PAPER Special Section on Parallel and Distributed Computing and Networking

A Network Clustering Algorithm for Sybil-Attack Resisting

Ling XU[†a)], ***Nonmember***, **Ryusuke EGAWA**[††b)], **Hiroyuki TAKIZAWA**[†c)], ***and*** **Hiroaki KOBAYASHI**[††d)], ***Members***

SUMMARY The social network model has been regarded as a promising mechanism to defend against Sybil attack. This model assumes that honest peers and Sybil peers are connected by only a small number of attack edges. Detection of the attack edges plays a key role in restraining the power of Sybil peers. In this paper, an attack-resisting, distributed algorithm, named *Random walk and Social network model-based clustering* (RSC), is proposed to detect the attack edges. In RSC, peers disseminate random walk packets to each other. For each edge, the number of times that the packets pass this edge reflects the *betweenness* of this edge. RSC observes that the betweennesses of attack edges are higher than those of the non-attack edges. In this way, the attack edges can be identified. To show the effectiveness of RSC, RSC is integrated into an existing social network model-based algorithm called SOHL. The results of simulations with real world social network datasets show that RSC remarkably improves the performance of SOHL.

key words: *sybil attack, social network, attack edge, clustering*

1. Introduction

Peer to peer (P2P) is a model to organize large scale distributed systems. Unlike conventional Server-Client systems, in a P2P system, there is no central server, and users directly share resources with each other. This removal of central servers makes the system highly easy to extend.

However, the lack of central servers also makes P2P systems vulnerable to *Sybil attack* [1]. In a P2P system, each user can register a legal account. From the view of the system, this account is called a *peer*, and is used for the user to share resources. In a P2P system, there may exist malicious users whose objective is to break the system protocols. Due to the lack of central authorities, it is easy for a malicious user to obtain a large number of fake accounts. From the view of the system, these fake accounts are called *Sybil peers*. The malicious users can then control these Sybil peers to break the system laws. Sybil attack is a big threat to P2P systems, and thus has taken researchers great efforts to defend the system against the attack.

Recently, the *social network model* (SNM) [2], shown in Fig. 1, becomes to be frequently utilized to resist Sybil attack [3]. SNM assumes that, in a P2P system, honest peers and Sybil peers are only connected by a small number of edges, called *attack edges*. These attack edges form a small cut in the system. The existing SSR (SNM-base Sybil Resisting) algorithms utilize this small cut to probabilistically judge whether a peer is Sybil or not [3]. However, they cannot decide which peers are connected to the attack edges.

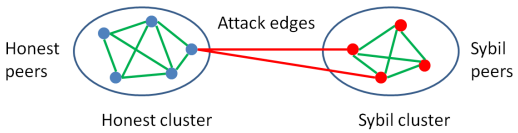


Fig. 1 Social network model.

The detection of the attack edge plays a key role in both detecting Sybil peers and restraining the communication from Sybil peers to honest peers. The motivation of this paper is hence to explicitly detect the attack edges. For this goal, we design an attack-resisting distributed algorithm, named *Random walk and SNM-based network Clustering* (RSC). To validate the potential of RSC in improving the performance of existing SSR algorithms, we integrate RSC into a *Sybil resisting One Hop Lookup* (SOHL) algorithm [4]. The resulting algorithm is called *RSC-based Sybil Resisting* (RSSR). We evaluate RSSR by simulations using real world network datasets, and show that RSSR achieves notable performance improvement from multiple aspects over SOHL. These results demonstrate that RSC can be generally used as a underlying component to improve the performance of SSR algorithms.

The rest of this paper is organized as follows. Section 2 introduces the social network model, the SSR algorithms, and a network clustering algorithm that enlightens our work. Section 3 elaborates the design of RSC. Having obtained RSC, Sect. 4 integrates it into SOHL to construct RSSR. Then, Sect. 5 evaluates the performance of RSC by comparing SOHL with RSSR by simulations. Finally, Sect. 6 concludes this paper.

2. Related Work

2.1 Social Network Model

In many P2P systems, a new peer needs to first contact a *bootstrap* peer – a peer that has existed in the systems, to

Manuscript received January 6, 2011.
Manuscript revised June 10, 2011.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980–8578 Japan.
[††]The authors are with Cyberscience Center, Tohoku University, Sendai-shi, 980–8578 Japan.
a) E-mail: xling@sc.isc.tohoku.ac.jp
b) E-mail: egawa@isc.tohoku.ac.jp
c) E-mail: tacky@isc.tohoku.ac.jp
d) E-mail: koba@isc.tohoku.ac.jp

DOI: 10.1587/transinf.E94.D.2345

join the systems. The bootstrap then helps the new peer to connect to the systems. Therefore, a malicious bootstrap can introduce a large number of other malicious peers to break the system protocol. This is a special kind of Sybil attack. To solve this problem, Danezis et al. [5] noticed that although a malicious bootstrap can introduce arbitrarily many Sybil peers, it cannot arbitrarily forge connection between honest peers and the new coming Sybil peers. This idea later evolves into SNM [6].

SNM assumes that in a P2P system, peers of the same types closely connect with each other, while peers of the different types loosely connect with each other. The edges connecting peers of different types are called the attack edges. The network can thus be largely divided into an honest cluster and a Sybil cluster, connected by a few number of attack edges.

2.2 Social Network Model-Based Sybil Attack Resisting Algorithms

Many SSR algorithms have been proposed in the recent years. SybilLimit [2], SOHL (Sybil Resisting One Hop Lookup) [4] and Whānau [7] aim to increase (decrease) the probability that honest peers accept honest (Sybil) peers. The basic principle of these algorithms is as follows. In these systems, since the number of attack edges is small, the attack edges forms a small cut. This cut reduces the probability that the random walk packets of Sybil peers reach honest peers, and therefore limits the communication between peers of different types. However, to the best of our knowledge, none of the existing SSR algorithms concentrates on the detection of the attack edges. A good summary of the existing SSR algorithms can be found in [3].

2.3 Sybil Resisting One Hop Lookup

Here, we give a brief introduction of SOHL [4]. SOHL will be used as the baseline to evaluate the algorithms proposed in this paper.

The objective of SOHL is to increase the number of honest peers, and decrease the number of Sybil peers, accepted by honest peers. Here, "peer v accepts peer u" means that v accepts to communicate with u. In SOHL, each peer disseminates a certain number of random walk packets, called *probing random walks*. Besides, each peer v maintains a set, named a *finger set*, denoted by $v.fingers$. $v.fingers$ contains the IDs of the destinations of the probing random walks of v. To decide whether to accept a peer x, v first checks whether x is in $v.fingers$. If x is not found, for each finger u in $v.fingers$, v asks u whether x is in $u.fingers$. x will be accepted by v if and only if x is contained in either the finger set of v, or in the finger set of any finger of v.

2.4 Random Walk Betweenness-Based Network Clustering

The problem of detecting the attack edges is similar to the problem of detecting the front peers (peers connecting to the attack edges) in the well studied network clustering problem. Newman [8] proposed a *random walk betweenness-based network clustering* (RWBC) algorithm, where the goal is to identify the front peers and thus the clusters in the network. Specifically, in RWBC, each peer disseminates one *absorbing random walk* packet to each of the other peers. Here, an absorbing random walk packet is a packet that moves in a random walk manner until it reaches its destination. For each peer v in the system, the number of times that absorbing random walks pass v reflects the *betweenness* of v. RWBC shows that the betweennesses of front peers are higher than those of the non-front ones. In this way, the front peers, and thus the clusters in the network, can be detected.

However, RWBC cannot be directly used to detect the attack edges in P2P system, because RWBC is a centralized algorithm, and is not attack-resistible.

3. RSC – Detecting Attack Edges

This section proposes RSC, which is designed to accurately detect attack edges. This paper assumes that the resources (CPU, memory, bandwidth, etc.) of the malicious users behind Sybil peers is finite. Therefore, the total number of Sybil peers in the system is comparable to that of honest peers.

3.1 A More Detailed Introduction of RWBC

Basically, RSC is a distributed, attack-resisting variation of RWBC. Therefore, first, we give a more detailed explanation of RWBC.

We use (s,t) *random walks* to denote the absorbing random walks that start from peer s and end at peer t. For each peer s, s disseminates one absorbing random walk to each of the other peers, t. The expected numbers of times that (s,t) random walks passing an edge $e = (v,u)$ from v to u and from u to v are denoted by $e.i_v^{(s,t)}$ and $e.i_u^{(s,t)}$, respectively. The absolute difference between $e.i_v^{(s,t)}$ and $e.i_u^{(s,t)}$ (i.e., $|e.i_v^{(s,t)} - e.i_u^{(s,t)}|$) is called the *partial betweenness of e for the* (s,t) *random walks*. Then, for all peer pairs, the summation of partial betweennesses of e (i.e., $\sum_{\forall(s,t)} |e.i_v^{(s,t)} - e.i_u^{(s,t)}|$) is called the *betweenness of edge e*. Let $E(v)$ and $N(v)$ denote the incident edges and neighbors of v, respectively. The summation of the betweennesses of the incident edges of v, namely

$$\sum_{\forall(v,u)\in E(v)} \sum_{\forall(s,t)} |e.i_v^{(s,t)} - e.i_u^{(s,t)}|, \tag{1}$$

is the *betweenness of peer v*.

RWBC shows that the betweennesses of front peers are higher than those of the other peers. Therefore, by computing the betweennesses of peers, front peers can be detected.

3.2 Basic Protocol of RSC

We estimate that in a SNM-based system, the betweennesses

of attack edges are averagely higher than those of the non-attack edges. Intuitively, in RWBC, front peers have high betweennesses because all random walks that connect peers of different types have to pass the front peers. All these random walks also have to pass the attack edges. Moreover, SNM ensures that the number of attack edges is small. Accordingly, the average number of random walks passing an attack edge, and thus the betweennesses of attack edges should be high.

Based on this estimation, RSC detects the attack edges by computing the betweenness of each edge, in a distributed manager. To do this, as with RWBC, peers disseminate absorbing random walks. In RSC, each peer has an unique ID, which is a hash value of its IP address. Each peer v maintains a *destination set*, denoted by $v.Des$. $v.Des$ stores the IDs of peers, to which v disseminates absorbing random walks. We discuss the way to construct the destination set in Sect. 3.4.

For each peer u in $v.Des$, v disseminates C absorbing random walks of length L to peer u, where L and C are two system parameters. In practice, each peer sets its L and C to be as large as possible according to their respective computing capacities. Each absorbing random walk rw contains the IDs of its source and destination, denoted by $rw.s$ and $rw.t$, respectively. Besides, for each incident edge $e = (v, u)$ and each set of (s, t) random walks, v holds two variables $v.u_-^{(s,t)}$ and $v.u_+^{(s,t)}$. These two variables are used by v to record the expected number of times that (s, t) random walks passes e from v to u and u to v, respectively. When rw enters (leaves) v from e, v increases $v.u_-^{(s,t)}$ ($v.u_+^{(s,t)}$) by one. In this way, $v.u_-^{(s,t)}$ and $v.u_+^{(s,t)}$ are approximations of $e.i_v^{(s,t)}$ and $e.i_u^{(s,t)}$ of RWBC, respectively. Then, v can compute the partial betweenness of e for the (s, t) random walks, denoted by $v.i_u^{(s,t)}$, as

$$v.i_u^{(s,t)} = |v.u_-^{(s,t)} - v.u_+^{(s,t)}|. \quad (2)$$

By summing up $v.i_u^{(s,t)}$ for all (s, t) pairs, the betweenness of edge e, denoted by $v.i$, can be computed. Algorithm 1 details the behaviors of each peer v in this basic RSC algorithm.

```
// Initiate:
foreach w ∈ v.Des do
    for i = 0; i < C; i++ do
        forms a (v, w) random walk message m = {s : v, t : t};
        sends m to a random neighbor;
    end
end
// Received a absorbing random walk m from a
   neighbor u:
if v == m.t then
    stop successfully;
else
    if v.u_-^(m.s,m.t) does not exist then
        v.u_-^(m.s,m.t) = v.u_+^(m.s,m.t) = 0;
    end
    v.u_+^(m.s,m.t) += 1;
    randomly chooses a neighbor w;
    v.w_-^(m.s,m.t) += 1;
    foreach (s, t) pair where v.u_-^(s,t) exists do
        v.i_u += Σ_{∀u∈N(v)} |v.u_-^(s,t) − v.u_+^(s,t)|;
    end
end
```

Algorithm 1: Basic RSC

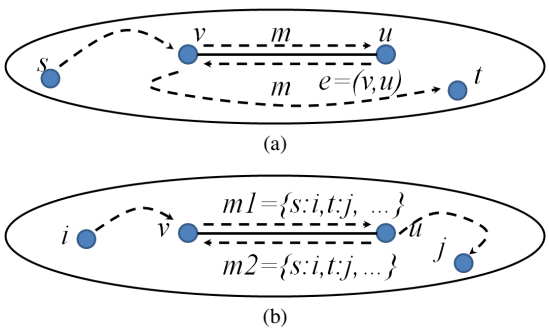


Fig. 2 Examples of Attack 1 and Attack 2.

3.3 Resisting Attacks

We have introduced the basic protocol of RSC, which can compute the edge betweenness in a benign environment. However, in malicious environments, Sybil peers would launch attacks to break the system protocols. Let $e = (v, u)$ be an attack edge, where v is honest and u is Sybil. We discuss solutions for typical attacks in this section.

Attack 1: Suppose that u has received a random walk from v. Instead of randomly selecting a neighbor peer as the next hop, u can always chooses v as the next hop for this random walk. In this way, as Eq. (2) shows, the betweenness of e will always be zero. An instance of Attack 1 is provided in Fig. 2 (a): peers v and u are connected by edge e, where v is honest and u is Sybil. To reduce the betweenness of e, on receiving a random walk packet m from v, u simply returns m back to v.

Attack 2: Sybil peers can reduce the betweennesses of attack edges by forging random walks. Suppose that a (i, j) random walk has passed through e from v to u. u can then form a random walk message m, setting the source and destination of this message to be the IDs of i and j, respectively, and send m to v through e. In this way, the (i, j) random walks that enter and leave v from e offset each other. As a result, the betweenness of e decreases. Figure 2 (b) gives an example of Attack 2: To reduce the betweenness of e, having received a (i, j) random walk $m1$ from v, u forges a (i, j) random walk $m2$ and sends $m2$ to v.

We design a special kind of random walks, called the *steered absorbing random walk* to defend against these attacks. For the simplicity of expression, we simply use *the*

```
// Initiate:
forms a Q message m = {s : v, t : t, Pre : v};
sends m to a random neighbor u;
// Received a Q message m from u:
if m.s! = hash(IP(u)) || m.Pre ∉ N(v) then
    stops incorrectly;
else
    if v == m.t then
        stops succeeded;
    else
        sends an A message m = {IPs : IPs(v)} to u;
    end
end
// Received a A message from u:
forms a Q message m = {s : v, t : t, Pre : u};
chooses a random IP from m.IPs(u);
sends m to IP;
```

Algorithm 2: The protocol of steered random walk on peer v

steered random walks to denote the steered absorbing random walks. Beside, we call the steered random walks that have honest sources and Sybil destinations the *first type steered random walks*, and call the other steered random walks the *second type steered random walks*. The steered random walk is designed to satisfy the following requirements:

Requirement 1: Each first type steered random walk increases the betweenness of at least one attack edge by one. This makes sure that attack edges still have a high betweennesses even under Attack 1.

Requirement 2: On receiving a random walk rw with $rw.s$ being the ID of peer s, peer v can identify whether this random walk is really originated by peer s or not. This enables honest peers to reject the forged random walks of Attack 2.

The steered random walk runs in the following way. Suppose that peer s disseminates a (s,t) random walk rw, and suppose that rw is currently on peer v. On receiving rw, v sends $IPs(v)$ – the set of the IP addresses of neighbors of v, to s. Then, s randomly chooses an IP from the IP set, and relays rw to the chosen IP. This process continues until rw arrives at t. The key to defend against the attacks is as follows. When peer v receives rw from a sender s', v relays rw to the next peer only if $rw.s$ is equal to $hash(IP(s'))$. Here, $IP(s')$ denotes the IP address of s', and $hash()$ can be any collision-free hash function. Algorithm 2 describes the behaviors of each peer v in the relay of steered random walks.

Now, we show that the steered random walk satisfies Requirements 1 and 2. As for Requirement 1, let rw be a first type steered random walk. Let k denote the number of times that rw traverses attack edges during its movement. No matter whether rw eventually reaches its destination or not (Sybil peers can arbitrarily stop the relay of rw), rw must finally stop in the Sybil region. This means that k is an odd number. Figure 3 provides an example showing this property. In this example, the dot dash line is a first type (s,t) random walk, denoted by rw. $ae1$, $ae2$ and $ae3$ are the attack edges of the system. The number of times that rw traverses the attack edges must be an odd number (three in this example).

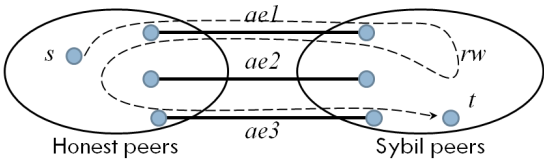


Fig. 3 A first type steered random walk.

Supposing that the first $k-1$ times of traverses cancel off each other completely, the left one traverse will still pass a certain attack edge from the honest peer side to the Sybil peer side. Therefore, the betweenness of at least one attack edge will be increased, and Requirement 1 holds. Then, according to the algorithm of the steered random walk, Requirement 2 holds naturally.

3.4 Discussion

Now, we discuss the problem of constructing the destination set for each peer. Intuitively, it would be better to have a large destination set, so that RSC can more accurately approximate the edge betweennesses. However, maintaining an accurate ID set of all peers is expensive in distributed malicious environments.

Fortunately, as shown in Sect. 3.3, for each peer, the key to detect attack edges is that a sufficient number of first type steered random walks are disseminated. This means that RSC only needs to ensure that each honest peer maintains a destination set that contains a sufficient number of Sybil peers. For this goal, for each honest peer v, once a peer u requires peer v for communication, v adds u to $v.Des$. Therefore, the number of Sybil peers in the destination set of v increases as long as Sybil peers try to communicate with v. In a long term, a sufficient number of first type steered random walks will be disseminated to detect attack edges.

4. RSSR

We have introduced RSC in the previous section. To show the potential of using RSC in improving the performance of existing SSR algorithms, this section proposes RSSR, a RSC-based Sybil resisting algorithm.

Basically, RSSR is a combination of RSC and SOHL. Recall that in SOHL, honest peers disseminate probing random walks to find each other. Let us denote by *escape rate* (ER) of random walks the average probability that a random walk disseminated by an honest peer enters the Sybil region. First, RSSR uses RSC to detect attack edges. Then, RSSR reduces the escape rate of probing random walks by preventing them from passing the detected edges. This en-

ables honest peers to accept more honest peers and less Sybil peers.

Specifically, the basic protocol of RSSR is just the same as that of SOHL, as shown in Sect. 2.3. The difference is in the way the probing random walks advance. In RSSR, let *rw* be a probing random walk. Suppose that *rw* is currently on peer v, and u is a neighbor of v. The probability that v selects u as the next hop is inversely proportional to the betweenness of edge (v, u). We call this kind of random walks the *waterfall random walks*. Since the attack edges are expected to have a high betweennesses, the escape rate of probing random walks can be decreased. Hence, the performance of SOHL can be improved.

In RSSR, the length of each probing random walk is $O(\log(n))$ and the number of probing random walks disseminated by each peer is $\Theta(\sqrt{M})$. Here, n is the number of honest peers and M is the number of edges among honest peers. For a peer w, let us define the *degree* of w as the number of neighbor peers of w, denoted by $k(w)$. And, we denote by *rw* a random walk of length $O(\log(n))$ that starts within the honest cluster. On one hand, SybilLimit [2] has shown that under SNM, *rw* would stay within the honest cluster during its movement with a high probability. Accordingly, for an honest peer u, the probability that u accepts Sybil peers is low. On the other hand, as shown in SOHL (Sect. 3.2, [4]), *rw* will stop at w with a probability $\Theta\left(\frac{k(w)}{M}\right)$. Therefore, by disseminating $\Theta(\sqrt{M})$ probing random walks, the probability that u accepts each of the other honest peers, v, is high:

$$\begin{aligned} & 1-\left(1-\Theta\left(\tfrac{k(v)}{M}\right)\Theta(\sqrt{M})\right)^{\Theta(\sqrt{M})} \\ > & 1-\left(1-\Theta\left(\tfrac{\sqrt{M}}{M}\right)\right)^{\Theta(\sqrt{M})} \\ = & 1-O(e^{-1})=\Omega(1). \end{aligned} \tag{3}$$

Similar to SOHL, we assume that each peer knows n and M. This is because the performance of SOHL and RSSR is not very sensitive to small changes in these two parameters, and each peer only needs a brief approximation of n and M [4].

5. Evaluation

In this section, we compare the performance of RSSR with SOHL by simulations to evaluate the effectiveness of RSC.

5.1 Simulation Configuration

The networks for simulations are designed in the following way. First, the networks of the honest and Sybil clusters are constructed separately. Then, attack edges are added to connect these two clusters. The networks of the honest clusters are extracted from the real world social network datasets, and the Sybil cluster networks are constructed using the Erdos-Renyi random network model [9].

We select three datasets of real social networks and extract subnetworks from them to construct the honest clusters. The datasets used are: *slashdot* for Slashdot.org, *epinions* for Epinions.com, and *hepTh* for the Hep-Th (High Energy Physics) archive [10]. Slashdot.org is a technology-related news website known for its specific user community. It allows users to tag each other as friends or foes. The slashdot dataset contains friend/foe links between the users in Slashdot.org. Epinions.com is a general consumer review website, where users can decide whether to trust each other. The epinions dataset represents the trust relationship among users. The hepTh dataset contains the citation relationship among papers in Hep-Th. In hepTh, if a paper i cites paper j, the graph contains an edge from i to j. We choose these datasets because they are available for public use, and all of them have been used for academic researches [11].

Table 1 Networks used in the simulations.

	dataset	|Honest Peers|	|Sybil Peers|
N1	epinions	1017	500
N2	hepTh	1053	500
N3	slashdot	1078	500
N4	epinions	1781	500

Then, we extract subnetworks from the datasets as the honest clusters. We use extracted subnetworks instead of the original datasets for two purposes: 1) to reduce the computing intensity of our simulations; 2) to make the peers more closely connected, according to the assumption of SNM (honest peers are closely connected). Specifically, we process each dataset in three steps. Intuitively, the connection density of the central part of a network is higher than that of its border part. Therefore, first, we find a *center node* of the dataset as an initial subnetwork. Here, the center nodes of a network is the nodes whose maximal distances to the other nodes are equal to the radius of this network. Then, gradually, we extend the current subnetwork in a breadth-first search manner. Finally, we remove from the subgraph the nodes with degrees less than three, as SybilLimit [12] does. After these three steps of processing, we obtain a subnetwork, in which the peers are closely connected. Four networks constructed in this way are used in the simulations, as shown in Table 1.

Having created the honest and Sybil clusters, we connect these two clusters by $|AE|$ attack edges. That is, each time one random honest peer and one random Sybil peer are chosen to connect, and this procedure is repeated $|AE|$ times.

During the simulations for RSSR and SOHL, the length of each probing random walk is $\log(n)$, and the number of probing random walks disseminated per peer is $\sqrt{M}$.

5.2 Evaluation Metrics

The performance is evaluated by four metrics: ER (escape rate), *HVS* (honest peer number versus Sybil peer number), *ANHP* (average number of honest peers) and *ANSP* (average number of Sybil peers). HVS denotes the ratio of the number of honest peers to the number of Sybil peers in the view of an average honest peer:

$$HVS = \left\langle \frac{|\text{honest peers in } v.view|}{|\text{Sybil peers in } v.view|} \right\rangle, \ \forall v \in HP.$$

ANHP (ANSP) denotes the percentage of honest (Sybil) peers in the view of an average honest peer:

$$ANHP = \left\langle \frac{|\text{honest peers in } v.view|}{|HP|} \right\rangle, \ \forall v \in HP.$$

$$ANSP = \left\langle \frac{|\text{Sybil peers in } v.view|}{|SP|} \right\rangle, \ \forall v \in HP.$$

5.3 Simulation Results and Discussion

Figure 4 shows that in the simulations, the average betweenness of the honest edges is lower than that of the attack edges. In each sub-figure, the two curves represent the averages of the betweennesses of honest edges and attack edges, respectively. We can see that when the number of attack edges is low, the average betweenness of attack edges is much higher than that of honest edges. Moreover, the average betweenness of attack edges decreases as the number of attack edges increases. These results match well with our assumption and explanation of the edge betweenness in Sect. 3.2.

Figure 5 shows that the ER of RSSR is usually much lower than that of SOHL. The four sub-figures show the results obtained from the four networks listed in Table 1, respectively. In each sub-figure, the horizontal axis represents the ratio of the number of attack edges to the number of honest peers in the system. Each sub-figure compares the ERs between RSSR and SOHL. We can see that as the number of attack edges increases, ERs of both RSSR and SOHL increase. However, ER of RSSR is much lower than that of SOHL. This is because waterfall random walks are less likely to go into the Sybil region than the normal random walks do.

Figure 6 shows the changes of ANHP and ANSP in RSSR and SOHL, as the number of attack edges increases. It shows that, first, as the number of attack edges increases, ANHP decreases and ANSP increases in both RSSR and SOHL, which is a direct result of the increasing of escape rate. Second, ANHP of RSSR is slightly lower than that of SOHL. This is because that in RSSR, the honest peers near the attack edges are less likely be visited by probing random walks. Third, however, RSSR averagely has much lower ANSPs, which is owed to the low escape rates of RSSR.

Figure 7 reveals the changes of HVS's in SOHL and RSSR, as the number of attack edges increases. Two features can be observed. First, when the number of attack edges is small, HVS of RSSR is much larger than that of SOHL. Intuitively, HVS is approximate to $\frac{ANHP}{ANSP}$. When attack edges are few, ANSP of RSSR is much smaller than that of SOHL, and hence HVS of RSSR is larger. Second, as the number of attack edges increases, HVSs decrease quickly in both SOHL and RSSR. In all the simulations, HVSs of SOHL and RSSR almost have overlapped each other when the number of attack edges is larger than 1000. The reason

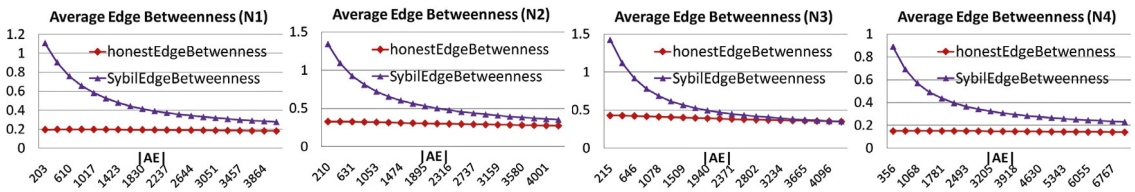


Fig. 4 The changes of the averages of edge betweennesses.

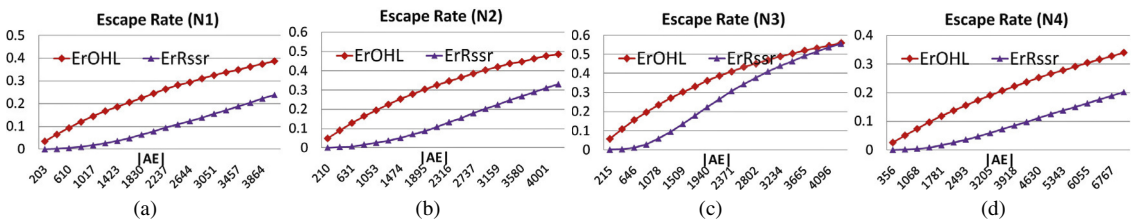


Fig. 5 Comparison between the ERs of RSSR and SOHL.

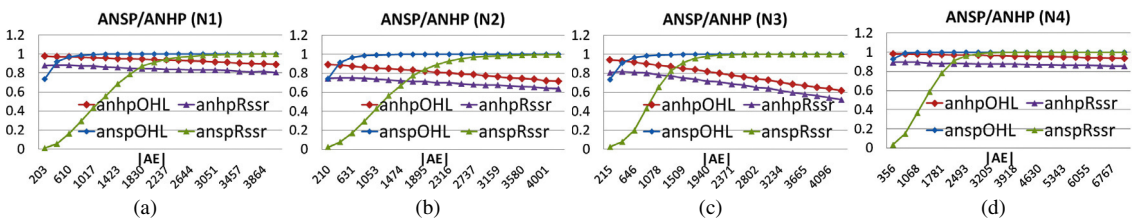
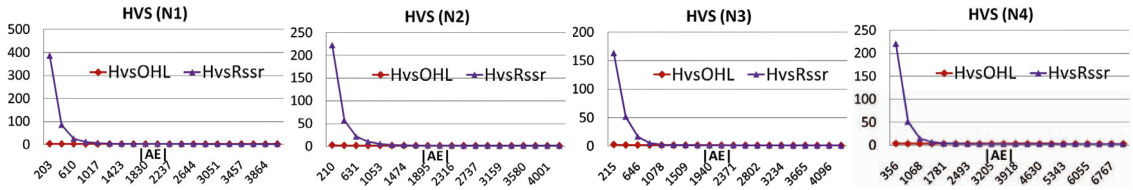


Fig. 6 Comparison between ANHPs and ANSPs of RSSR and SOHL.



(a) (b) (c) (d)

Fig. 7 Comparison between HVSs of RSSR and SOHL.

of this feature is also intuitive: HVS is approximate to $\frac{ANHP}{ANSP}$. As the number of attack edges increases, ANHP decreases and ANSP increases simultaneously, which causes the quick decrease of HVS.

In conclusion, compared with SOHL, RSSR can effectively increase the number of honest peers (reduce the number of Sybil peers) accepted by honest peers. RSSR achieves such improvement because waterfall random walks can effectively reduce the escape rate, which accordingly increases ANHP and decreases ANSP.

Note that RSC can be used not only for SOHL, but also for Whānau, SybilLimit, and many other existing SSR algorithms to improve their performance. Take Whānau for example. Whānau is a SSR algorithm that is a refined variation of SOHL. As with SOHL, Whānau also uses random walk as the basic protocol. The Sybil resisting performance of Whānau thus closely depends on the escape rate of random walks. Therefore, RSC and the waterfall random walk technique of RSSR can also be used for Whānau to decrease its Sybil accept rate.

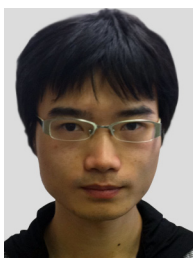
6. Conclusions

The motivation of this study is to design a distributed, attack-resisting algorithm to detect the attack edges in SNM. To this end, we designed RSC. To evaluate the effectiveness of RSC, it is embedded into SOHL. Then, the performance of the new algorithm is compared with that of the original one. The simulation results indicate that RSC notably improves the performance of SOHL from multiple dimensions. To the best of our knowledge, RSC is the first Sybil-resisting attack edge detecting algorithm designed for distributed systems.

As future work, we plan to extend this research from two directions. First, it is interesting to investigate the possibility of finding a general method to embed RSC to the existing SSR algorithms. Second, RSC uses the steered random walk to resist attacks. The problem of steered random walk is that it brings high message cost on each peer. Hence, the efficiency of RSC should be improved.

References

[1] J. Douceur, "The sybil attack," Peer-to-Peer Systems, pp.251–260, 2002.
[2] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against sybil attacks," IEEE Symposium on Security and Privacy, pp.3–17, 2008.
[3] B. Viswanath, A. Post, K.P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," SIGCOMM Comput. Commun. Rev., vol.40, pp.363–374, Aug. 2010.
[4] C. Lesniewski-Laas, "A sybil-proof one-hop DHT," Proc. 1st Workshop on Social Network Systems, SocialNets '08, pp.19–24, New York, NY, USA, 2008.
[5] G. Danezis, C. Lesniewski-Laas, M. Kaashoek, and R. Anderson, "Sybil-resistant DHT routing," Lect. Notes Comput. Sci., pp.305–318, 2005.
[6] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "SybilGuard: Defending against sybil attacks via social networks," SIGCOMM Comput. Commun. Rev., vol.36, pp.267–278, 2006.
[7] C. Lesniewski-Laas and M.F. Kaashoek, "Whānau: A sybil-proof distributed hash table," Symposium on Networked System Design and Implementation, San Jose, California, April 2010.
[8] M. Newman, "A measure of betweenness centrality based on random walks," Social Networks, vol.27, no.1, pp.39–54, 2005.
[9] Networkx. http://networkx.lanl.gov/contents.html
[10] http://arxiv.org/archive/hep-th
[11] http://snap.stanford.edu/data
[12] H. Yu, P.B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against sybilattacks," Tech. Rep. TRA2/08, National Univ. of Singapore, School of Computing, March 2008. http://www.comp.nus.edu.sg/~yuhf/sybillimit-tr.pdf, 2008.



Ling Xu is a Ph.D. student in Graduate School of Information Sciences in Tohoku University. He received his B.E. Degree in Computer Science, Wuhan University of Technology, and M.E. Degree in Information Science, Tohoku University. He is interesting in the analysis and design of distributed algorithms for P2P systems, social network systems and other distributed systems.

Ryusuke Egawa received the B.E. degree, Master Degree on information sciences from Hirosaki University in 1999, 2001, respectively, and the Ph.D. degree on information sciences from Tohoku University in 2004. He is currently an assistant professor of Cyberscience Center, Tohoku University. His research interests include computer architecture, VLSI design and high-performance computing.

Hiroyuki Takizawa is currently an associate professor of Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computing systems and their applications. He received the B.E. Degree in Mechanical Engineering, and the M.S. and Ph.D. Degrees in Information Sciences from Tohoku University in 1995, 1997 and 1999, respectively. He is a member of the IEEE CS, and the IPSJ.

Hiroaki Kobayashi is currently Director and Professor of Cyberscience Center and Professor of the Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computer architectures, grid and P2P computing, and multimedia applications. He received the B.E. Degree in Communication Engineering, and the M.E. and D.E. Degrees in Information Engineering from Tohoku University. He is a senior member of IEEE CS, and a member of ACM and IPSJ.