A Survey of Sybil Attacks in Networks

Wei Chang and Jie Wu

Department of Computer and Information Sciences

Temple University, Philadelphia, PA 19122

Email: {wei.chang, jiewu}@temple.edu

Abstract

Most peer-to-peer systems are vulnerable to Sybil attacks. The Sybil attack is an attack wherein a reputation system is subverted by a considerable number of forging identities in peer-to-peer networks. By illegitimately infusing false or biased information via the pseudonymous identities, an adversary can mislead a system into making decisions benefiting herself. For example, in a distributed voting system, an adversary can easily change the overall popularity of an option by providing plenty of false praise, or bad-mouthing the option through these fake identities. In this paper, we summarize the existing Sybil defense techniques, and further provide some new research areas. Unlike traditional surveys about Sybil defense, we first categorize the Sybil defense methods, mainly according to their designed time, and then classify the methods by their approaches. We believe that by understanding the evolution of the solutions, readers could essentially have more insights on the problem. In a nutshell, the research on the Sybil defense technique has experienced four phases: (1) traditional security key-based approaches, (2) specific peer-to-peer system feature-based solutions, (3) social network-based methods, and (4) social community-based techniques. Besides all of these anti-sybil methods, readers will also find some Sybil attack-related topics, such as Sockpuppets in online discussion forums. By the end of the paper, we will provide some predictions about directions for future research.

Index Terms

Network structures, peer-to-peer systems, reputation systems, security, social networks, Sybil attacks.

I. Introduction

We are entering a distributed computing era where a problem is cooperatively computed by many participating entities (also called nodes). Such cooperation mechanisms require that each participant trust one another. Just like in a team project, each person needs to trust his co-workers. In a typical (distributed) peer-to-peer system, the participators usually play three roles, simultaneously. First, they are the data owners, and they also share certain data with others. This data could be local raw data, such as sensor readings, or could also be partial computing results. Second, they are data processors: each participant locally processes the data according to some rules or algorithms. Third, they are also data transmitters. In a large-scale peer-to-peer system, a direct connection between each pair of nodes is impossible; therefore, the participating nodes usually build up a networks, and a message is transmitted from one node to another via the relay operations of multiple intermediary nodes (transmitters). If attackers control one or more participating nodes, they could modify the local raw data, local computed results, or all of the transmitted data. Clearly, by such an attacking mechanism, the attackers can modify the overall computation results of a peer-to-peer system, or even subvert the whole system. Therefore, security is a very important aspect of the research of peer-to-peer systems.

In this paper, we investigate the Sybil attack, a particularly harmful attack in distributed peer-to-peer systems. Almost all distributed peer-to-peer systems are based on a common assumption that each participating entity controls exactly one identity. However, whenever the assumption cannot be satisfied, the system is subject to sybil attacks. In a Sybil attack, an adversary creates a large number of forging/fake/pseudonymous identities (also named Sybil identities), and since all Sybil identities are controlled by the adversary, she can maliciously introduce a considerable number of false opinions into the system, and subvert it, by making decisions benefiting herself. Essentially, Sybil attacks break and manipulate the trust mechanism behind peer-to-peer systems.

For a better understanding of what a Sybil attack is, here, we provide three examples. First, in some distributed systems, critical resources are assigned based on the voting results of participants: usually, only the node that has received the highest number of votes can access the resources. If an attacker illegitimately creates a large number of Sybil identities, then the adversary can proportion more resources by instructing the fake identities to vote in certain ways, such as always voting for her fake identities. Since votes are collected indirectly (recall that, instead of through direct communication between remote nodes, most data are transmitted by the replay of other nodes), it is hard to detect the illegitimate votes. Another example comes from an application of sensor networks called 'pervasive temperature monitoring.' In a large region, multiple sensors are randomly and uniformly deployed. Each sensor measures its surrounding temperature, and further forwards the readings to a sink node, which collects the data. From the sink node, an average temperature can be computed. However, if the attackers launch Sybil attacks and let each Sybil identity report one more temperature degree, then the average temperature result will be incorrect. Our third example comes from a Facebook voting application. If an adversary maliciously creates many identities, she can easily change the overall popularity of an option by providing plenty of false praise, or bad-mouthing of the option through Sybil ids. Since the false opinions of the Sybils may essentially change the final decision of any distributed system, the research works on Sybil defense techniques hold the most important position.

The researches on Sybil attacks have passed three phases, and now, they are just entering the fourth phase. Note that our classification is based on the mainstream of research trends. Sybil attack is named after the subject of the book *Sybil*, a case study of a woman diagnosed with dissociative identity disorder. The name was suggested in or before 2002, by Brian Zill from Microsoft Research. The term ”pseudo spoofing” had previously been coined by L. Detweiler on the Cypherpunks mailing list, and was used in the literature on peer-to-peer systems for the same class of attacks prior to 2002. However, this term did not gain as much influence as “Sybil attack.” Phase I of the research on Sybil defense techniques began in 2002, and mainly ended in 2004. Within this phase, researchers tried to find some general mechanisms to defend themselves from all types of Sybil attacks in various networks or systems. In a nutshell, people tried to prevent Sybil attacks through redesigning system architectures and by involving secure mechanisms, such as digital signatures and identity authentication. However, the majority of approaches found at this phase faced a common problem: their schemes required a central authority for the verification of identities. Clearly, the trusted third party is the bottleneck of systems, which could easily become a target point. Moreover, it is impractical, since there is definitely not a globe agency who can be trusted by the entire public.

Around 2004 - 2006, the research trend entered phase II, which included specific peer-to-peer system feature-based solutions. Within this period of time, researchers focused on designing a defense system for a specific peer-to peer application system. Different application systems hold several unique features. By exploring these features, Sybil attacks can be detected, or the damage they cause can be bounded. For example, in sensor networks, nodes are static. By monitoring the received signal strength of each received message, Sybil nodes can be detected. However, readers need to be aware that such anti-Sybil systems are specially designed; an efficient solution for one application is typically not suitable for the others. Moreover, during the period from 2005 to 2006, a majority of researchers shifted to other secure problems, instead of studying Sybil attacks; the research on Sybil attacks was cooling down.

In 2006 ACM SIGCOMM, a novel paper [1] was presented, which led the research on Sybil defense to enter phase III. With goals dissimilar to those of phase I, the authors [1] aimed to adopt the concept of social networks. They wanted to detect Sybils based on a unique structure of friendships. Through observations, they found that, although attackers can create plenty of Sybils and further create plenty of friendships (also known as social links) among the Sybils, the number of links between Sybils and honest users is limited. This is so, because the links/friendships on a social network are built based on a trust relationship, as well as physical interactions among real people. Based on this key observation, many creative and interesting Sybil defense approaches were proposed at phase II, and the Sybil attacks got the attention of the public once again: in any network, or security related conferences or journals, you can easily find several papers mentioning Sybil attacks. These types of Sybil defense systems are also called social network-based Sybil defense. Note that these solutions do not detect Sybil attacks in social networks. Instead, they explore the social networks behind a peer-to-peer system. More details can be found in journal [2].

In the year of 2010, also in ACM SIGCOMM, another paper [3] provided a new trend for Sybil defense, which argues that the Sybil nodes can be detected by the community structures of social networks, since there is a short cut on the social graph of an attacked system. Community detection is a relatively mature topic in computer science, and plenty of useful techniques have been proposed. The paper [3] suggested that several community detection algorithms may be directly applied to the anti-Sybil problem. Note that partial methods in phase II essentially detect the honest community part of a social network in an implicit way, while the others just use other social features. We can also regard the phase IV, social community-based Sybil detections, as an extension of phase III.

The remainder of the paper is organized as follows. In Section II, we formally introduce the Sybil attacks on peer-to-peer systems. The examples about typical vulnerable systems are given in Section III. From Section IV to Section VII, we provide the general description of the anti-Sybil approaches for each phase. In Section VIII, we provide the prediction of future research, and Section IX concludes the paper.

II. Taxonomy of Sybil Attacks

To better understand the Sybil attacks, in this section, we provide a taxonomy of different types of Sybil attacks. The capability of the attacker is determined by several characteristics: (1) insider vs. outsider; (2) selfish vs. malicious; (3) directed vs. indirected communications; (4) simultaneously vs. gradually obtained Sybil identities; (5) busy vs. idle; (6) discarded or retained.

A. Insider vs. Outsider

Whether an attack is an insider or outsider directly determines the capability of the attacker, and the hardness of launching a Sybil attack. For an insider, the attacker holds at least one legitimate identity and claims that she receives certain data from the other nodes, by using the fake identities. Usually, a distributed system assumes that each node is trustworthy, and therefore, the false data can be forwarded to the whole system. However, for an outsider, she is any illegitimate entity; before launching a Sybil attack, she must first access the system. However, distributed systems typically employ some kind of authentication to prevent illegitimated access, such as a password for entering, or data encryption. The outsider needs to understand all the mechanisms of the system prior to launching Sybil attacks. Therefore, distributed systems are more vulnerable to inside attackers.

B. Selfish vs. Malicious

For security-related problems, there are two different types of attackers: either selfish or malicious. Selfish attackers manipulate the false data just for their own benefit, while malicious attackers attempt to subvert a system. Whether an attacker is selfish or malicious is usually determined by the different types of targeted distributed system and final attacking effects. For example, in our critical resource accessing example, if the attacker has resource accessing rights all to herself, then she is a malicious attacker, since others cannot use the resource. However, if other users can also access the resource with less probability, then she is selfish. Since malicious attacks usually have more serious effects, it is of higher importance to defend against potentially malicious attacks than those that are potentially selfish.

C. Directed vs Indirected Communications

How Sybil nodes communicate with honest nodes is also a significant consideration during the designing of Sybil defense mechanisms. The attacker can directly communicate with an honest node by using one of her Sybil identities, or she can use only her real identity to communicate with others, and route the Sybil data via this real identity. For the attackers, the easiness of direct communication with honest nodes directly influences the success of attacking, and whether honest users can see through the attack. In general, the attackers with more directed communications are harder to detect. However, for certain distributed systems, direct communication may be difficult to establish.

D. Simultaneously vs. Gradually Obtained Sybil Identities

The attacker can obtain all of her Sybil identities simultaneously, or she can gradually generate them one-by-one. For an intelligent attacker, the more diverse features the Sybil nodes have, the harder it is to identify Sybil nodes. Gradually creating Sybil nodes may potentially differentiate the first appearing time of the Sybils. However, the process may delay the attacking time, and increases the explosion time of some Sybils: if a distribution randomly checks the authentication of some identities, previously generated identities have a higher chance of being caught.

E. Busy vs. Idle

All Sybil identities can participate in a distributed system simultaneously, or only some of them can work, while others are in an idle state. Essentially, the selection of these two schemes is determined by how cheap it is to obtain an identity. If the attacker can easily get plenty of fake identities, having some idle Sybil nodes could make them much more real, since an honest node may leave or rejoin the system multiple times. However, the power of Sybil attacks results from the quantity of the identities. If obtaining a large number of identities is very difficult, the attacker has to use all of them in order to launch a successful attack.

F. Discarded vs. Retained

For an attacker, how to manage the old Sybil identities is important. After finding a Sybil node, one can further (and gradually) identify the others by monitoring the claimed communication between a suspect node and the detected Sybil node. Since the attacker is not aware of whether the old identities have been detected yet, once in a while, she has to determine whether or not to discard them. Consider that generating Sybil identities has certain costs, and the possible naming space is not infinite. The capacity of attacks are related with the naming costs and the mechanism of using old identities.

III. Examples of Vulnerable Systems

The Sybil attack is an attack wherein a distributed system is subverted by forging identities. Usually, peer-to-peer networks are vulnerable to Sybil attacks. In this section, we will provide several realistic examples of these vulnerable systems. Moreover, we will also provide another two examples, which are very similar to the Sybil attacks.

A. Vehicular Ad hoc Networks (VANETs)

A Vehicular Ad-Hoc Network is a technology that uses moving cars as nodes to create a special mobile network, which takes safety as its main purpose. In VANETs, each participating car can communicate with roadside base stations or other cars. However, this type of network is vulnerable to Sybil attack. For example, a selfish driver may launch a Sybil attack by claiming that many vehicles are traveling nearby. If this is the case, other cars may falsely believe that there is a traffic jam on the corresponding road, and therefore pick up an alternative road. The selfish driver will enjoy better traffic, with others paying the cost. Moreover, the Sybil attacks can also cause serious safety threats: a malicious driver may drop the warning messages. In VANETs, when a crash happens or speed significantly reduces, a warning message for slowing-speed will be generated, and is further forwarded to the following vehicles, one-by-one. By claiming many fake identities, the warning messages may all be transmitted to the malicious driver's car. If she drops these messages, other following cars will be in danger.

B. Distributed Voting Applications in Peer-to-peer Systems

Any distributed voting aggregation system is vulnerable to Sybil attacks. Usually, a distributed voting system consists of a collection of identities which vote for different objects. Most voting systems assume that each user only holds one identity, and each identity can provide only one vote. Based on this restriction, if attacks have many identities, then she can offer many votes. The vote can be in any form, from the simplest case, where each vote represents a positive or a negative opinion, to more complex cases, where the value of a vote can range within a given set of values. To rank objects, a ranking mechanism typically collects (or aggregates) the votes from distributed participants and further combines the votes in a certain method, such as the majority rule. By Sybil attack, the real users' major decision can be out-voted by the attacker: since the attacker can easily create many fake identities, the false opinions can be introduced into the system by these identities. Here, we need to claim that, although the Sybil nodes may be held by different attackers in reality, for the ease of description, researchers always assume that the Sybil identities are owned by a single adversary. This is due to the idea that this assumption will not influence the effects of the attacks, and will also not affect the results of defense approaches.

The example of Amazon's user feedback system in the introduction is essentially an aggregating voting system, since the reputation of each merchant is determined by the votes from customers. However, we also have to mention that the Amazon voting system is a centralized system, where all of the voting processes are controlled by a central server. However, generally, an aggregating voting system can also be a distributed system: each node can launch a vote, and the range of votes' values may different.

C. Distributed Storage Applications in Peer-to-peer Systems

Peer-to-peer storage systems adopt replication and fragmentation mechanisms, and usually the mapping from data to the corresponding stored nodes is performed by distributed hash tables. From the consideration of system stability and easy accessing, the mapping function is in the form of one-to-many. However, if the attacker is an insider, she can manipulate the values of her Sybil identities such that all the replicated data may actually be stored on the same malicious node, although the data seems to be stored at different nodes outwardly. Without multiple copies of data, the attacker can easily launch many followed attacks without being detected. For example, she can modify some data. Since she holds all of the data copies, no one can detect the modification of the data.

D. Routing in a Distributed Peer-to-peer System

To improve the performance or fault tolerance, wireless networks usually adopt a concurrent multi-path routing technique. Instead of using a single routing path, multi-path routing has multiple alternative paths throughout a network. The computed multi-paths may or may not be overlapped. This technique provides better load balancing and fault tolerance than traditional routing methods. However, in wireless sensor or ad hoc networks, Sybil attacks can easily invalidate the technique: a computed multi-path routing, which seemingly consists of multiple disjoint paths, could in fact only go through the same malicious node, which holds several Sybil identities. Other wireless routing mechanisms, such as the decentralized object location and routing (DOLR) algorithm, and the geographic routing algorithm, are also vulnerable to Sybil attacks. In peer-to-peer networks, nodes communicate with each other by relaying messages from one node to another, and the quality of the selected relaying paths directly influences the performance of a network system. In some extreme cases, Sybil attack may even isolate one part of a network from the other part.

E. Sockpuppets in Online Discussion Forums

In online discussion forums, in order to cheat people on the Internet, for instance, to believe that a product is a good buy, or that a particular investment plan has an extremely high return and low risk, a common trick is to use different fake online identities pretending to be different people. This is done to praise or create the illusion of support for the product [4]. In the same forum, different online entities which belong to the same person are referred as ‘sockpuppets.’ Note that sockpuppet does not belong to Sybil attack, since online discussion forums are not peer-to-peer systems. However, because sockpuppets have several features similar to Sybil attacks, we want to mention them. First, both attacks are based on the usage of multiple identities belonging to the same person. Second, their success is related to the same assumption that each user is associated with one, and only one, identity. Third, they all break the reputation mechanism behind a given system. Last, for some distributed peer-to-peer systems, such as mobile social networks, there are social features and friendships associated with each identity; this also applies to an online discussion forum. Due to these similarities, the solution to one attack may help the design of the other.

F. PageRank in Searching Engines

Another attack, which is similar to, but different from Sybil attacks, is called spoofed PageRank [5]. For modern search engines, the ranking of a page is determined by the quality and quantity of referenced links. In order to promote a page’s ranking, the page owner (a selfish attacker) may create a lot of small and meaningless spoofing pages, and let them link to the page. This type of attack is called spoofed PageRank. If we consider each page as an identity, and its corresponding PageRank as reputation or trust, then spoofed PageRank is the same as Sybil attack; both of them promote the opinion or reputation of a malicious entity by using plenty of fake entities. Again, these similar features may help to design some new Sybil defenses. For example, Advogato Trust Metric [6] is a Sybil defense system, where users certify each other in a kind of peer-review process. As the author observed that his notion of a trust metric was fundamentally very similar to the PageRank algorithm, the solution of identity-spoofing related schemes in a field may inspire the design of defense in another field. Since trust, PageRank, and Sybil attacks have delicate relationships, we present several details about them.

In a centralized ranking system (like Google PageRank), the rank of an entity is computed from a stationary probability distribution of a Markov chain, in which a random walker moves from one node to another by following the edges with a constant probability d (also named damping factor), or randomly jumping to another.

Let $\mathbf{P'} = [p'_{ij}]$ be a $m \times m$ transition matrix, which is a row-normalized link matrix whose value shows the probability of transmitting from v_i to v_j during random walks (by following the link structure of a network). If there is an edge $v_i \to v_j$, then $p'_{ij} = 1/D^+(v_i)$; otherwise, $p'_{ij} = 0$. $\overrightarrow{Q}$ is a $1 \times m$ vector, $\overrightarrow{Q} = [q(v_1), q(v_2), \cdots, q(v_m)]$, and it stands for a preference vector during a random jump, $q(v_i) \geq 0$ and $\sum_{i=1}^{m} q(v_i) = 1$. The computation of PageRank is an iteration process; we use $\overrightarrow{R}(t)$ to represent the PageRank at the t^{th} computing round. The steady value of $\overrightarrow{R}(t)$ is the final PageRank $\overrightarrow{R}$, which is a $1 \times m$ vector, $\overrightarrow{R} = [r(v_1), r(v_2), \cdots, r(v_m)]$. $r(v_i)$ means the PageRank of the node v_i. $\overrightarrow{R}(t+1)$ is defined as follows:

$$\overrightarrow{R}(t+1) = d\overrightarrow{R}(t)\mathbf{P'} + (1-d)\overrightarrow{Q}$$

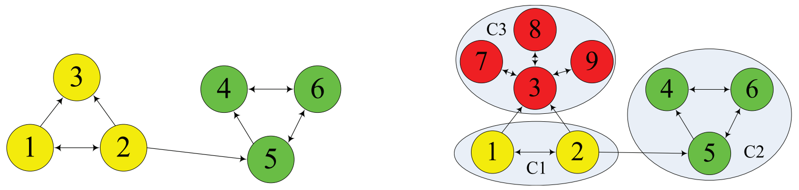
Realistically, when $|\overrightarrow{R}(t+1) - \overrightarrow{R}(t)| < \epsilon$, we let the value of $\overrightarrow{R}(t+1)$ be $\overrightarrow{R}$. For the ease of description, we let $\mathbf{P}$ be a probability transition matrix used for computing PageRank, $\mathbf{P} = [p_{ij}]_{m\times m}, p_{ij} = d \cdot p'_{ij} + (1-d)q(v_j)$. Algorithm 1 gives the procedure for computing PageRank. Take Fig. 1(a) as an example. For v_2, $D^+(v_2) = 3$, $p'_{21} = p'_{23} = p'_{25} = 1/3$, and $p'_{22} = p'_{24} = p'_{26} = 0$. When the damping factor $d = 0.85$ and $q(v_1) = q(v_2) \cdots = q(v_6) = 1/6$, p_{21} can be computed as follows: $p_{21} = d \cdot p'_{21} + (1-d) \cdot q(v_1) = 0.85 * (1/3) + (1-0.85) * (1/6) = 0.308$.

$$\mathbf{P} = \begin{bmatrix} 0.025 & 0.450 & 0.450 & 0.025 & 0.025 & 0.025 \\ 0.308 & 0.025 & 0.308 & 0.025 & 0.308 & 0.025 \\ 0.167 & 0.167 & 0.167 & 0.167 & 0.167 & 0.167 \\ 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.875 \\ 0.025 & 0.025 & 0.025 & 0.450 & 0.025 & 0.450 \\ 0.025 & 0.025 & 0.025 & 0.450 & 0.450 & 0.025 \end{bmatrix}$$

After obtaining **P**, we used Algorithm 1 to compute the PageRank of each node. Initially, we let each node's PageRank equal $1/m$. Algorithm 1 iteratively computes the PageRank via using $\overrightarrow{R}(t+1) = \overrightarrow{R}(t)\mathbf{P}$, and the algorithm stops when $\overrightarrow{R}(t)$ is stable. For Fig. 1(a), the final PageRank is $\overrightarrow{R} = [0.0517\ 0.0574\ 0.0737\ 0.2686\ 0.1999\ 0.3487]$. Clearly, the webpage v_3 ranks as number 4 among these nodes. However, in Fig. 1 (b), through adding three spoofing pages (alternatively, we called these fake identities as Sybils in a peer-to-peer system), the attacker, v_3, ranks as number 1, since those fake pages grab some ranking values from others and uniquely support v_3. Obviously, Google

Algorithm 1 PageRank

```
1: Assign initial values for entities: $\overrightarrow{R}(0) = [1/m, 1/m, \cdots, 1/m]$
2: $\overrightarrow{R}(1) = \overrightarrow{R}(0)\mathbf{P}$, $t = 0$
3: while $(|R(t+1) - R(t)|) > \varepsilon$ do
4:     $t = t + 1$, $\overrightarrow{R}(t+1) = \overrightarrow{R}(t)\mathbf{P}$
5: Return $\overrightarrow{R}(t+1)$ as $\overrightarrow{R}$
```



(a) A directed network. (b) The community structure of pages.

Fig. 1. The change of a network's structure before and after having a spoofing page attack. Fig.(a) shows the network structure without the spoofing pages. In Fig.(b), v_3 employs v_7 to v_9 as spoofed pages.

PageRank is vulnerable to spoofed PageRank attacks. Similarly, the other famous web page ranking algorithm, HITS, also has the same weak point.

IV. Sybil Defense in Phase I: Secure Key-based Anti-Sybil Techniques

According to information theory, in order to detect Sybil nodes, one must possess asymmetric knowledge, which means the detecting algorithms must hold more information about either the Sybil part or the honest part. For the techniques in Phase I, they usually only provide valid keys to the honest nodes. Here, the "key" is a general concept; in reality, these could be symmetric or antisymmetric keys. Also, the keys could be session keys or some permanent keys.

A. Trusted Certification (Centralized Solutions)

Sybil attacks can be avoided by using trusted certification. This type of method assumes that there is a special trusted third party or central authority, who can verify the validity of each participant, and further issues a certification for the honest one. In reality, such certification can be a special hardware device [7] or a digital number [8], [9]. Note that essentially both of them are a series of digits, but are stored on different medias. Before a participant joins a peer-to-peer system, provides votes, or obtains services from the system, his identity must first be verified. Actually, this method is the most commonly used Sybil defense in our daily lives. For example, when we are applying for a credit card, we need to provide our social security number for verification; when we are voting in election years, we also need our official ID card for getting a ballot.

When a malicious user launches Sybil attacks, defense mechanisms usually require that a message be sent together with a signature, which could be used for authenticating the validity of the sender or the data. Actually, according to a paper [10], trusted certification is the only approach that has the potential to *completely* eliminate Sybil attacks. Since almost all authentication steps require the participation of the central server, we categorize this type of solution as a centralized trusted certification.

B. Trusted Certification (Semi-Centralized Solutions)

Centralized trusted certification approaches are often implemented by asymmetric (such as public/private keys) cryptography. However, the computational cost is a big problem. Some researchers proposed another type of solution, where partial identity verifications can be done without using the public/private keys. We named these kinds of solutions ‘semi-centralized solutions.’ Paper [11] provided a solution by using symmetric keys. They assumed that each node shares a unique symmetric key with a trusted base station. After verifying the validity of each other via a Needham-Schroeder like protocol, a pair of nodes can establish a shared key. During data transmission between neighboring nodes, they can use the shared key for mutual authentication, and can also encrypt the data.

C. Common Problems with Techniques in Phase I

Trusted certification usually relies on a centralized trusted authority for assigning and verifying identities. The authority must ensure that each node is assigned exactly one identity, and that the identity list (also called a registration list) is well protected. In real-life, the process of assigning identities is usually performed by human beings, which is costly and becomes the bottleneck of systems. Moreover, the central authority also needs to deal with lost identities and updates. The performance of real systems obstructs the usage of these solutions.

We summarize several obvious shortages of central authority-based methods, as follows:

1) Single point of attack. In these schemes, the central authority can easily become a target. Besides Sybil attacks, attackers can also launch plenty of other attacks, such as denial of service, to crash the server.
2) Performance bottleneck. If several users access a central authority simultaneously, the authority may crash due to the huge workloads.
3) Communication cost. In this type of method, the authority is often required during the data transmission. For example, two strange nodes need the help of the authority for verifying each other. There is a considerable amount of extra communication between nodes and the authority.
4) Scaling. It is hard to construct an authority, which can be trusted by all participants, especially when a peer-to-peer system wants to include more users from diverse places.

D. Other Defenses in Phase I

Identity Fee: Unlike the trusted certification-based approaches, some other papers [12]–[15] add an economical “fee” with each certification. They argue that the attackers cannot easily subvert a peer-to-peer system unless they spend a lot of money. Indeed, they intend to build a system letting the cost of an attack outweigh the benefits of the attack.

Secure Hardware: As we have mentioned in the beginning of this section, that the certification key is a general concept, a Sybil defense system can also be built based on the usage of some secure hardwares. Usually these types of hardware periodically generate some time-sensitive token. Whenever a participant wants to verify the validation of his encounters, they just determine whether the token is valid or not. Note that there is an assumption behind the scheme that only honest users could get the secure hardwares, and each valid user can only get one. Although the verifying process does not need the central authority, it is still responsible for dispensing the hardwares to valid users. Moreover, before giving out hardware to a user, the authority must verify the user's identity.

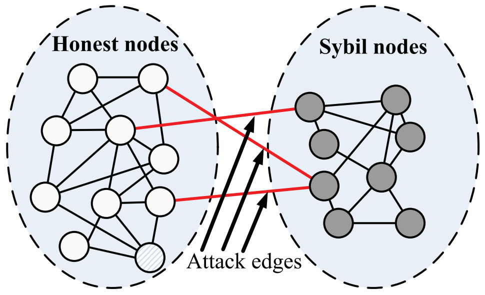
Resource Testing: Usually, each user owns only one identity, and each identity works on a single machine. However, when Sybil attacks are launched, the Sybil identities work on a single computer. When we give some time or resource consuming tasks to a group of identities, if they can finish the work within a threshold, then it is highly possible that they are honest users; otherwise, part of them may be Sybil. In general, the goal of resource testing [16]–[19] is to determine whether the selected identities have a reasonable amount of resources. The tests, which are adopted for these kinds of approaches, include: (1) checking computing ability; (2) checking storage ability, and (3) checking network bandwidth. Readers must be aware that resource testing is not an efficient approach [10], but we still adopt them for deterring or discouraging the attackers.

V. Sybil Defense in Phase II: Specific System Features-based Anti-Sybil Techniques

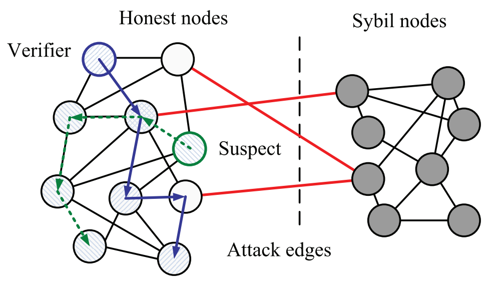
Since attackers have a limited number of real devices in wireless ad hoc networks or sensor networks, a group of Sybils actually shares one device, and therefore, Sybils can be detected by letting honest users monitor signals' features or the moving patterns of co-existing identities. Consider that there are channel conflicts during the communication of honest users, while Sybil identities never have real data transmission. Paper [20] proposed a Sybil detection method by monitoring the neighbors' channel conflict rate. They assume that there is a central server that records the rate of each identity. Whenever a channel conflict happens, certain nodes should report the event to the server. If some identities have an abnormally low rate, then the server will regard them as Sybil identities. However, the readers should understand that this method will be inefficacious if the attackers are aware of the defense mechanism, and will further purposely send certain signals for conflicts.

Considering the fact that Sybils usually appear together, paper [16] adopts moving patterns for Sybil detections. In order to increase the accuracy, they also introduce a signal collision-based improvement, based on the observation that the collision rates inside the Sybil groups are lower than that of the normal groups. However, when the density of honest users is not large enough, the accuracy of this type of Sybil defense will not be guaranteed.

Sensor networks are static. By exploring this feature, paper [19] proposed an RSSI-based Scheme for Sybil detection. They assumed that each node has the ability to measure signal strength. When receiving a message, the receiver node will associate the received signal strength indicator (RSSI) with the identity of the sender, which is included in the message. For attackers, a Sybil node is able to send messages with different sender identities. Consider that all sensors cannot move, and multiple Sybil identities are essentially sent from the same attacking sensor. Later, if the messages of another sender possesses the same RSSI, then the receiver can treat these senders



(a) Special link structure of Sybil nodes in social networks.



(b) General idea of SybilGuard.

Fig. 2. A social network-based Sybil defense scheme: SybilGuard. Fig.(a) shows the unique social network's structure for the distributed systems under Sybil attacks. Based on the fast-mixing feature of social networks and the unique structure (a short-cut), SybilGuard tries to identify Sybil nodes through random walks, as shown by Fig.(b).

as Sybil nodes. This approach essentially verifies whether several identities share the same physical locations, and other papers [7], [21], [22], also call this type of approach 'position verification.'

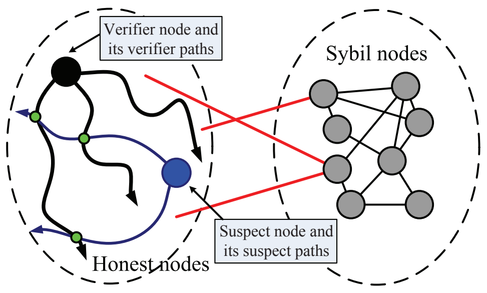
Vehicular Ad Hoc Networks (VANETs) are also vulnerable to Sybil attacks. Paper [23] proposed a Sybil detection protocol by using vehicles' historical geographic information. The core technique for this method is position verification of mobile nodes: the method measures a possible existing area of a car, based on the car's and its neighbors' historical positions.

VI. Sybil Defense in Phase III: Social Network-based Anti-Sybil Techniques

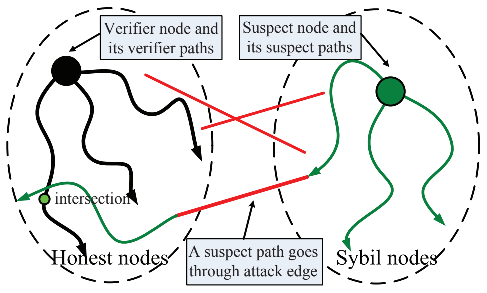
In 2006 ACM SIGCOMM, paper [1] presented a novel idea on Sybil defense, which explores social networks *behind a given peer-to-peer system*. The authors want to detect Sybils based on a unique structure: although attackers can create plenty of Sybil identities, and further establish several links among them; the total number of links between the Sybil and the honest users is limited, since the trust relationship on a social network is built based on the trust relationship among real people. In other words, the corresponding social graph of an attacked peer-to-peer system contains a small cut structure. Admittedly, in real online social networks, such as Facebook or Twitter, a user may accept the friend request of a stranger. However, by using interaction networks, which provides a closeness rate based on historical interactions, or by using some special Apps, which allow users to manually enter some trust degree, we still can guarantee the limited number of trusted relationships between the honest and the Sybil users. After the publishing of this paper, many researchers came back to the works of Sybil defense, and even until now, there are still a lot of researchers working on this idea. In this section, we will introduce several typical anti-Sybil approaches based on the usage of social networks.

A. *SybilGuard and SybilLimit*

SybilGuard [1], and SybilLimit [24] are two famous Sybil defenses that use social networks. Since their core techniques are similar, we will only introduce SybilGuard. SybilGuard defines two terms: 1. a trusted path; 2. a trusted node. Similarly, for breaking the symmetric information constriction, SybilGuard also assumes that there


Verifier node and its verifier paths
Sybil nodes
Suspect node and its suspect paths
Honest nodes

(a) A honest path verification.


Verifier node and its verifier paths
Suspect node and its suspect paths
intersection
Honest nodes
A suspect path goes through attack edge
Sybil nodes

(b) A Sybil path's verification.

Fig. 3. The difference between the verification of honest path and sybil path in SybilGuard. Fig.(a) shows the condition that the verifier is checking an honest node. Fig.(b) gives the case that a verifier is checking a Sybil node.

is a known trusted node. From this trusted node, there are K random paths with a fixed length l. For the ease of description, we call these paths verifiers. From a suspect node, SybilGuard also sends K random paths. If a path encounters a verifier once, then we call the path 'been verified once,' as shown in Fig. 2(b). If a path has been verified S times, then the path is a trusted path. When the majority paths of a suspect node are trusted paths, the suspect node will be regarded as a trusted node; otherwise the node is a Sybil. Essentially, these random walks measure how well a suspected node and a verifier node are connected. The reason for SybilGuard working well is that the number of attack edges is limited in social network, as shown in Fig. 2(a). A majority of verifiers and a majority of the random paths from suspect nodes will remain in their resident communities.

Now, consider the case that a verifier comes into a Sybil region. Although this verifier can encounter plenty of Sybil initialized paths, most of the encountered Sybil paths cannot get enough verifications, since only a very limited number of verifiers falsely enter the Sybil region. On the other hand, if a Sybil path enters the honest region through the attack edges, the path may intersect with verifiers many times, and therefore, become verified. However, because the number of attack edges is a limited number, the majority of Sybil initialized random paths cannot be verified, as shown in Fig. 3(b). So, from the consideration of bounding the effects of Sybil attacks, SybilGuard works well.

B. *SumUp*

SumUp [25] is an anti-Sybil technique designed for a distributed voting system. Before we discuss the general idea of SumUp, we first need to understand the meaning of a credit network. Credit network [26] is a concept used in the electronic commerce field, and it is designed for building and measuring transitive trust among users. Note that, in the field of electronic commercial, trust is usually pairwise. Whenever a node (identity) trusts another node, a trust link will be established, together with certain trust value (credits). When a node gets services from others, the node can use the associated credits to pay for the services. Note that the credit network could also be used as a payment infrastructure between nodes that do not directly extend credit to each other [27]. Two remote nodes, which do not directly trust one another, can interact with each other when there exists credit paths between them. In some systems, such interactions will cost credits from the paths.

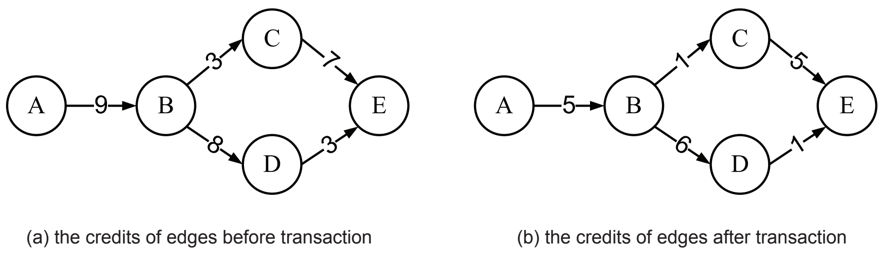

A
B
C
D
E
9
3
8
7
3
(a) the credits of edges before transaction
A
B
C
D
E
5
1
6
5
1
(b) the credits of edges after transaction

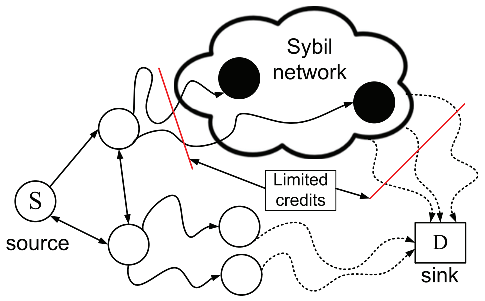
Fig. 4. A credit network. Credit networks are directed and weighted graphs, where the weights of edges indicate the credits between two neighboring nodes. Interactions between two nodes are also called transactions; each transaction costs some credits on a trust path.

Formally, a credit network can be represented by a directed graph; each directed edge is associated with a credit value. In general, the credit is a dynamic value: each transection (one time of interaction) consumes a fixed amount of credit. Note that if the credit of an edge becomes zero, then the corresponding two nodes are not able to trade any further. The interactions between two remote nodes are also allowed (suppose node v wants to get service from node w), if and only if there is at least one path from v to w, and each hop on the path can "pay" a required credit. Moreover, such payment can also be split across multiple paths if they exist. Take Fig. 4 as an example [27]. When node A gets service from a remote node E, we assume that the system requires 4 path credits in total. Node A adopts an equal splitting mechanism, as follows: it first pays 2 credits along the path $A \longrightarrow B \longrightarrow C \longrightarrow E$ (note that each hop costs 2 credits on the path), and pays another 2 credits along the path $A \longrightarrow B \longrightarrow D \longrightarrow E$.

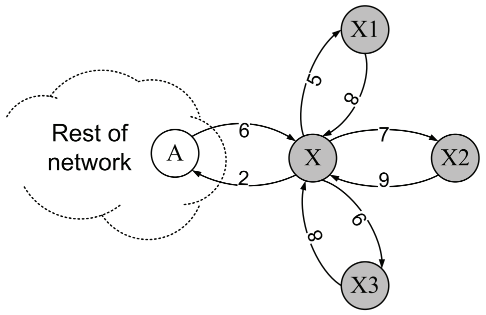
The general idea of SumUp is that, in credit networks, the links between honest users and Sybil users are limited (as shown in Fig. 5(a)), and the social networks in the honest users part are fast-mixing; most honest users can participate in a voting, since there are plenty of trusted paths from the honest user to a sinker node. However, since the number of credit links is limited, most Sybil nodes cannot provide their false opinions to the system. Fig. 5(b) is an example. Although the attacker, node X, has three Sybil nodes, X_1, X_2, and X_3, the credit on the directed link from Sybil to the honest is only 2. Hence, if we assume that each action takes 1 credit, then no matter how many Sybils the attacker could create, he can only give 2 actions at most, which definitely will not change the voting result.

C. Canal

Canal [27] also adopts a credit network, and we can regard it as an extension of SumUp. In a credit network, each interaction between nodes always requires the system to first find at least one available credit path; clearly, such a process has a high computational cost. Essentially, the procedure of searching such paths is equivalent to the maximum flow problem. However, even the most efficient algorithms have the computation complexity in $O(V^2 \log(E))$. Instead of finding one or several "best" credit paths, Canal approximates the paths: they first partition the graph into several layers and regions, and adopts a landmark routing-based technique to find the paths. Landmark routing [28] is an old technique: in order to find a path between a pair of nodes, they first determine the paths from the nodes to several pre-selected nodes (also known as landmark nodes). Since the paths between



(a) General idea of SumUp.



(b) An example.

Fig. 5. A Sybil defense named SumUp, which can be used in online distributed voting systems. Sumup essentially restricts the damage of Sybil attacks by using credit networks. Note that, although an attacker may create plenty of Sybils, the credits from the Sybils to the honest nodes are limited.

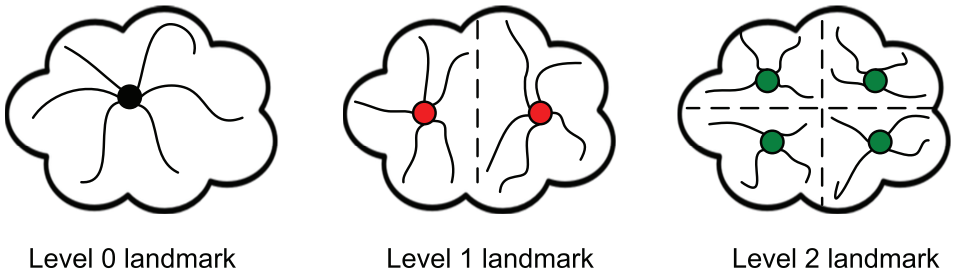


Fig. 6. A extension of SumUp, named Canal system. The Canal system adopts landmarks to partition the whole network into layers. The trust routing between nodes are conducted by the routing between local landmarks.

landmarks are known, the resultant path will be a special path, which goes through at least one landmark. In Canal, as shown in Fig. 6, the author uses multiple-level landmarks: if a credit needs to be transmitted to nearby nodes, the path will go through the lower level landmark; when a credit is transmitting to far way nodes, the shortest path will pass a higher level landmark. However, since the landmark absorbs the paths which may cause the credits of nearby paths to decrease quickly, the landmarks should be randomly generated and periodically updated.

VII. Sybil Defense in Phase IV: Social Community-based Anti-Sybil Techniques

Suppose that, in a peer-to-peer system, there are m honest nodes and n Sybil nodes. After the central authority obtains all of the data, based on the features of these data, he may predicate whether a node is honest or not. However, assume that the attackers somehow replicated all of the m honest users' data and built a network with the same structure as the honest one, and then, sent their replicated data to the collector. If this is the case, no matter how, the collector cannot discriminate a Sybil node from others, since all of information is exactly the same. For breaking such symmetric information, a Sybil defense system must build on some asymmetric information. Recall that social network-based Sybil defense algorithms always assume that the executants of the algorithms regard themselves as the verifier (each user at least knows that it is trustworthy, itself). The reason for having this assumption is just to break the symmetric information.

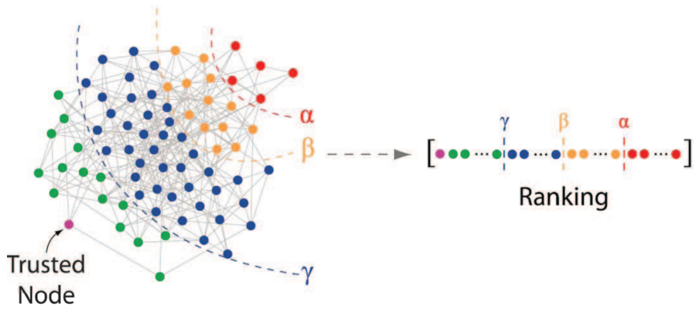


Fig. 7. The essential approach of the existing Sybil defenses in social networks [3]. Most random walk-related Sybil defense algorithms essentially assign weights to other nodes based on their distance. By providing a threshold, the nodes with lower weights are regarded as Sybil.

In 2010, paper [3] analyzed several classic social network-based anti-Sybil algorithms, and found that those algorithms essentially detected the community structure of honest users. Following this discovery, the research community has begun to work on community-based Sybil defense approaches. Note that the exact concepts used by different papers may be different; some papers focus on that of traditional social networks, while some others work on signed social networks; the traditional communities are in a global view, however, it could also be in a local view. Since the concept of community-based Sybil defense is relatively new, we are not sure whether it will become a mainstream, or if it is just an extension of social network-based solutions.

A. Pure Social Community-based Sybil Detection

In 2010 ACM SIGCOMM, another novel paper [3] provided a new trend for Sybil defense, which argues that the Sybil nodes can be detected by their community structures. Since the research on community detection has been there for many years, and there are plenty of useful techniques that have been proposed, the paper [3] may open up another option for anti-Sybil approaches. Since the paper is very fresh and, currently, we do not know how many followers there will be, we just say that the researches may, or may not, enter into the third phase.

[3] exams all the famous Sybil defenses in social networks and finds out that, indeed, these methods partition a given network into communities; the pre-known honest node's resident community is treated as an honest community, and all of the others' communities are regarded as Sybil nodes' resident regions. As shown in Fig. 7, the authors found that all the existing Sybil defenses are essentially rating systems, which assign a value to each node, based on the distance towards pre-known honest nodes. Different Sybil defenses may use different thresholds to partition the nodes. Hence, the authors proposed that the Sybil nodes may be detected through community detections.

The authors of [3] proposed a Sybil defense by using conductance-based community detection. They regard all of the nodes in a social network as one of two types: the nodes resident in the honest community and nodes resident in the Sybil community, as shown in Fig. 9. Through some simulations on synthetic data, they argue that their method can successfully detect Sybil nodes, and their resident community.

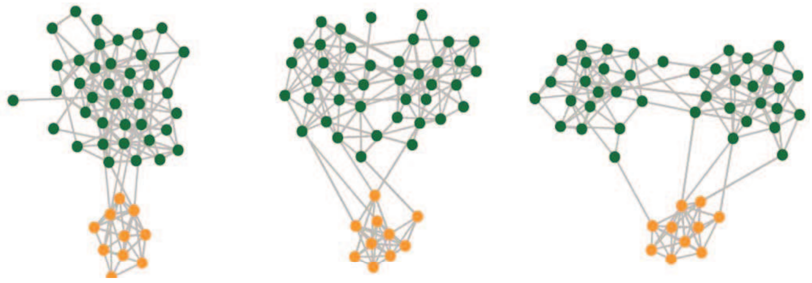


Fig. 8. Future research directions [3]. In the experiment part of paper [3], the author shows that the accuracy of social network-based Sybil defense is related with the structure of honest nodes. When the honest nodes present multiple communities, most social network-based Sybil defense algorithms have high false positive results. In this graph, the upper green nodes stand for honest nodes, and the lower orange nodes represent Sybils. From left to right, the figure shows the process that the structure of honest nodes changes from single community to two communities.

B. SybilDefender

SybilDefender is another famous approach for anti-Sybil. It consists of two steps: Sybil node detection and Sybil community detection. For a suspect node, based on pre-known honest nodes' statistical features, SybilDefender determines whether the suspect node is a Sybil or not. After finding a Sybil node, based on the assumption that Sybil nodes are more likely to connect with other Sybil nodes, the defense will detect the Sybil community in which the Sybil node resided. This defense is based on two assumptions: 1. the number of links between honest users and Sybils are limited; 2. the size of the Sybil community is smaller than that of the honest. The second assumption is realistic, since typical social networks contain millions of users; for the attacker to register such a large number of identities is impossible. From an honest user, we can send a fixed number of random walkers to pass an l-length random path, assuming there are k walkers. At other nodes, we can compute the times that these random walkers passed through this node, and call the times their 'visiting frequency.' After that, we can calculate the statistic distribution of the visiting frequency. If the random walks from a suspect node do not follow some statistic distribution, then the suspect is a Sybil.

Fig. 10 illustrates the SybilDefender. In Fig. 10, suppose that we have already known an honest node. From this node, we send out k random walks with a fixed length l. Since social network (in the honest region) is fast-mixing, which means that any pair of nodes can reach one another at an $O(\log n)$-length random path, a circle region in the honest community will be covered by the random walk. However, because the size of the Sybil community is smaller than that of the honest one, the majority of random walks in the Sybil region will be reflected, which indicates that the distribution of the visiting frequency in the Sybil region is different from the honest one. By this way, a suspect node can be verified.

However, since we do not know the size of Sybil communities, how to determine the length of random walk is a problem. As shown in Fig. 11(a), if the length is longer than the radius of the honest community, or as shown in Fig. 11(b), if the size of a Sybil community is greater than a random walk's length, the distribution of visiting

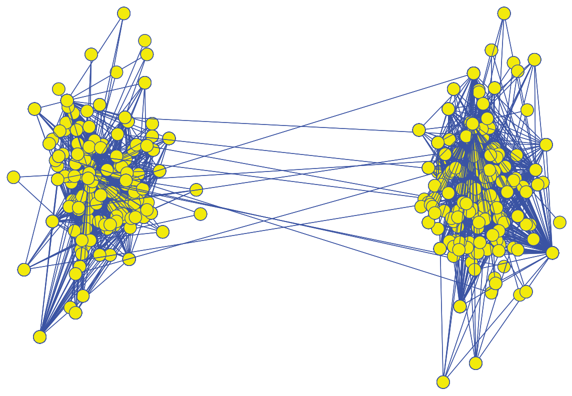


Fig. 9. An honest community and a Sybil community. For the synthetic data, people usually creates two communities; let one of them be the honest community, and the other the Sybil community. The edges within each community are randomly generated, and the node degrees follow the power law distribution. By using a rewiring operation, a limited number of attack edges are added between the communities.

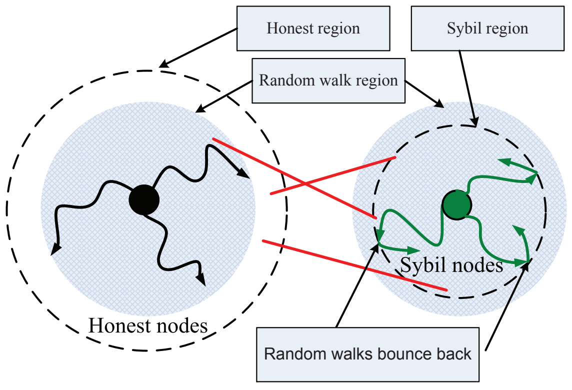
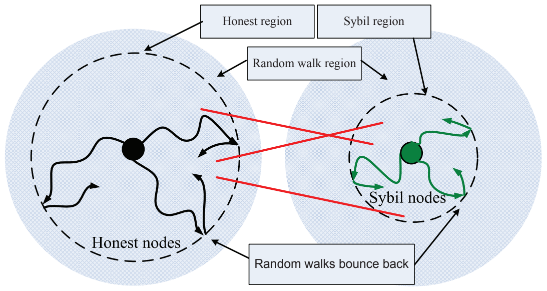


Fig. 10. The idea of SybilDefender. One fundamental assumption of SybilDefender is that the size of Sybil community is much smaller than that of the honest community. When the length of random walks is appropriately set up, one can detect the existence of Sybil nodes via monitoring nodes' visiting frequency.

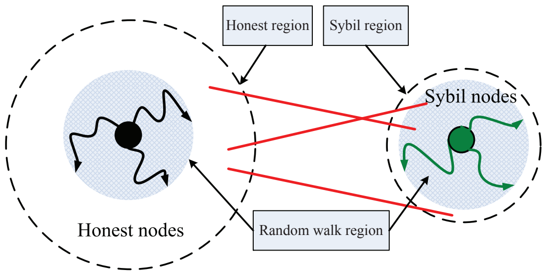
frequency may not show the difference. In other words, detecting by the nodes' visiting frequency may fail if the length of random walks is not set up appropriately. Hence, if the cases of Figs. 11(a) and (b) happens, we have to try other random walk lengths.

Another challenge with the SybilDefender is that of how to extract the correct visiting frequency distribution from the honest region. Clearly, if we select an honest node which has been fooled by an attacker, the computed statistic feature of the honest node will definitely be different from that of other honest nodes, which locate at the core of an honest community. From this consideration, before computing the statistic feature of an honest node, SybilDefender first finds several K-hops neighbors. Considering that the links between honest and Sybil nodes are very limited, a majority of these K-hops neighbors will also be honest. Moreover, since the social networks are fast-mixing, the statistic features of the nodes can correctly reflect that of the whole honest region.

Since SybilDefender does not know the size of the Sybil community, it initializes several groups of random



(a) The length of random walk is too long.



(b) The length of random walk is too short.

Fig. 11. The appropriative length of random walk is related to the size of a Sybil community. The size of Sybil community is important to know for setting the length of random walks. In Fig. (a), the length of random walk is too long since the walks bounce back at both honest region and Sybil region, while in Fig. (b), the length is too short because the random walks do not bounce back at both regions.

walks with different lengths. Then, during the verification of a suspect node, several random walks (with different lengths) are conducted. If the distribution of the visiting frequency of the suspect node does not share the same distribution as the honest one, this suspect node will be regarded as a Sybil node; otherwise, SybilDefender will regard the suspect as an honest node.

After finding a Sybil node, the SybilDefender can also detect its resident Sybil community, based on the fact that Sybil nodes are more likely to connect with other Sybil nodes. The detection of a Sybil community can be done by using loop-free random walks. Consider that when a random walker passes the same node twice, it means the random walkers reach the boundary of the Sybil community. SybilDefender renders a random walker dead if it arrives at the same node twice. Similar to the process of verifying a suspect node, SybilDefender also initializes several random walks with different lengths. Again, the reason is that the size of the Sybil community is unavailable. If the dead ratio of the L-length group of random walks is greater than a pre-defined threshold, then all the passed nodes will be regarded as members of a Sybil community.

C. *Signed Social Network-based Sybil Defense* [29]

Most social network-based Sybil defenses adopt the assumptions that the honest region is a fast-mixing network, and that Sybil entities can only fool a limited number of honest entities. 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. For addressing these two important problems, we propose a local ranking system for estimating the trust level between users in mobile social networks.

Unlike traditional Sybil defenses, our proposed scheme has three unique features. First, our system creates a signed social network, which contains both trust and distrust relations. Second, consider that each mobile phone only has a relatively small storage, and frequently accessing a remote server increases the amount of data flow, which costs an extra fee. In our solution, instead of storing the entire social graph, each user carries a limited amount of information related to him. Last, but not least, our system weakens the impacts of attack edges by

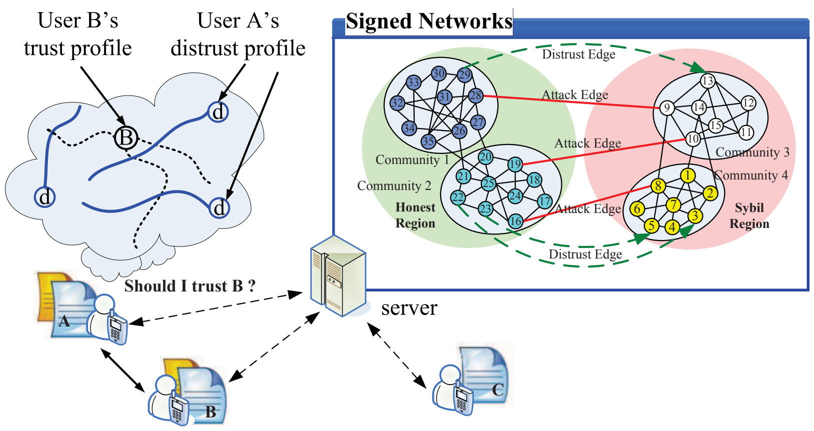


Fig. 12. System model. The system consists of two parts as follows: (1) a central server, which stores signed social networks; (2) users' smartphones, which detect the physical encounter of other users, reports other users' misbehavior to the server, and estimates the trust level of the encounter.

removing several suspicious edges with high centrality.

In mobile social networks, our system consists of two parts: a remote server, and several users, as shown in Fig. 12. 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. Note that now we are using two networks: a mobile social network and a created signed social network. The mobile social network is the network formed by physical interactions of phone users, while the signed network is created for Sybil detection. The positive edges on the created signed social network represent trusted social relationships, which could be obtained from an online-social network. The negative edges are generated based on users' physical interactions with each other. We assume that each honest user has one mobile phone, which is associated with a single real identity, while the attacker may hold more than one phone, and each phone runs multiple fake identities. Each identity is required to periodically send a special message to the server, and the server will return updated social profiles; otherwise, the identity will be deleted from the system. Unlike traditional social network-based Sybil defense models, we assume that the honest region of a social network may not be fast-mixing. Exactly how many honest communities may be formed is determined by the social networks being considered. For instance, if we use a social network of political opinions, then the honest nodes may be gathered into two communities, e.g., the Democratic and Republican parties. However, if we adopt Facebook, the honest users may only be clustered into one community.

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. Moreover, in mobile social networks, each identity used by a mobile phone is associated with a physical human, and some users may remember the appearance of others. When several honest users, who have been fooled by the same attacker, 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 for the generation of negative edges.

The general procedure of our system is as follows. Each user locally stores two *randomly sampled* social profiles: a trust profile and a distrust profile, which are assigned and periodically updated by the server. Whenever two strangers encounter one another, and want to have some cooperative service, each user's phone will exchange the trust profile, and locally compute a trust and a distrust score to determine whether the other user is trustworthy. In order to increase the accuracy, a special pruning algorithm is running on the server.

When a new user V joins our system and provides his friend/foe lists, the server will generate a trust-relation profile and a distrust-relation profile. The generating procedure for the trust-relation profile is as follows: the server first sends out K random walkers, and each of them will conduct an l-length random walk from V. The walkers only move along trust edges, and each path represents one possible way of trust propagation from V. As a result, there will be K random paths beginning with V, and the visited node list will be sent to V as a trust social profile. Obviously, the profile is a random sample of V's l-hop friendship. Consider that, in a mobile social network, a user may have different physical contacting frequencies to his directly trusted friends. User V is able to locally assign different weights to the paths, according to the frequencies.

In order to cheaply impersonate real users, and to benefit from certain applications, Sybils always support each other by adding trust edges among themselves[1]. Therefore, friends of a distrusted node are likely to be distrustful. Moreover, the majority of random walks from a node will still reside in their own community. Based on these observations, the server creates V's distrust social profile by using the distrust relations of both V and his trusted friends. Before creating the profile, a distrust seed set needs to be generated: along trust edges, the server computes K short-length random paths from V, and nodes directly distrusted by the nodes on these paths form the seed set. Another l-length random walk will be produced from each seed, and the trails will be used as the distrust social profile of V. For instance, in Fig. 13, the solid green line p represents one of the short-length random paths from V, and there are 3 distrust edges (dashed green lines) initiating from the nodes on p. The distrust seed set consists of 3 nodes (shadowed circles). From each of the seeds, the server conducts an l-length random walk, and the generated random paths compose V's distrust social profile; again, all random walkers are moving along trust edges.

When an honest user V encounters another user S, a trust-level will be computed based on the similarity of their locally-stored profiles. They first exchange their trust-social profiles, which are assigned by the server. Note that a user's identity (usually we adopt the user's public key as its identity), his trust (or distrust) social profiles, and the profile's valid time are signed together by the server's private key; the attacker cannot create or modify it. After obtaining the trust profile of user S, user V will verify it first, since the attackers may steal others' profiles. V generates a short random number and encrypts it by the public key of S; S needs to find out the random number,

[1]The conditions in which Sybil actively friends honest users is out of the scope of this paper, since each Sybil normally interacts with honest users, instead of cheaply creating a fake identity.

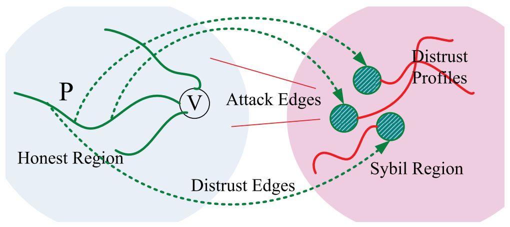


Fig. 13. The generation of distrust social profiles. In order to obtain enough distrust relations, the server first creates 3 short length random paths from V. Then, the server takes all the distrust relations of the nodes on the paths as a distrust seed set. In this figure, we use the shadowed green circles to indicate the seeds. From each seed, the server generates a random path along the trust edges, and these edges compose the distrust social profile. Here, we use solid red lines to represent the random paths belonging to V's distrust profile.

encrypt the random number by V's public key, and send it back to V.

After the process of mutual verification, node 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. 14. If there is a common node on both a verifier path and a suspect path, then we say that the suspect path is verified once; when a suspect path has been verified more than k_t times, where k_t is a constant, 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 a total of K random paths in a trust social profile. In regard to V, the trust score of S is given by:

$$Trust(V, S) = \frac{Ver(V, S)}{K} \tag{1}$$

For the computation of distrust score, SNSD considers both the distrust social profile of V and the trust social profile of S, as shown by Fig. 15. We name the paths from V's distrust social profile *distrust verifier paths*, and we use K' to represent the size of V's distrust social profile. Similar to the computation of trust score, 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:

$$DisTru(V, S) = \frac{Dis(V, S)}{K'} \tag{2}$$

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, $1 \geq \alpha > \beta \geq -1$.

$$L(V, S) = \begin{cases} \text{Trusted} & \text{for} \quad z \geq \alpha \\ \text{Neutral} & \text{for} \quad \alpha \geq z \geq \beta \\ \text{Distrusted} & \text{for} \quad \beta \geq z \end{cases} \tag{3}$$

The accuracy of a majority of the existing Sybil defense systems is bounded by the number of attack edges. In order to improve the accuracy of our system, a special pruning algorithm called Sybil gateway-breaking (SGA) will run on the remote server once in a while. Essentially, the algorithm prunes some suspicious edges of the signed social network stored on the server. Consider that all of the paths connecting honest and Sybil nodes must go

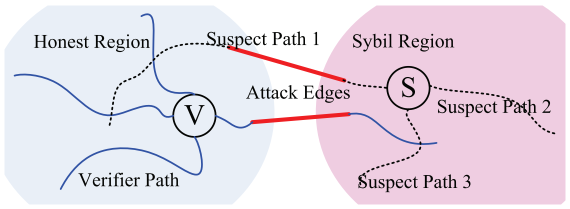


Fig. 14. The computation of trust score. When the honest user V encounters a stranger S, they first exchange their social profile. In this figure, the solid blue lines indicate V's trust social profile, and the dashed black lines represent the trust social profile of S. In V's view, he regards himself as the verifier, and the paths in his trust profile are named as the verifier paths. Moreover, V regards the paths of S as suspect paths. V locally checks the connectivity between his profile and the others'. A trust score will be determined based on the intensity of connectivity. In this figure, if V sets the verifier threshold $k_t = 2$, then only suspect path 1 will be fully verified. Therefore, $Ver(V, S) = 1$, $|K| = 3$, and $Trust(V, S) = 1/3$.

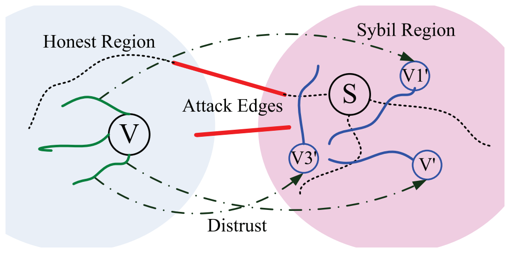


Fig. 15. The computation of distrust score. In this example, the distrust social profile of V contains 3 distrust paths, which is represented by solid blue lines. V will compute a distrust score by using his distrust profile and S's trust profile. If we define a verified distrust path as a suspect path that comes across at least half of the distrust verifier paths, then, in this figure, only suspect path 3 is verified. As a result, $Dis(V, S) = 1$, $K' = 3$ and $DisTru(V, S) = 1/3$.

through the attack edges, and that the number of attack edges is limited. So, the connectivity from an honest region (a group of honest nodes) 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 some attack edges, the accuracy of *any* social network-based Sybil defense will surely be enhanced.

The SGA consists of three sub-algorithms, as shown by Algorithm 2: (1) Suspicious Edge Selection Algorithm; (2) Gateway Verification Algorithm (GVA); and (3) Attack Edge Detection Algorithm. Note that a gateway is an edge that connects two communities together. SGA first selects high centrality nodes/edges as the suspicious ones, and then verifies whether the associated edges are gateways. For gateway edges, the algorithm further detects whether they are Sybils. There are two options for finding the suspicious edges. The first one is based on node centrality, and the second one focuses on the edge centrality.

First, the attacker may adopt a target attack by densely adding attack edges to a target node and its neighbors; although the target region and Sybils are well connected, the nearby regions of the target may observe a weak connectivity from themselves to Sybils. Therefore, the server can first generate a local map (G) of each node, and then compute each node's centrality on G. The gateway verification algorithm is only applied to the edges, which are connected to the high centrality nodes. The criteria of centrality we used is called the betweenness centrality [30],

Algorithm 2 Sybil gateway-breaking algorithm (SGA)

```
1: Suspicious Edges Selection:
2: Select edges with high local betweenness as suspicious edges.
3: Use signed network-based Sybil defense algorithm (section V) to determine honest and Sybil.
4: Find shortest paths from the honest nodes to Sybil nodes.
5: Compute the visiting frequency of edges.
6: Take the edges with high visiting frequency as suspicious.
7: Gateway Verification:
8: Generate the initial neighbor set, ${u}$, ${v}$, and ${w}$.
9: Compute the number of unique paths from ${v}$ to ${u}$, and from ${v}$ to ${w}$
10: for Predefined times do
11:     Respectively add $\Delta k$ disjoint neighbors into ${u}$, ${v}$, ${w}$.
12:     Compute the number of unique paths from ${v}$ to ${u}$, and from ${v}$ to ${w}$
13:     Compute the growing speeds of unique paths, $S_{vu}$ and $S_{vw}$
14:     if $|S_{vu} - S_{vw}|$ is greater than a threshold then
15:         $E^+_{uw}$ is a gateway;
16: $E^+_{uw}$ is not a gateway.
17: Attack Edge Detection:
18: Find attack edges from detected gateways by distrust relations; break them.
```

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.

The second option is that, 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 distrusted 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.

Nodes within the same community share certain characteristics [31]. Whether two nodes are 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. 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.*

Since the amount of unique paths connecting two nodes is bounded by the node-degrees of the ends, we compute the unique paths from a region to another region. Assume that the ends of a given edge are nodes u and w. Let v

be the third node for checking the connectivity. $UP(u,w)$ represents the number of unique paths from u to w.

Generally speaking, based on the community structure of these three nodes, there are three possible cases that can be observed on connectivity:

1) Either u or w shares the same community as v.
2) Both u and w are located in the same community as v.
3) None of them come from v's community.

For the first case, assuming that node u resides in the same community as v, $UP(v,u)$ 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. As we mentioned above, we count the amount of unique paths from region to region. 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 those from u's region. However, for the second and third cases, we will not observe such differences. Based on this feature, we design a gateway verification algorithm for checking whether a given edge is a gateway, or not.

GVA respectively calculates the growing speeds of unique paths from v to u, and from v to w. 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 the amount of unique paths between the regions. Since we only care whether the edge E^{+}_{uw} is a gateway or not, we only need to check the existence of the growing speeds' difference for a given node v.

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 distrusts the other, it is very likely that the gateway is an attack edge. However, since the server does not know the exact community structures of the created social graph, the scheme of counting the total amount of distrust links from one community to another is infeasible. Consider that the majority of random paths are trapped inside their own communities; instead of all of the nodes within a community, we adopt random sampling to estimate the intensity of distrust relations between communities.

Our attack edge detection algorithm works as follows. First, for each gateway, the server temporarily breaks it. Then, from both its 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 directly 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. Finally, the server keeps the gateways with large intersection sets being broken, and restores other gateways.

D. Multiple Local Communities-based Sybil Defense [32]

Based on real data, paper [33] finds out that the probability of contagion is tightly controlled by the number of connected components (local communities) in an individuals contact neighborhood, rather than by the actual size of the neighborhood. Based on this observation, we wonder whether there is a better way to measure the trust values among nodes in a peer-to-peer system. Consider that traditional social network-based Sybil defenses, such as SybilGuard, estimate the trust value by measuring the strength of trust paths between any pair of nodes. Since trust and contagion are closely related, instead of computing the trust paths at the node-level, we should measure the strength of the paths at the local community level. Moreover, in traditional social network-based Sybil defense algorithms, each node must locally determine whether to accept all others' nodes. Therefore, the overall computation cost of these algorithms is a problem, especially in some large systems. Consider that, in a social graph, the neighbors of an honest user are still honest. Once similar users are clustered into groups, a Sybil detection algorithm only needs to determine whether a group (clump) is Sybil. Here, we propose three social clump-based Sybil defense algorithms, which can save plenty of computation costs, while maintaining the same level of accuracy as the traditional solution.

In social networks, all friendships between honest users are established based on their physical interactions, and therefore, the edges between users indicate credibility between them. In order to impersonate real users, the attacker has to create social profiles for each sybil identity, and manipulate the friendship edges among them. Consider an extreme case in which the attacker copies whole social networks of real users, and replaces the real identities with the Sybil ones. It is impossible to discriminate Sybil ids from the real ones by simply checking the friendship structures. However, since friendship is a mutual relation and is built by physical interactions between users, attackers cannot tamper with the friendships of real users, and therefore, the total number of friendships between Sybils and honest users is few; we use g to represent the number. In most cases, g should be smaller than the average number of honest users' friends. Suppose that we have previous knowledge about the value of g, and that the attacker can arbitrarily add these g attack edges on a social graph.

The strength of a friendship is measured by the number of common friends. A pair of friends usually has several common friends. $N(u)$ is adopted to represent the friend set of node u, and let $|\cdot|$ stand for the cardinality of a given set. If the cardinality of the common friend set is large enough, then both users should be clustered into the same clump.

***Theorem** 1:* When a pair of friends, u and v, shares more than g common friends ($|N(u) \bigcap N(v)| > g$), the users u and v must be both honest or both Sybil.

However, the requirement for having more than g common friends causes some honest nodes to have no clump to participate. Intuitively, in our solution, the more nodes that are clustered into clumps, the more computational costs that can be saved. Consider that an honest clump is a subgraph of the honest part. It should be cohesive inside, and have low betweenness from the global view. Based on this observation, we propose a pair of rules for finding the members of a clump. A collection of nodes can locally form into a clump if and only if the majority of their friends reside in the same clump, and the nearby non-clump members are still reachable within several hops

Algorithm 3 Distributed Clump Generation Algorithm

```
1: Each node $u$ sets up its label $l_u = u$
2: if $|N(u)| > g$ then
3:     if $\exists v \in N(u)$ and $|N(v) \cap N(u)| > g$ then
4:         Set the labels of $l_u$ and $l_v$'s clumps as $\min(l_u, l_v)$
5:     if $\exists v \in N(u)$ and $|N(v) \cap N(u)| \geq \alpha|N(u)|$ then
6:         Let $C = \{w | l_w = l_v\}$
7:         if There are at least $g - |N(u) \cap C| + 1$ unique paths between $C$ and $N(u) \cap \overline{C}$ then
8:             Set the labels of $l_u$ and $l_v$'s clumps as $\min(l_u, l_v)$
9: else
10:    if $\forall v, w \in N(u), l_v = l_w$, $l_u \neq l_v$ then
11:        Set the label of $u$ as $l_u = x$
12: When the labels of $N(u)$ are stable, run Algorithm 2
```

after removing the clump. In details, the members of constructed clump C must satisfy the following two rules:

Rule 1: if $u \in C$, then $|N(u) \cap C| \geq \alpha|N(u)|$

Rule 2: if $u \in C$, then there exists $g - |N(u) \cap C| + 1$ unique paths between C and $N(u) \cap \overline{C}$, whose length is less than β.

where $\overline{C}$ represents the complementary set of C. α, β are two parameters.

***Definition** 2: Unique paths indicate a group of paths connecting two distinct nodes or regions without sharing a common edge.*

We adopt rule 1 to guarantee that nodes are well connected inside each clump. Rule 2 is set up in order to keep the bridge structure unchanged after grouping. However, for the honest nodes with lower node degree, they cannot be accepted by any clump by using Rules 1 and 2. We provide an additional rule:

Rule 3: if $|N(u)| < g$ and $\forall v \in N(u), v \in C$, then $u \in C$

Note that Rules 1 and 2 work together, and they only apply on the nodes with a large node degree. The additional rule, Rule 3, only suits nodes with a small friend set.

Algorithm 3 provides the procedure for clump generation. We associate one variable, called label, with each node. For the ease of description, we use l_u to represent the label of u. Initially, each node uses its own identity as the label. Based on theorem 1, when two nodes, u and v, have more than g common friends, their resided clumps will merge into one (by adjusting the labels of both clumps' members into the least identity value). If the cardinality of the common neighbor set between a node u and its neighbor v satisfies Rule 1, then Rule 2 will be checked. When the rule is satisfied, then u's resided clump merges with v's. For a node, whose node degree is less than g, if all of its neighbors have the same label (Rule 3), the node will join its neighbors' clump.

Algorithm 4 Clump Pruning Algorithm

```
1: Based on labels, find the members of $u$'s clump $C$
2: if $|N(u) \cap C| < \alpha|N(u)|$ then
3:     Set the label of $u$ as $l_u = u$
4:     Re-conduct Algorithm 1 by replacing line 5 with Rule 1
5:     Clump $C$ will update its label based on connectivity
```

Note that line 5 of Algorithm 1 is not exactly the same as Rule 1. Since each node only regards itself as a clump in the very beginning, we need to put several similar nodes together. Admittedly, Theorem 1 can find the similar nodes, which definitely belong to the same clump. However, the number of nodes may not large enough. So, we first adopt a modified Rule 1 as shown in line 5 of the algorithm, and then, after the process of labeling becomes stable, a special pruning algorithm will run on every node (except the nodes that satisfy Theorem 1), as shown by Algorithm 4. Essentially, the pruning algorithm checks whether the current label of a node satisfies Rule 1. If not, the node needs to redetermine its label by running Algorithm 1 again.

In traditional random walk-based Sybil defense algorithms, nodes need to launch a bunch of random walks since, initially, they only trust themselves. Moreover, from security concerns, these random walks are usually associated with secure signatures at each step. It is obvious that costs of the algorithms are too high. Consider the fact that most neighbors of an honest node are still honest. We propose three clump-based random walk strategies. The proposed random walks can increase the efficiency of all existing random walk-based Sybil defense algorithms.

The strategy of Clump-based Random Walks (CRW) is modified from traditional lazy random walk, where each walker randomly picks up one neighbor of its current location as the destination of next hop. By using CRW, a random walker will transit from one clump to another, and the stationary distribution of each node depends on both the degree of the node and the stationary distribution of its resident clump. In order to keep the distribution of each node unchanged, we add a self-loop at each clump.

Given a clump $c =< V_c, E_c >$, $c \subseteq G$, the transition probability $p(c, c')$ from clumps c to c' is defined as follows:

$$p(c, c') = \frac{\sum_{i \in c, j \in c'} a_{ij}}{\sum_{i \in c} deg(i)}.$$

More specifically, when $c = c'$, $p(c, c) = 2|E_c| / \sum_{i \in c} deg(i)$.

***Theorem** 2:* In CRW, clump C's stationary distribution π_C equals the summation of stationary distributions of its members (π_i) in traditional random walk. $\pi_C = \sum_{i \in C} \pi_i = \sum_{i \in C} \frac{|N(i)|}{2|E|}$, where $|E|$ is the total amount of edges in G.

A majority of existing social network-based Sybil defense algorithms adopt short-length random walks. As was introduced in the background section, there is a time gap between the first mixing time at the honest region and the mixing time of the whole graph. Essentially, the Sybil defense algorithms explore the features related with this time gap. The length of the short-length random walk usually is less than or equal to the first mixing of the honest region. Since we have already known that the honest region is fast-mixing, and that the mixing-time of a fast-mixing network is proportional to the cover time of the network, we will discuss the variance of cover time by using clumps.

***Theorem** 3:* The cover time on honest region will be reduced in clump-based random walks.

In a majority of random walk-based Sybil defense algorithms, the walkers are always easily spread to intensively connected nodes. Moreover, the similarity between social nodes in social networks can be used for measuring the strength of social links [34]. Based on the observations, we claim that by giving random walkers the ability to jump among clumps with high similarity, the mixing time at honest regions can be reduced. In other words, by assigning a higher probability of being visited, the speed of spreading trust scores to far away nodes can be accelerated.

For two given clumps c_i and c_j, which are not directly connected, the similarity between them is defined as $S(c_i, c_j)$.

$$S(c_i, c_j) = \frac{N(c_i) \bigcap N(c_j)}{N(c_i) \bigcup N(c_j)}$$

c_j of both c_i and c_j, if the cardinality of the percentage of the common neighbor sets, $|S(c_i, c_j)|$, is greater than a predefined threshold γ, the random walkers are allowed to jump between c_i and c_j. Essentially, CRWGJ shrinks the length of jumping in intensively connected regions from 2-hops to single hop.

Before conducting any random walks, every clumps in the given network needs to locally measure the common neighbor sets with direct neighbors, and compute similarity scores with their 2-hop neighboring clumps. If the result score is greater than the threshold, a virtual edge will be generated between the corresponding clumps. Suppose that the graph consisting of all virtual edges is presented by E_v. Clearly, $E_v \subseteq A^2$ ($A^2 = [a^2_{ij}]$), since a virtual edge indicates that its two ends intensively connect with each other in 2-hops. During the computing of CRWGJ, a random walker will equally select an edge from its direct neighbors or virtual edges to be the destination of the next hop.

Based on the above strategy, the nodes with a large neighbor set, or that are intensively connected with surrounding neighbors, will have higher stationary distribution values. In the case that attack edges are randomly attached to honest nodes, CRWGJ can reduce the mixing time of the honest part and the false positive rate[2] of Sybil detection results.

***Theorem** 4:* The mixing time of CRWGJ at the honest region is less than that of CRW.

However, considering target attacks, where the attacker may do everything she can to establish social relations with a target node and its neighbors, CRWGJ may also result in the appearance of extra virtual edges between honest and Sybil parts; this is the trade-off between speed and accuracy.

Consider the fact that the position of a random walker will be independent from the initial location after several steps of transition. Most traditional random walk-based Sybil defense algorithms have two problems. First, although the probability for random walkers transiting from honest regions to Sybil regions is small, once a walker enters the Sybil regions, it is hard to get out. Second, when the number of random walkers is not large enough for a given

[2]False positive rate indicates the percentage that an honest node is regarded as Sybil.

graph, a node with a high degree may get unfair trust results. This may be due to the node trusting some relatively far away nodes instead of believing in some nearby nodes, which are not visited by the random walkers.

Here, we propose a new strategy called Clump-based Random Walks with Limited Memory (CRWLM). In CRWLM, each random walker keeps a previous location list $l =< l_0, l_1, \ldots, l_{h-1} >$ with length h. During transition, the walker can either move to one of its 1-hop neighbors, or jump back to the nodes in the list. Note that random walk with restart (RWR) is a special case of CRWLM, where the size of memory is 1, and each clump contains only one member. Assume that a random walker locates at clump c. The transition probability is given by the following:

$$p(c,c') = \begin{cases} \frac{W_{cc'}}{R+W_{c\overline{c}}} + \frac{R}{h(R+W_{c\overline{c}})} & \text{for} \quad c' \in N(c), c' \in l \\ \frac{1}{R+W_{c\overline{c}}} & \text{for} \quad c' \in N(c), c' \notin l \\ \frac{R}{h(R+W_{c\overline{c}})} & \text{for} \quad c' \notin N(c), c' \in l \\ 0 & \text{for} \quad \text{otherwise} \end{cases}$$

where $W_{cc'} = \sum_{i \in c, j \in c'} a_{ij}$ and $W_{c\overline{c}} = \sum_{i \in c, j \notin c} a_{ij}$.

The structure of traditional random walk's trajectory is a line, while CRWLM presents a dendriform structure. Moreover, such a tree contains an h-length trunk, and several branches. One problem with traditional RWR is that, when a node i sends out k random walkers, $k >> deg(i)$, the walkers will repeatedly visit i's nearby nodes, which means the walkers do not spread out well. When adopting CRWLM, each walker first randomly creates a trunk, and then expands several relatively short branches from it. Clearly, in CRWLM, a random walker has a better chance of visiting nearby nodes of the walk initiator, and possesses more opportunities for returning to honest regions after having gotten into a Sybil region.

VIII. Prediction

Here, we provide several directions for further research. First, since there are certain types of friendships that are private, what will happen if we combine the privacy and Sybil attack problems together? Second, can we use the community detection method to deter Sybil attacks in a directed network? Third, the community detections are always time-consuming; can we find some other light algorithm which can detect the community structures quickly and accurately? Fourth, social networks contain multiple dimensions, how are we to combine this feature with traditional approaches? Based on the number of potential research directions, we believe that the community-based Sybil defenses may become the fourth phase of anti-Sybil techniques.

IX. Conclusion

Peer-to-peer systems play an ever-increasingly important part of our daily lives. However, most of the peer-to-peer systems are vulnerable to Sybil attacks. In order to design more efficient and practical Sybil defenses, we write this survey. This article is the first survey focusing on the developments of Sybil defenses. We first give the definition of Sybil attacks, and provide the classification of Sybil attacks. Then, we give several realistic systems which are vulnerable to Sybil attacks. After that, defense mechanisms and their corresponding strengths and weaknesses were discussed. Unlike other surveys, we describe these mechanisms according to anti-Sybil approaches' developing stages. By the end of this survey, we provide some directions for future research.

References

[1] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *Proc. of ACM SIGCOMM*, vol. 36, no. 4, 2006, pp. 267–278.

[2] H. Yu, "Sybil defenses via social networks: a tutorial and survey," *SIGACT News*, vol. 42, no. 3, pp. 80–101, 2011.

[3] B. Viswanath, A. Post, K. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *Proc. of ACM SIGCOMM*, vol. 40, no. 4, 2010, pp. 363–374.

[4] X. Zheng, Y. Lai, K. Chow, L. Hui, and S. Yiu, "Sockpuppet detection in online discussion forums," in *Proc. of IEEE IIH-MSP*, 2011, pp. 374–377.

[5] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," *Technical report, Stanford University*, 1999.

[6] J. Golbeck, B. Parsia, and J. Hendler, "Trust networks on the semantic web," *Cooperative Information Agents VII*, pp. 238–249, 2003.

[7] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proc. of ACM IPSN*, 2004, pp. 259–268.

[8] J. Ledlie and M. Seltzer, "Distributed, secure load balancing with skew, heterogeneity and churn," in *Proc. of IEEE INFOCOM*, vol. 2, 2005, pp. 1419–1430.

[9] G. Mathur, V. Padmanabhan, and D. Simon, "Securing routing in open networks using secure traceroute," in *Technical report MSR-TR-2004-66, Microsoft Research*, 2004.

[10] J. Douceur, "The sybil attack," *Peer-to-Peer Systems*, 2002.

[11] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc Networks*, vol. 1, no. 2, pp. 293–315, 2003.

[12] B. Awerbuch and C. Scheideler, "Group spreading: a protocol for provably secure distributed name service," in *Automata, Languages and Programming*. Springer, 2004, pp. 183–195.

[13] R. Gatti, S. Lewis, A. Ozment, T. Rayna, and A. Serjantov, "Sufficiently secure peer-to-peer networks," *Workshop on the Economics of Information Security*, 2004.

[14] N. Margolin, B. Levine, N. Margolin, and B. Levine, "Quantifying and discouraging sybil attacks," *Computer Science Technical Report*, vol. 67, 2005.

[15] Y. Reddy, "A game theory approach to detect malicious nodes in wireless sensor networks," in *Proc. of IEEE SENSORCOMM*, 2009, pp. 462–468.

[16] C. Piro, C. Shields, and B. Levine, "Detecting the sybil attack in mobile ad hoc networks," in *Proc. of IEEE Securecomm*, 2006, pp. 1–11.

[17] B. Xiao, B. Yu, and C. Gao, "Detection and localization of sybil nodes in VANETs," in *Proc. of ACM DWANS*, 2006, pp. 1–8.

[18] Q. Zhang, P. Wang, D. Reeves, and P. Ning, "Defending against sybil attacks in sensor networks," in *Proc. of IEEE ICDCS*, 2005, pp. 185–191.

[19] M. Demirbas and Y. Song, "An RSSI-based scheme for sybil attack detection in wireless sensor networks," in *Proc. of IEEE WoWMoM*, 2006, pp. 564–570.

[20] C. ZHENG and D. S. GILBERT, "Thwarting sybil attacks and malicious disruption in wireless networks," *http://www.comp.nus.edu.sg/ zheng-10/talk/grp-2012-09-14-paper-v3.pdf*.

[21] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. of IEEE CCS*, vol. 2, 2000, pp. 775–784.

[22] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. of ACM MobiCom*, 2000, pp. 32–43.

[23] Y. Hao, J. Tang, and Y. Cheng, "Cooperative Sybil Attack Detection for Position Based Applications in Privacy Preserved VANETs," in *Proc. of IEEE GLOBECOM*, 2011, pp. 1–5.

[24] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: a near-optimal social network defense against sybil attacks," in *Proc. of IEEE Symposium on Security and Privacy*, 2008, pp. 3–17.

[25] N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting," in *Proc. of USENIX NSDI*, 2009, pp. 15–28.

[26] P. Dandekar, A. Goel, R. Govindan, and I. Post, "Liquidity in credit networks: A little trust goes a long way," *Arxiv preprint arXiv:1007.0515*, 2010.

[27] B. Viswanath, M. Mondal, K. Gummadi, A. Mislove, and A. Post, "Canal: Scaling Social Network-based Sybil Tolerance Schemes," in *Proc. of ACM EuroSys*, 2012, pp. 309–322.

[28] P. Tsuchiya, "The Landmark Hierarchy: A new hierarchy for routing in very large networks," in *Proc. of ACM SIGCOMM*.

[29] W. Chang, J. Wu, C. Tan, and F. Li, "Sybil defenses in mobile social networks," *submitted for publication*.

[30] P. Marsden, "Egocentric and sociocentric measures of network centrality," *Social Networks*, vol. 24, no. 4, pp. 407–422, 2002.

[31] S. Chan, I. Leung, and P. Liò, "Fast centrality approximation in modular networks," in *Proc. of ACM CNIKM*, 2009, pp. 31–38.

[32] W. Chang and J. Wu, "Clump-based sybil defense in crowdsensing," *submitted for publication*.

[33] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, "Structural diversity in social contagion," *Proceedings of the National Academy of Sciences*, vol. 109, no. 16, pp. 5962–5966, 2012.

[34] A. Mohaisen, N. Hopper, and Y. Kim, "Keep your friends close: incorporating trust into social network-based sybil defenses," in *Proc. of IEEE INFOCOM*, 2011, pp. 1943–1951.