume that only paths with smaller *cos* than K can be taken into account. It's clear that all the four paths above are *(n + 2)*-similar. If we define K > n + 2, then all of them will be accepted. But if we define k = n + 2, then only one of them will be accepted and u_9 can finally get four paths only. What's more, if we denote $K < n$, then all these peers can only get one path because the *cos* between any two paths is larger than *K*.

3.2 The Procedure of Notification

Our framework is built in a decentralized environment. Initially, everyone only knows its direct neighbors, but have no knowledge about others. The first step is to inform the others the paths leading to the verifier. We achieve this through a path notification procedure.

Every peer should define its own methods to encrypt and decrypt. We represent them as *encrypt*() and *decrypt*() respectively. They are out-of-band, any approach is feasible.

First, the verifier v initials a path $P_0 = \{v\}$ and a token T_0. After encryption on T_0, both the path and the encrypted token are sent to v's neighbors in notification messages.

If a peer u_i receives a notification message, it should first check the effectiveness of the embedded path *P*. There're some relevant conceptions need to define first.

(a) The length of *P* is shorter than the max hop ϕ.
(b) $\mathrm{u_i}$ does not exist in *P*.
(c) The *cos* between *P* and any path in $\mathrm{u_i}$'s path table is smaller than K.
(d) P is shorter than its K-similar path in $\mathrm{u_i}$'s path table.

If *P* is effective to u_i, then $\mathrm{a} \cap \mathrm{b} \cap (c \cup d)$ should be satisfied. In that case, u_i need to update its path table. First *P* is added in. Then all the *K*-similar paths are discarded if exist. Once u_i finished updating, the updates need to be propagated to the neighbors in new notification messages.

The new notification messages also consist of both of the new path and the new token. The new path is generated by appending u_i to the end of *P*. And the new token is a re-encryption on the original token. Anyone that receives such a message should repeat the procedures above until the path become ineffective.

3.3 The Procedure of Aggregation and Verification

Once all peers have finished notification and no long receive any notification messages, the verifier can carry out admission control to decide which one to accept.

Anyone who wants to be verified first submits its path table to the verifier. The verifier will decide whether to trust the peer or not according to the count of paths submitted. But first, the credibility of the paths needs to be checked because the Sybil peers may disobey the rules and make up inexistent paths arbitrarily.

Two sets, *VS* and *US*, are defined to store the paths has been verified and wait to be verified respectively. *VS* is initialized as *null*, while *US* consists of all the submitted paths.

Every time if *US* is not empty, the shortest path is chosen and validated with its token as the algorithm shown in Algorithm 1.

Algorithm 1. validate the credibility of a path

Input: path $p_i = \{u_1, u_2 \ldots u_n\}$ and its token T_i
Output: True or False

```
Delete $P_i$ from $US$
If n=2, $u_2$ decrypts on $T_i$ and sends v the result $T'$.
    If $T'=T_0$, return True; else return False;
Else do
    If $P_{i-1}=\{u_1, ..., u_{n-1}\}$ is NOT in $VS$, return False;
    Send $P_i$ and its token $T_i$ to the last node $u_n$
    If $P_{i-1}$ is not in $u_n$'s path table, return False.
    Decrypt on $T_i$ and compare the result $T'=decrypt_{un}(T_i)$ with $P_{i-1}$'s token $T_{i-1}$. If $T' \neq T_{i-1}$, return False;
End else
Add $P_i$ into VS, Return True;
```

If *US* becomes *null*, it indicates that all the paths have been validated and all the credible ones have been kept in *VS*. Then the verifier can distinguish the Sybil peers according to the counts of credible paths. A threshold δ is defined. A peer is accepted as long as it provides more than δ credible paths. It's worthy to say that the value of δ is adjustable. A bigger δ rejects more peers while a smaller δ accepts more peers. An ideal protocol accepts most honest peers but rejects most Sybil peers.

4 Experiment Result

In this section, we evaluate the effectiveness of KFOut in synthetic networks. The results are discussed below.

4.1 Experimental Methodology

We synthesize our networks as the methodology of Barabasi and Albert [12]. A small fraction of peers are randomly chosen to be the pretended peers. Additional fake peers are introduced to establish Sybil group, which is connected as the same methodology.

Two factors are used to characterize the system performance, the accept rate of honest peer (AR) and the reject rate of Sybil peer (RR). We call them accept rate and reject rate for short respectively. Our goal is to achieve high rates for both of them.

First we test the performance in different scales of network. We generate three networks: a 1000-peer network, a 5,000-peer network and a 10,000-peer network. The static properties of these networks are shown in Table 1. 10 % of the peers are chosen to be the pretended peers. Fake peers are introduced with an equal number.

Table 1. Static properties and average node degrees of synthetic networks

Peers	Links	Avg. degree
1,000	20,320	19.352
5,000	103,027	19.672
10,000	197,609	19.820

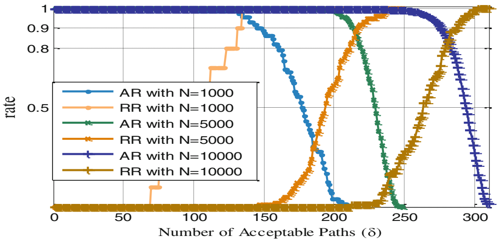

Fig. 2. Accept Rate(*AR*) and Reject Rate(*RR*) with different scales(*N*) as a fraction of the threshold value of acceptable paths(δ).

Figure 2 measures the fractions of *AR* and *RR* under these three networks. The value of δ represents the threshold of acceptable paths. A peer can be accepted as long as it provides more than δ credible paths. In the beginning, the value of δ is small, so almost all the peers can be accepted. As the increase of δ, more Sybil peers are rejected because the lack of relations. Some honest peers are also rejected for the same reason. Finally, the value of δ has increased too much, both the honest and the Sybil cannot get enough paths. So the *RR* is high but the *AR* is low.

It's obvious that a higher *AR* results in a lower *RR*. However, our goal is to gain high values for both of them. So a proper δ is needed to get a balance. As shown in Fig. 2, for each network, the two curves cross with each other. We define the best performance at the cross point, where *AR* and *RR* are similar to each other. In the rest of this paper we use the same definition when referring to the best performance.

4.2 Impact of the Count of Sybil Peers

In our framework, there're two kinds of Sybil peers, the pretended and the fake. We also investigate the impacts of them to the performance of KFOut respectively.

We synthesize a network with 5000 peers. First 1000 peers are chosen to be the fake peers. The number of the pretended peers is increased from 0 to 1000. Then 1000 peers are chosen to be the pretended peers, and the number of the fake peers is increased from 10 to 3000. Figure 3 depicts the distribution of *AR* and *RR*.

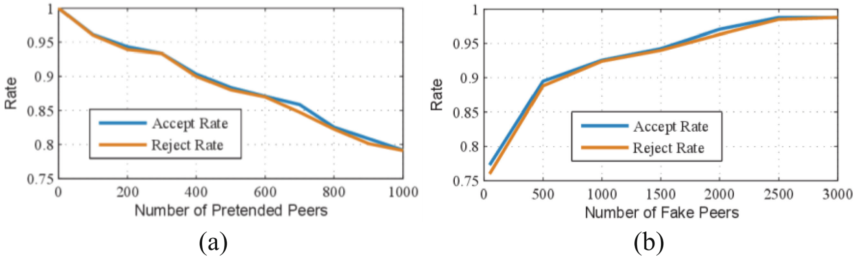

Fig. 3. Accept Rate(*AR*) and Reject Rate(*RR*) under the number of pretended peers (a) and fake peers (b)

We can see that the curves give a decline as the increase of the count of pretended peers. Since the pretend peers are similar to honest peers in behavior and connections, it's easy to understand that if a Sybil user creates more pretended peers, he get more convenience to manage them to be accepted, because there're more paths to share. We can see that the experimental results are inspiring. Even if 20 % of the peers are pretended, our system can still get a promising result, both the *AR* and the *RR* are as high as 80 %.

It's interesting to see that as the increase of the count of fake peers, both the *AR* and the *RR* increase too, which is different to the pretended peers. That's because the additional Sybil peers only contribute to the total number of Sybil peers, but not the connections with the honest. And it may result in a disproportion between honest peers and Sybil peers.

The experiment results suggests that for a Sybil user, if he wants to enhance the power of his Sybil peers, he should focus on the pretended peers and find a proper count for the fake peers. If the fake peers are few, it's hard to detect for the defending system, but the power of the Sybil group is also limited. On the contrary, if the number is high, the Sybil group can be powerful but is easy to be detected.

5 Related Works

Although Sybil attack is defined nearly by Douceur [7], it has been universal in P2P systems long before that. Despite the fact that it's not possible to eliminate Sybil attack completely nowadays, many works have been attempted to mitigate the corrupt threat.

Resource testing is built on the assumption that every identity consumes some resources and a user's resources are limited. So some fierce tests, such as check for computing ability, storage ability and network bandwidth, the count of IP addresses, are proposed on the identities [13, 14]. The intensity of every test is designed delicately so the ordinary users can afford but the Sybil users would not because they have to handle for multiple identities. However, this method can only be used in some specific fields and taking such a test would be exhausted because the test machine consumes the same resource as the machine being tested. Besides, the facilitation of NAT and botnet has also made it impossible to detect the Sybil through that way. So researchers turn to defense mechanisms to reduce the influence of attack.

SybilGuard [11] is the first attempt to deal with Sybil attack with social network. It assumes that malicious users can create many identities but few trust relations, thus the poor connectivity of the Sybil peers may result in a disproportionately small cut between honest peers and Sybil peers in the graph. This assumption has also been adopted by many other works. SybilGuard is decentralized and has been improved as SybilLimit [15], which leverages the same insight as SybilGuard but provides more precise results.

Gatekeeper [10] is another decentralized protocol. It uses a ticket distribution to detect Sybil peers. An admission controller randomly chooses multiple peers as ticket sources to distribute tickets. Each peer who receives tickets should keep one and propagate the others to its direct neighbors. When the ticket distribution is finished, the admission controller examines the number of ticket that the others receive. Sybil peers are separated because of the poor connectivity.

SybilInfer [16] is a typical centralized algorithm that uses a Bayesian inference approach to distinguish the Sybil. The main idea is that in the social network, the mixing between honest peers is fast, while that between honest peers and Sybil peers is slow. So the problem of computing the set of honest peers can be related to the problem of computing the bottleneck cut of the graph that result in slow mixing.

Another centralized algorithm is SumUp [17], which uses adaptive vote flow to prevent from arbitrarily manipulating voting results. The goal of SumUp is to use a Sybil resilient manner to collect votes, some of which are from Sybil identities. The number of l votes is limited to no more than the number of attack edge.

Sybil attack is not unique in P2P systems. Many other systems are vulnerable to Sybil attack too. There're also some researches aiming to using the system features to deal with Sybil attack on a system basis, such as in commercial sites [6], recommender systems [18], Ad hoc [19] and wireless networks [20].

6 Conclusion and Discussion

Sybil attack is prevalent in P2P systems. Many fields and applications are vulnerable to Sybil attack. The openness of Internet makes it easy to launch but difficult to detect. In this paper, we presented KFOut, a decentralized defending protocol against Sybil attack. KFOut leverages social relations to detect Sybil peers. The basic assumption is that the fake peers lack connections with the honest peers, which results in an unsymmetrical topology in social graph. Simulation results demonstrate that KFOut can detect Sybil peers with a very high accuracy. Even in the worst cases, KFOut can accept most honest peers and reject most Sybil peers.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No. 61170285 and Grant No. 61379103.

References

1. Xiao, R.Y.: Survey on anonymity in unstructured peer-to-peer systems. J. Comput. Sci. Technol. **23**(4), 660–671 (2008)
2. Aberer, K.: Self-organization and P2P systems. IEEE Intell. Syst. **18**(4), 79–81+85 (2003)
3. Khan, S.K.A., Tokarchuk, L.N.: Interest-based self-organization in group-structured P2P networks. In: 2009 6th IEEE Consumer Communications and Networking Conference, CCNC 2009, January 10–January 13 2009. Institute of Electrical and Electronics Engineers Computer Society, Las Vegas (2009)
4. Bocek, T., Peric, D., Hecht, F., Hausheer, D., Stiller, B.: PeerVote: a decentralized voting mechanism for P2P collaboration systems. In: Sadre, R., Pras, A. (eds.) AIMS 2009 Enschede. LNCS, vol. 5637, pp. 56–69. Springer, Heidelberg (2009)

5. Yang, B., Song, G., Zheng, Y.: The analysis and enhancement of voting behaviors in P2P networks. In: 2010 International Symposium on Intelligent Information Technology and Security Informatics, IITSI 2010, April 2–April 4 2010, pp. 407–410. IEEE Computer Society, Jinggangshan (2010)
6. Bhattacharjee, R., Goel, A.: Avoiding ballot stuffing in eBay-like reputation systems. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems, Philadelphia, Pennsylvania, USA (2005)
7. Douceur, J.R.: The Sybil attack. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems (2002)
8. Jiang, J., Shan, Z., Sha, W., Wang, X., Dai, Y.: Detecting and validating Sybil groups in the wild. In: 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012, June 18–June 21 2012, pp. 127–132. IEEE Computer Society, Macau (2012)
9. Viswanath, B., Post, A., Gummadi, K., Mislove, A.: An analysis of social network-based Sybil defenses. In: SIGCOMM 2010: Proceedings of the ACM SIGCOMM 2010 Conference on SIGCOMM, vol. 40, pp. 363–374 (2010)
10. Tran, N., Li, J., Subramanian, L., Chow, S.S.M.: Optimal Sybil-resilient node admission control. In: IEEE INFOCOM 2011, pp. 3218–3226. Institute of Electrical and Electronics Engineers Inc., Shanghai (2011)
11. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.: SybilGuard: defending against Sybil attacks via social networks. In: ACM SIGCOMM, pp. 267–278. Association for Computing Machinery (2006)
12. Kimmo, K., et al.: Emergence of communities in weighted networks. Phys. Rev. Lett. **99** (22), 228701 (2007)
13. Cornelli, F., Damiani, E., Samarati, S.: Implementing a reputation-aware Gnutella servent. In: Proceedings of the International Workshop on P2P Computing (2002)
14. Freedman, M.J., Morris, R.: Tarzan: a peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, (2002)
15. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: SybilLimit: a near-optimal social network defense against Sybil attacks. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy (2008)
16. Danezis, G., Mittal, P.: SybilInfer: detecting Sybil nodes using social networks. In: NDSS 2009, San Diego, CA (2009)
17. Tran, N., Min, B., Li, J., Subramanian, L.: Sybil-resilient online content voting. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, pp. 15–28. USENIX Association, Boston (2009)
18. Noh, G., Kang, Y., Oh, H., Kim, C.: Robust Sybil attack defense with information level in online Recommender Systems (2013)
19. Park, S., Aslam, B., Turgut, D., Zou, C.C.: Defense against Sybil attack in the initial deployment stage of vehicular ad hoc network based on roadside unit support. Secur. Commun. Netw. **6**(4), 523–538 (2013)
20. Abbas, S., Merabti, M., Llewellyn-Jones, D., Kifayat, K.: Lightweight sybil attack detection in MANETs. IEEE Syst. J. **7**(2), 236–248 (2013)