SyMon: A practical approach to defend large structured P2P systems against Sybil Attack

Jyothi B. S., Dharanipragada Janakiram
Dept. of CSE, Indian Institute of Technology, Madras
jyothibs@cse.iitm.ac.in, djram@iitm.ac.in

***Abstract*— Sybil attack is one of the most challenging problems that plague current decentralized Peer-to-Peer(P2P) systems. In Sybil attack, a single malicious user creates multiple peer identities known as sybils. These sybils are employed to target honest peers and hence subvert the system. In this paper, we describe a novel solution that enables all honest peers to protect themselves from sybils with high probability in large structured P2P systems. In our proposed sybil defense system, we associate every peer with another non-sybil peer known as SyMon. A given peer's SyMon is chosen dynamically such that the chances of both of them being sybils are very low. The chosen SyMon is entrusted with the responsibility of moderating the transactions involving the given peer and hence makes it almost impossible for sybils to compromise the system. We show the effectiveness of our proposed system in defending against Sybil attack both analytically and experimentally. In addition to this, we explore the practical feasibility of our proposed solution by considering reputation systems for P2P based file sharing applications and P2P systems susceptible to Denial-of-Service(DOS) attack, systems known to be highly vulnerable to Sybil attack, as our case studies. In each of our case studies, we discuss possible ways in which our solution can be employed to defend the system against Sybil attack.**

Keywords: Sybil Attack, P2P Networks, Structured Overlay, Reputation System, DOS Attack

I. Introduction

Peer-to-Peer(P2P) overlay networks are known for their much desired attributes like openness, anonymity, decentralized nature, self-organization, scalability and fault tolerance. Unlike in traditional client-server systems, every peer[1] in P2P systems plays a symmetrical role as a client as well as a server. These networks are typically built over the Internet and hence known as overlay networks. The core services offered by overlay networks include efficient object location and routing within the network.

Many popular P2P applications like P2P file sharing systems (e.g. Napster [1], BitTorrent [2], Kazaa [3]) have been built over the overlay networks and use the services offered by them. Of late, the increasing popularity of P2P applications has attracted the unwanted attention of malicious users. These malicious users exploit some of the salient features of P2P systems like openness and anonymity to abuse the system.

Among the security attacks that plague existing P2P systems, Sybil attack [4] is the most difficult and challenging problem to solve. In Sybil attack, a single malicious user creates a large number of peer identities called *sybils*. These sybils are used to launch other types of security attacks, both at the application level and at the overlay level. At the application level, sybils can target other honest peers while transacting with them whereas at the overlay level, sybils can effectively disrupt the services offered by the overlay layer like routing, data storage, lookup protocol, etc. In this paper, we mainly concentrate on analyzing the impact of Sybil attack and its defense strategies at the application level in P2P systems.

In P2P systems, the cost associated with joining the system - *entry barrier*, is typically low or nonexistent to encourage new peers to join the system. Because of the low or zero *entry barrier*, the cost incurred by malicious users while launching Sybil attack is very minimal. In addition to this, anonymity, one of the desirable features of P2P systems, makes the problem of sybil detection much harder to solve. Some examples of P2P applications which are known to be vulnerable to sybils include P2P file sharing systems integrated with a reputation system and servers that can be subjected to Denial-Of-Service(DOS) attack.

In P2P file sharing applications, sybils can lower the content availability of the system by serving polluted content to other honest peers in the system. Even P2P reputation systems are ineffective in preventing this. In these P2P reputation systems, sybils can perform *ballot-stuffing* [5] by spoofing transactions among themselves in order to raise their reputation. They can spoof transactions by logging fake feedbacks without actually performing them. Most importantly, there is no limit on the number of fake transactions that can be performed by even a very small number of sybils. Hence, in these systems, the reputation of a peer may not truly reflect its past behavior. Once sybils attain high reputation, they can start *milking* their reputation by serving corrupt files.

A P2P server can protect itself from Denial-of-Service (DOS) Attack by forcing each of its clients to solve a computational puzzle [6], [7] (of some difficulty level) before serving its request. Unless sybils possess unlimited amount of computational resources, this approach can discourage them from sending a large number of requests to the server. However, the process of forming the puzzle challenge, when performed by the server itself or any other centralized entity [8], can unwittingly subject it to DOS attack. On the other hand, if any random peer is chosen as the puzzle server [9], then sybils can easily circumvent this by electing one of their sybils as the puzzle server.

[1] A *peer* refers to a node/computer in the case of P2P overlay networks and a human user in the case of P2P applications.

Motivation: Researchers have attempted to defend against Sybil attack through various approaches. In [4], it has been stated that Sybil attack cannot be fully prevented unless a vigilant central authority generates a single identity for a peer. However, the central authority can itself act as a single point of failure and a performance bottleneck. Various distributed approaches have been proposed to control the peer identity generation process [10], [6], [11]. The main goal of these approaches is to limit the total number of sybils so as to contain their adverse impact on the system. These approaches are useful to protect a class of P2P applications that require the total number of sybils to be a small fraction of the overall size of the system. In systems like [12], which rely on online voting schemes to gather user opinion and then rank user-generated online content, it is important to prevent sybils from *out-voting* non-sybils. In such systems, limiting the total number of sybils can be effective.

There exists another class of P2P applications where even a limited number of sybils can wreak havoc in the system. For example, P2P reputation systems are known to be highly susceptible to even a small number of sybils [13], [14].

In typical Internet scale P2P applications, transactions involving unknown peers, is the norm rather than an exception. In the absence of any information about the unknown peers, the chances of the honest peers getting cheated by sybils are very high. This necessitates the need for a mechanism by which every honest peer can protect itself from sybils during transactions.

Our approach: In this paper, we describe a fully decentralized novel solution to protect honest peers from sybils in large structured P2P systems. In our proposed system, every peer is *associated* with another peer, known as ***SyMon***(***Sy****bil* ***Mon****itor*). The *SyMon* of any given peer is chosen dynamically such that the probability of both of them being sybils is very low. The chosen *SyMon* prevents the given peer(sybil) from targeting other honest peers by monitoring the transactions involving the given peer.

The process of monitoring a transaction is application dependent. Initially, we assume that it is robust to enable *non-sybil SyMon* peers to protect honest peers from sybils during transactions. With this assumption, we mainly focus on the *SyMon* selection process. We propose four methods for *associating* any given peer with its *SyMon*. Our simple discovery protocol can find *SyMon* for any given peer in the system. Moreover, the selection of a *non-sybil* peer as *SyMon* is verifiable and this verification can be performed by any peer in the system. Each of our selection methods differs in its efficacy and the overhead incurred while choosing an eligible *SyMon* in the system. We analyze our sybil defense scheme theoretically and show how hard it is for an *adversary* to break our system. We also show the effectiveness and scalability of our system experimentally.

We explore the feasibility of our proposed solution by integrating it with some of the practical P2P systems, known to be susceptible to sybils. In this paper, we discuss possible ways of incorporating *SyMon* into reputation systems for P2P based file sharing applications and P2P systems known to be vulnerable to DOS Attack. In each of these case studies, we also highlight the role played by the monitoring process in aiding *SyMon* in its fight against sybils.

Summarizing, we claim that this paper makes the following contributions.

- *SyMon*, our sybil defense scheme protects honest peers from sybils by dynamically choosing *non-sybil* peers to monitor the transactions. To the best of our knowledge, ours is the first attempt to defend against Sybil attack through the transaction monitoring process.
- Each of our four proposed *SyMon* selection methods inflicts various costs on malicious users involved in Sybil attack and hence offers different security margins against sybils.
- As part of our case study on reputation systems for P2P based file sharing systems, we discuss potential ways of exploiting *SyMon* to discourage sybils from increasing their reputation through *ballot-stuffing* and hence from lowering the content availability of the system.
- Our case study on P2P based systems prone to DOS attacks suggests possible ways of safeguarding P2P servers from sybil client peers with the aid of *SyMon*.

The rest of the paper is organized as follows. In section II, we provide the necessary background information relevant for our solution. Section III describes our system model. In section IV, we present details about our system and then discuss its strengths and weaknesses against a range of attacks in section V. We show the robustness of our proposed solution through simulation results in section VI. We compare our solution with existing research work in section VIII. Section VII describes our case studies on reputation systems for P2P based file sharing applications as well as P2P systems prone to DOS Attack. Finally, we conclude this paper in section IX with a mention about possible future research directions.

II. BACKGROUND

A. P2P Systems

P2P networks are characterized by a network architecture of interconnected peers wherein each peer plays the dual role of a client as well as a server. These systems can function well, self-organize and even scale in the presence of transient peers. Moreover, their most attractive feature is that the resources to form the network are generally contributed by the peers themselves unlike traditional systems which are built from the resources dedicated by a central authority. These systems offer routing infrastructure for efficient storage as well as retrieval of data objects. Many applications have been built over the P2P overlay substrate which exploit the services offered by the underlying overlay infrastructure. Some examples of currently popular P2P applications include Gnutella [15], BitTorrent [2], Skype [16], etc.

Broadly, a P2P overlay network can be classified as either a *structured overlay* or an *unstructured overlay* based on the overlay network topology organization [17].

1) Unstructured Overlay: Unstructured overlays, like Gnutella [15], are characterized by peers which are connected in the form of a random graph. Hence, the system is resilient to frequent peer joins and leaves. In these systems, there is no correlation between peers and the data stored by them. Hence, controlled flooding or random walks is the main strategy adopted to locate the required data. This form of routing is effective in locating highly replicated (popular) data. However, a search for unpopular data (i.e. data that is not widely replicated) increases the system load as the query has to be routed to a large number of peers. Moreover, the system does not scale well since the load incurred by peers increases linearly with the total number of queries as well as the size of the system.

2) Structured Overlay: In structured overlays like Pastry [18] and Chord [19], the network topology is well organized. These systems use *Distributed Hash Tables(DHT)* so that the data is placed deterministically on specific peers. Both peers and objects are assigned *unique* identities randomly from the *same* identity space, known as peer identities and *keys* respectively. A live peer whose identity is closer to some object's key is responsible for its storage. This data storage strategy is scalable as it can give stronger guarantees for locating both popular and unpopular data in no more than log_N steps on average where N is the size of the network. However, it can also lead to frequent migration of objects between peers or even data loss due to excessive peer joins and leaves.

In subsequent paragraphs, we provide a brief overview about Pastry and Chord with a focus on aspects relevant for our solution.

a) Pastry: In Pastry, every peer maintains the routing state information that includes a routing table, a *neighborhood set* (containing a set of *live* peers geographically proximal to the given peer) and a *leaf set* (containing a set of *live* peers on either side of the given peer in the identity space).

A *live* peer closest to the given peer in the identity space can be found from its *leaf set* entries. Similarly, 'k' closest *live* neighbors of a given peer can be found in the given peer's *leafset* or in the *leafsets* of a few of its neighbors.

b) Chord: In Chord, peers are ordered in form of an identifier circle (known as the *chord ring*), based on their peer identities. Here, the routing state information maintained at each peer includes a routing table and pointers to its immediate *successor* and *predecessor* peers in the ring.

A *live* peer closest to the the given peer in the identity space is either its *predecessor* or *successor* in the chord ring. Similarly, 'k' closest *live* neighbors of a given peer can be found on either side of the given peer in the ring.

III. System Model

In this section, we describe our system model and the assumptions made.

A. P2P Overlay

The system comprises of a fully decentralized large structured P2P overlay infrastructure like Pastry or Chord that implements DHT. We also assume that the underlying overlay infrastructure exposes APIs like:

- *p = Lookup(x)*: Find a *live* peer 'p' closest to the given key 'x' in the identity space.
- *Route(x, Msg)*: Route the given message ('Msg') to a *live* peer whose identity is closest to the given key 'x' in the identity space.
- *npset = findKClosestNeighbors(p)*: Find 'k' closest *live* neighbors ('npset') of the given peer 'p' in the identity space. When k=1, it returns the closest *live* neighbor of the given peer 'p'.

The underlying overlay infrastructure is assumed to be outside the control of malicious users.

B. Peer Identity Generation

Before joining the system, every peer creates its public/private key pair using 1024-bit RSA [20] and then hashes the public key using SHA-1 [21] to generate its peer identity. The peer generates its Identity Certificate (using SPKI [22] or X.509 [23]) containing its identity (represented as a 40-digit hexadecimal number) and its public key.

A cryptographic hash function like SHA-1 is employed so that peer identities are generated randomly and also distributed *uniformly* over the circular identity space ranging from 0 to $2^{160} - 1$. This approach prevents sybils from choosing their own desired identities [24].

The RSA public/private key pair is used to encrypt all messages exchanged between peers so as to assure authentication, confidentiality and non-repudiation.

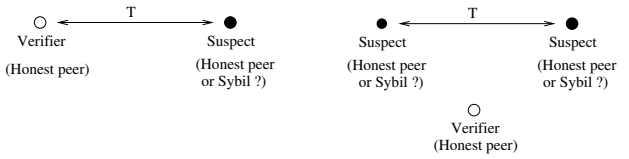
C. Proof-Of-Work

In order to prevent malicious users from creating a large number of sybils, every attempt to generate a peer identity is associated with some *Proof-of-Work* model [25] like computational puzzles [4], [24], [6], [11], [9]. In this model, a peer is forced to solve a puzzle before generating its identity. The puzzle result is embedded in the peer's Identity Certificate so that others can verify the same.

Any verifiable computational puzzle (with varying difficulty level) can be associated with the peer identity generation process. The puzzle's difficulty level is chosen such that a honest peer can easily generate its peer identity once whereas a malicious user incurs huge cost when it attempts to generate a large number of sybils.

In our model, we chose the puzzle in which a peer is required to find the preimage of its public key under a given hash function. Under this puzzle, the number of hash operations required to generate a peer identity is given by: $\Gamma = 2^n$ where 'n' is the number of bits in the hash output[2] [26]. This value also determines the difficulty level of the puzzle and can be set to some value depending on the application needs.

[2]The theoretical bound on a preimage search holds good since the input (i.e. the public key and hence the peer identity) cannot be modified arbitrarily.



(a) Transaction Model 1 (b) Transaction Model 2

1: Transaction threat models

D. Threat Model

1) Peer model: The system includes honest users as well as malicious users. Each honest user follows the protocol (i.e. rules that are to be abided by a peer while being part of it) and generates its peer identity once before joining the system. Each malicious user may generate multiple peer identities called sybils. Malicious users may collude with each other. Sybils, created by all malicious users, are considered to be under the control of a single *adversary*. They are byzantine in nature and attempt to subvert the system at all times by helping each other.

Since the size of the system is large, a honest peer may know a very small subset (maybe none) of the other peers in the system. However, sybils know all other sybils under the control of the *adversary*.

2) Transaction model: We consider two transaction models in this paper.

a) Model I: In this model (shown in Fig. 1(a)), the transaction involves two peers, an honest peer and the other unknown peer. The honest peer is at risk of being abused by the other peer during the transaction. Hence, the honest peer is considered as the *verifier* and the other unknown peer is considered as the *suspect*.

b) Model II: In this model (shown in Fig. 1(b)), the transaction involves three peers, an honest peer and two unknown peers. The two unknown peers transact among themselves. Through this transaction, they can cheat the third honest peer. Hence, the honest peer is considered as the *verifier* and the other two peers as *suspects*.

A peer can be a *verifier* or a *suspect* depending on the role it plays in a transaction. A *verifier* peer attempts to determine if the *suspect* peer(s) is a honest peer or a sybil.

An example would make the above mentioned transaction threat scenarios clear. Consider P2P reputation systems for file sharing applications in which sybils attempt to lower the content availability of the system. To accomplish this, sybils fake transactions among themselves to raise their reputation. Once they attain a high reputation, they serve low quality files to other honest request peers in the system. A new (honest) requester peer analyzes the past transactions of a provider peer before selecting it for a transaction (as per *model II*). Once a provider peer is chosen, it can still cheat the honest requester peer by sending a *decoy* file (as per *model I*).

We use the notations shown in the following table in the rest of this paper.

Symbol	Description
$\mathbb{N}$	Size of the P2P network
K_A^+	Public key of a peer A
K_A^-	Private key of a peer A
ID_A	Identity of a peer A
ID_A^*	Transient identity of a peer A
$\{ID_A, ID_B\}_\Phi$	Set of two peer identities that match by (prefix/suffix) Φ consecutive digits
M_{TR}	Timestamped message containing a list of peer identities and their digitally signed Ack messages
H	A cryptographic hash function whose output is uniformly distributed over the output space
$H(N_1) \rightarrow N_2$	N_1 is hashed using H to get N_2
N^x	Number whose 'x' leading/trailing consecutive bits are zero
R	Random number
T_n	Unique transaction reference number
$N_1 \cdot N_2$	Concatenation of N_1 and N_2 numbers

IV. SyMon

In this section, we describe a novel solution that enables all honest peers to protect themselves from sybils with high probability in large structured P2P systems. In our proposed system, we associate every transacting peer with another non-sybil peer known as *SyMon*. A given peer's *SyMon* is chosen dynamically such that the chances of both of them being sybils are very low. The chosen *SyMon* is entrusted with the responsibility of moderating the transactions involving the given peer which makes it almost impossible for sybils to compromise the system.

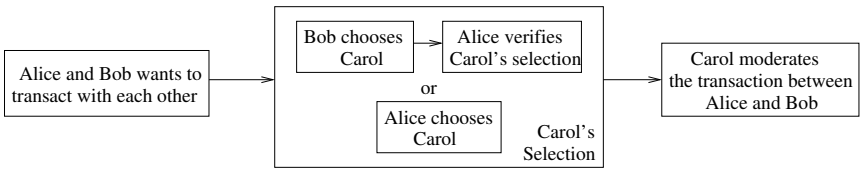
A. Challenges in choosing a SyMon

Choosing a *SyMon* for any given peer in a fully distributed P2P system prone to Sybil attack poses the following challenges:

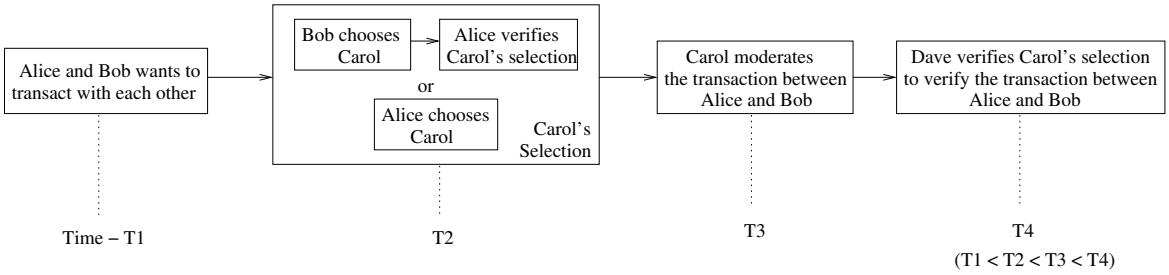
- The given peer and its *SyMon* should not be sybils. More specifically, a malicious user should incur a very high cost if it attempts to break this defense.
- Any peer should be able to verify with high probability whether the given peer and its *SyMon* are sybils.
- The selection process should be simple so that honest peers can choose their *SyMon* with minimal overhead.
- *SyMon* should be selected dynamically so that the same peer (or a small set of peers) is not chosen always.

B. SyMon Selection Overview

In this section, we propose four ways of *associating* any given peer with its *SyMon* in an attempt to meet the challenges listed in section IV-A. Our selection methods accomplish their main goal of choosing a *non-sybil SyMon* by exploiting the fact that it is very difficult for two sybils to be neighbors in the identity space when peer identities are generated randomly and when an *adversary* does not control a large percentage of sybils (see section III-B and section III-C). Each of these methods has varying complexity and incurs varying cost on honest peers as well as an *adversary*. Depending on the



(a) Transaction Model 1



(b) Transaction Model 2

2: Transaction threat models under SyMon

application needs, any one of our selection methods can be employed to defend against Sybil attack.

We describe a discovery protocol for dynamically choosing the most eligible *live* peer in the system as per our proposed *SyMon* selection criteria. This discovery protocol is fully decentralized and piggybacks on the underlying structured overlay routing constructs (outlined in section III-A). We also propose a verification mechanism to check if a *SyMon* was chosen as per one of our selection methods.

Consider a transaction involving two unknown peers, say *Alice* and *Bob* (see *Model I* in section III-D.2). *Alice*, a honest peer, is the *verifier* and *Bob* is the *suspect*. In our scheme, either *Alice* or *Bob* can choose *Bob's SyMon*, say *Carol*. If *Alice* chooses *Carol*, then our selection methods ensure with high probability that it is not a sybil of *Bob*. If *Bob* chooses *Carol*, then *Alice* can, with high probability, verify whether *Bob* and *Carol* are sybils. If *Alice* concludes that *Bob* has chosen its sybil as its *SyMon*, then the transaction is aborted.

A *non-sybil Carol* moderates the transaction between *Alice* and *Bob* to prevent *Bob* from cheating *Alice*. *Carol* also ensures the legitimacy of the transaction between *Alice* and *Bob* as shown in Fig. 2(a). For example, in P2P systems prone to DOS attack, *Carol*(puzzle server) can construct a puzzle challenge. *Alice*, the server, can decide to serve a client's (*Bob's*) request only after it solves the puzzle.

Any peer (say *Dave*) can determine the legitimacy of the transaction between *Alice* and *Bob* even when it was not directly involved in the transaction by i)verifying the selection of *Carol* (as depicted in Fig. 2(b)) and ii)considering *Carol's* feedback about the status of the transaction. For example, in P2P reputation systems, *Dave* (a new requester peer) analyzes the old transaction history of a potential provider peer (*Bob*) before selecting it. During this step, *Dave* can verify the selection of *Carol* which moderated the transaction between *Alice*(old requester peer) and *Bob* and consider *Carol's* feedback in order to assess the validity of *Bob's* past history as detailed in *Model II* of section III-D.2.

C. SyMon Selection Methods

In this section, we present different ways of choosing *SyMon* for a given peer. Each of our *SyMon* selection methods specifies the criteria in terms of the threshold minimal number of (prefix/suffix) consecutive digit match between peer identities. Only those peer identities which are very close to each other in the identity space will have maximum digit match. As per section III-B and section III-C, the computational cost associated with generating sybils that are closest neighbors in the identity space is prohibitively high. Since the chances of sybils being closest neighbors in the identity space are very low, our selection criteria enable us to choose a *non-sybil SyMon* for any given peer.

The minimal threshold value determines the cost incurred by an *adversary* in breaking each of our selection methods and is dependent on the size of the network. To compute its value, every peer should estimate the size of the network at periodic intervals by using any of the approaches proposed in [24], [27], [28], [29]. A new peer can obtain the network size estimates at different instances of time from some of its neighbors in the identity space. From this data, it can compute the average estimate at different times in the past. The chances of sybils corrupting this estimate is very low since a majority of the neighbors of any peer will be *non-sybil* peers.

1) Selection Method 1: Criteria: In this method, a peer whose identity matches by (prefix/suffix) Φ consecutive digits

with *Bob*'s identity is considered as *Bob's SyMon* (ID_S).

$$\{ID_S \mid \{ID_{Bob}, ID_S\}_\Phi \text{ and } \Phi \geq \Phi_\beta\} \text{ where } 0 \leq \Phi < 40$$

Here, Φ_β is a constant dependent on the size of the network. When it is set to an appropriate value, *Bob*'s closest neighbor(s) can satisfy this criteria.

Discovery: To discover Bob's closest *live* neighbor, a *TraceRoute* message(M_{TR}) is sent to *Bob*'s neighbors (see section II-A.2 and section III-A). Any neighbor peer which receives M_{TR}, appends its digitally signed Ack message to it so as to prevent sybils from corrupting or creating bogus M_{TR} messages. *Bob's SyMon* is finally chosen from M_{TR}, as per the criteria.

Discussion: In this method, either *Alice* or *Bob* can discover *Bob's SyMon* with minimal effort. Since a different *SyMon* is chosen for every peer, the transaction monitoring load is well balanced. However, if the neighborhood of a peer does not change, then the same peer is burdened with monitoring all transactions involving the given peer.

The main disadvantage with this approach is that if an *adversary* finds two sybils who are also neighbors, then they can be *reused* for multiple transactions with other honest peers.

2) Selection Method 2: Criteria: In this method, any two peers whose identities match by (prefix/suffix) Φ consecutive digits are considered as *Bob's SyMons* (ID_{S_1}, ID_{S_2}).

$$\{ID_{S_1} \text{and } ID_{S_2} \mid \{ID_{S_1}, ID_{S_2}\}_\Phi \text{ and } \Phi \geq \Phi_\alpha\} \text{ where } 0 \leq \Phi < 40$$

Here, Φ_α is a constant dependent on the size of the network.

Discovery: Every peer finds its closest neighbor, as mentioned in selection method 1 (see section IV-C.1). If the locally discovered peer identity set comprising of the peer's and its neighbor's identities satisfies the above mentioned criteria then it is advertised in the system. The advertisement of peer identity sets (along with M_{TR}) of some Φ value is similar to the advertisement of files(with associated metadata) in P2P file sharing systems. The Φ value can be considered as the name of the advertised set. When required, either *Alice* or *Bob* can query the system for an advertised set by specifying Φ as the search keyword. If multiple sets are returned, a set is chosen randomly. Both the peers found in the set are chosen as *Bob's SyMons* since the chances of two sybils being neighbors and hence being found in the same advertised set are very low.

Discussion: The discovery protocol in this method is complex and sensitive to churn. In systems with high churn rate, the process has to be repeated periodically to ensure the freshness of the advertised peer identity sets. Peers which store the advertised copies of the discovered sets can discard the sets after *some* time period depending on the churn rate.

In this method, there is a possibility that the same advertised peer identity set(s) is employed by all transacting peers as their *SyMon*. Hence, the peers found in this set have to incur the entire load of monitoring all the transactions in the system. The load can be balanced more evenly by setting Φ_α to an appropriate value and hence increasing the number of advertised sets as discussed in section V-A.

3) Selection Method 3: Criteria: In this method, a *transient identity* is generated for *Bob* for each of its transactions. A peer whose identity matches by (prefix/suffix) Φ consecutive digits with *Bob's transient identity* is considered as its *SyMon* (ID_S).

$$\{N^x \mid H_1(K^+_{Bob} \cdot K^+_{Alice} \cdot T_n \cdot R) \rightarrow N^x\}$$
$$\{ID^*_{Bob} \mid H_2(K^+_{Bob} \cdot K^+_{Alice} \cdot T_n \cdot R) \rightarrow ID^*_{Bob}\}$$
$$\{ID_S \mid \{ID^*_{Bob}, ID_S\}_\Phi \text{ and } \Phi \geq \Phi_\beta\} \text{ where } 0 \leq \Phi < 40$$

Bob can generate its *transient identity* by solving a computational puzzle. Though any verifiable computational puzzle can be adopted during this step, we have used the one mentioned in [24]. To solve the puzzle, *Bob* has to find a random number(R) such that the output of the hash function(H_1) contains 'x' leading/trailing bits as zero. *Bob*'s public key (K^+_{Bob}), *Alice*'s public key(K^+_{Alice}) and a unique number identifying the transaction(T_n) are given as parameters to the puzzle. The puzzle's result(R) and its parameters are hashed (H_2 - SHA-1) to generate *Bob*'s *transient identity*.

Discovery: The discovery protocol, in this method, involves finding a *live* peer closest to *Bob's transient identity* in the system. This is done by sending a message with *Bob's transient identity* as the destination address (as *Lookup(Bob's transient identity)*). *Bob's SyMon* is then chosen from among the neighbors of this discovered peer as detailed in selection method 1 (refer section IV-C.1).

Discussion: This selection process is applicable where *Bob* is required to choose its *SyMon* rather than *Alice*, so that the burden of solving the puzzle is on *Bob*. The unique transaction reference numbers ensure that the *transient identities* are uniformly distributed over the identity space. This results in the selection of a different *SyMon* for every transaction thereby distributing the monitoring load uniformly on all peers in the system.

The above mentioned puzzle parameters prevent the puzzle result from being *reused* by *Bob* (or *Bob*'s sybils) for a different transaction with *Alice* or with any other peer. The main drawback with this approach is that even an honest peer is required to solve the puzzle for every transaction. Moreover, a sybil can *precompute* the puzzle associated with its *transient identity* generation.

4) Selection Method 4: This method is similar to selection method 3, discussed in section IV-C.3. However, in this method, *Bob's transient identity* is associated with a validity period by adding *Nonce*, a random number, as one of the parameters to the puzzle.

$$\{N^x \mid H_1(K^+_{Bob} \cdot K^+_{Alice} \cdot T_n \cdot Nonce \cdot R) \rightarrow N^x\}$$
$$\{ID^*_{Bob} \mid H_2(K^+_{Bob} \cdot K^+_{Alice} \cdot T_n \cdot Nonce \cdot R) \rightarrow ID^*_{Bob}\}$$
$$\{ID_S \mid \{ID^*_{Bob}, ID_S\}_\Phi \text{ and } \Phi \geq \Phi_\beta\} \text{ where } 0 \leq \Phi < 40$$

Nonce can be specified by either *Alice* or a generic entity outside the system as mentioned in [30]. This ensures that sybils can't *precompute* their *transient identities*. However, the main drawback with this approach is the difficulty in determining the duration of the validity period. If this duration

is small, some honest peers (with limited resources) may have to solve the puzzle repeatedly. On the other hand, larger duration results in a more easily subvertable defense against sybils. The duration of the validity period should therefore depend on the application needs.

D. Verification of SyMon's selection

Any peer can check whether the *SyMon* of a given peer was *indeed* selected according to one of our selection methods as shown below:

1) Get an estimate of the size of the network at a time that is closest to the time when the peer identity set and the discovery protocol's *TraceRoute* message were created.
2) Determine Φ_α/Φ_β and check if the Φ value of the peer identity set meets the selection criteria that was adopted to choose the *SyMon*.
3) Verify if all the peers found in the discovery protocol's *TraceRoute* message have provided their digitally signed Ack messages.
4) If a puzzle is posed, check if the puzzle parameters and its result are valid. (This is applicable only to selection methods 3 and 4).
5) If some validity period is associated with the peer identity set, check if it has expired. (This is applicable only to selection method 4).

Verification fails if any of the above mentioned tests specified in step 3 to 5 fails.

V. Theoretical Evaluation of SyMon

In this section, we first discuss the formal model underlying our sybil defense scheme. Using this model, we determine an appropriate value for Φ_α and Φ_β that form an integral part of our *SyMon* selection methods. We discuss how these threshold values influence the cost incurred by an *adversary* to break our system. We then consider some of the attack strategies that can be adopted by the *adversary* to compromise our system. We also explore theoretically, the security margin provided by each of our selection methods under these attack strategies.

A. Choosing SyMon

The formal model underlying our solution draws parallels between the problem of finding a *SyMon* for any given peer and the *classical birthday problem* [31] and its family of problems. The *classical birthday problem* refers to the probability of finding *some* pair of people having the same birthday in a set of randomly chosen people. For example, collision search [32] on a hash function (like SHA-1, used to generate peer identities) is similar to the *classical birthday problem*.

In our model, we consider every peer identity to be the result of one hash operation(trial) on SHA-1. Therefore, the entire network represents the total number of trials(r) performed by all peers collectively on SHA-1. Using this model, we discuss the selection of *SyMon* in selection method 2 followed by a discussion on the rest of the selection methods.

Selection method 2: In this method, we try to discover *any* two peers whose identities match by at least Φ_α digits. The search for a *single* advertised set in the system containing these eligible peers is equivalent to performing a *partial collision search* on SHA-1. This is because only Φ digits collide between the peer identities where $0 \leq \Phi < 40$ and they can be found within some segment of the identity space. It is important to note that a *partial collision search* on a hash function (like SHA-1) is similar to the *almost birthday problem* [31] that refers to the probability of finding *some* pair of people having birthdays within a few days of each other in a set of randomly chosen people.

A *partial collision search* on SHA-1 can also be considered as a *collision search* on a hash function(H′), a version of SHA-1 with (reduced) unknown number of bits(x) in its hash output. In other words, the problem of determining the value of Φ_α, given the total number of trials(r), reduces to finding the number of bits(x) of H′ that results in a *collision*.

From [32], we can determine an approximate value of Φ_α as shown below:

$$r = \sqrt{2^x} = \sqrt{2^{4\Phi_\alpha}} \tag{1}$$

Here, the theoretical bound on *collision search* holds good since the input(peer identity) cannot be modified arbitrarily.

Selection methods 1, 3 and 4: In these methods, a peer whose identity matches with the given peer's identity (or its *transient identity*) by at least Φ_β digits is chosen as its *SyMon*. The search for an eligible peer in the system is equivalent to performing a *partial preimage search* on SHA-1. A *partial preimage search* on a hash function is similar to the *almost same birthday as you* problem [31]. This problem refers to the probability of finding another person whose birthday is within a few days of the given person's birthday in a set of randomly chosen people.

A *partial preimage search* on SHA-1 can also be viewed as a *preimage search* [26] on H′ as mentioned above. Here, a *preimage search* on a hash function is similar to the *same birthday as you* problem which refers to the probability of finding another person whose birthday is the same as the given person's birthday in a set of randomly chosen people.

If we want to choose a *SyMon* for a *single suspect* peer, we can determine an approximate value of Φ_β using the Equation from [26]:

$$r = 2^x = 2^{4\Phi_\beta} \tag{2}$$

Here, the theoretical bound on *preimage search* holds good since the input(peer identity) cannot be modified arbitrarily.

It is important to note that a *partial preimage search* on SHA-1 or a *preimage search* on H′ refers to finding a *SyMon* for a single *suspect* peer. However, we want every *suspect* peer in the system to find its *SyMon*. Our stronger condition of finding a *SyMon* for *every* peer is satisfied by considering the search for *SyMon* as a *strong preimage search* [31] on H′ or *strong partial preimage search* on SHA-1. A *Strong preimage search* on a hash function is similar to the *strong birthday problem* which refers to the probability that everyone, in a set of randomly chosen people, finds another person with the

same birthday. Thus, the problem of determining the value of Φ_β, given the total number of trials(r), reduces to finding the number of bits(x) of H$'$ that results in a successful *strong preimage search*.

From [31], we can determine an approximate value of Φ_β as shown below:

$$\frac{r'_1}{m} = log\left(\frac{m}{log\frac{1}{p}}\right)$$
$$\frac{r'_i}{m} = \frac{r'_1}{m} + log\left(\frac{r'_{i-1}}{m}\right) \text{ where } m = 2^{4\Phi_\beta} \quad (3)$$

where 'p' refers to the probability with which we want to find a *SyMon* for *all* peers. In this Equation, r' is an iterative approximation of the number of trials required. Φ_β can be determined for a value of r' that is closest to the given network size(r).

Load balancing in Selection Method 2: In this method, the search for a *single* advertised set through a *partial collision search* leads to highly skewed transaction monitoring load distribution. The load can be balanced well by controlling the number of advertised sets in the system. The number of advertised sets depends on Φ_α which can be determined by setting the value of 'k'- number of peers without a *SyMon*, to some fraction of the total size of the network in the following Equation from [31]:

$$p_k = \sum_{i=k}^{r} (-1)^{i-k} \frac{i!}{k!\,(i-k)!} \quad \frac{m!\,r!\,(m-i)^{r-i}}{i!\,(m-i)!\,(r-i)!\,m^r} \quad (4)$$

where p_k refers to the probability with which '$\mathbb{N}-k$' peers find their *SyMon* and r refers to the network size. Here, Φ_α is determined from the relation: $m = 2^{4\Phi_\alpha}$.

B. Attack Strategies of an Adversary

In our system, an *adversary* needs two sybils to cheat an honest peer; one to transact with the honest peer and the other one to act as *SyMon*. More importantly, these two sybils should possess identities that satisfy one of the selection criteria. In transactions involving three peers, the *adversary* needs to ensure that any two of its three sybils satisfy the identity matching criteria.

In this section, we discuss some of the strategies that can potentially be employed by the *adversary* to get this desired sybil pair.

Break RSA: An *adversary* can try to find the private keys of two honest peers who possess desired peer identities. However, as per [20], factoring 1024-bit RSA is still considered infeasible. It is almost impossible for the *adversary* to *steal* the private keys of honest peers through *Side channel attacks* [33] since peers in large P2P systems are generally widely dispersed across different geographical locations.

Partial Collision Search Attack on SHA-1: An *adversary* can launch *partial collision search attack(PCSA)* to find a pair of desired sybils as discussed in section V-A. This is done by randomly generating peer identities repeatedly until a desired pair is obtained. The expected number of trials required to find a pair of desired sybils is given by: r_{pcs} where r_{pcs} refers to the size of the system in Equation (1).

Partial Preimage Search Attack on SHA-1: An *adversary* can launch *partial preimage search attack(PPSA)* to find a sybil such that its identity matches with its existing sybil by Φ digits (see section V-A). This is done by randomly generating peer identities repeatedly until a desired sybil is found. The expected number of trials required to find a desired sybil(preimage) for an existing sybil is given by: r_{pps} where r_{pps} refers to the size of the system in Equation (2).

Precomputation and Replay Attack: An *adversary* can attempt to *precompute* the puzzle associated with generating a peer identity or its *transient identity*. Once the desired sybil pair is obtained, the *adversary* can also attempt to *reuse* them for future transactions with other honest peers.

I: SyMon selection methods under different attack strategies

Method	Breaking RSA	PCSA	PPSA	Precom putation	Replay
1	N	Y	Y	Y	Y
2	N	Y	Y	Y	Y
3	N	N	Y	Y	N
4	N	N	Y	N	N

II: Security margin offered by SyMon selection methods

Network Size	Method	Φ_α/Φ_β	Estimation of the number of sybils required to subvert the system	Attack Strategy
10^3	1	$1-2$	2^2 to 2^4	PCSA
	2	$4-5$	10^3	
	3, 4	$1-2$	2^4 to 2^8	PPSA
10^5	1	$3-4$	2^6 to 2^8	PCSA
	2	$8-9$	10^5	
	3, 4	$3-4$	2^{12} to 2^{16}	PPSA
10^7	1	$4-5$	2^8 to 2^{10}	PCSA
	2	$11-12$	10^7	
	3, 4	$4-5$	2^{16} to 2^{20}	PPSA

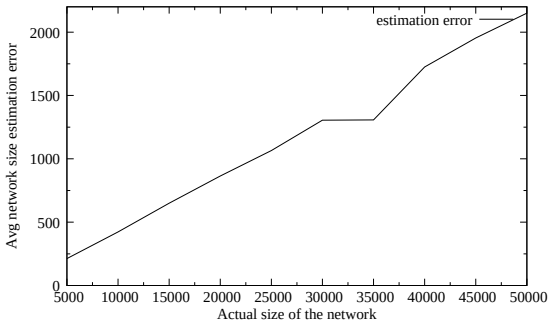
C. SyMon selection methods under attack

In this section, we show the security margin provided by each of our *SyMon* selection methods under different attack strategies of an *adversary*.

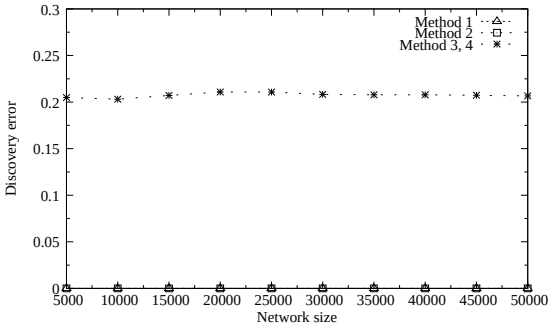
In method 1, an honest peer performs a *strong partial preimage search* to find *SyMon* through the discovery process. However, an *adversary* can launch *partial collision search attack* to find a pair of desired sybils. Hence, the security margin provided by this method is very less.

In method 2, an honest peer performs a *partial collision search* to find *SyMon* through the discovery process. An *adversary* should also launch *partial collision search attack* to find a pair of desired sybils. However, the security margin offered by this method is inversely proportional to the number of advertised sets (see section V-A).

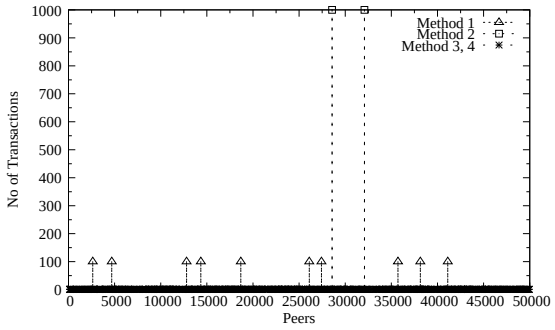
In methods 3 and 4, a *SyMon* is chosen based on the *transient identity* of the given peer. An honest peer performs



(a) RMSE between the actual network size and the average estimated network size



(b) RMSE between Φ_d and Φ_a as per different selection methods



(c) Load distribution on peers chosen as *SyMon* as per different selection methods

3: Effectiveness of the SyMon discovery protocol

a *strong partial preimage search* to find a *SyMon* through the discovery protocol. An *adversary* can either generate multiple *transient identities* for one of its sybils or generate multiple sybils for one of its sybils' *transient identity*. It can also adopt both these approaches to find a desired sybil. In either of these approaches, the *adversary* has to launch *partial preimage search attack* on SHA-1 to find a desired sybil identity.

Methods 3 and 4 ensure that even if a pair of desired sybils is obtained, it is not possible to *reuse* them for future transactions. Whereas, in methods 1 and 2, once the desired sybil pair is obtained, it can be *reused* for future transactions with other (honest) peers.

In methods 1, 2 and 3, it is possible to *precompute* the desired sybil pair. Method 4 prevents this by associating a validity period with a peer's *transient identity*. The validity period, when set appropriately, ensures that the cost, associated with solving the puzzle, is a recurring one for an *adversary*.

Table I and Table II compare each of our proposed *SyMon* selection methods under different attack strategies of an *adversary*. Table II also shows the approximate number of sybils required to subvert the system. The actual cost incurred by the *adversary*, in breaking our defense, is the product of the required number of sybils and the cost associated with generating each sybil (see section III-C). Thus, the security margin offered by each method is directly proportional to the size of the network and inversely proportional to the fraction of sybils present in the system. Hence, it is important to contain the total number of sybils in the system.

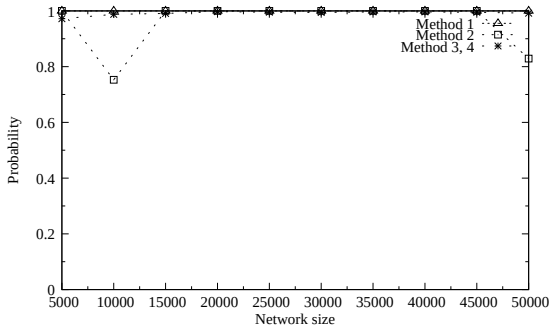
Man in the middle attack: During the *SyMon* selection phase, sybils can employ *man in the middle* attack strategy to enable their selection by honest peers. However, only if they satisfy the selection criteria, sybils can be chosen by honest peers as their *SyMon*. Our assumed robust (application dependent) transaction monitoring process prevents sybils from harming honest peers thereby rendering this approach unattractive.

Sybils can still adopt this attack strategy to disrupt message exchanges between other honest peers in the system. It is important to note that we consider the underlying overlay infrastructure to be outside the control of sybils (see section III-A). Hence, issues arising out of communication failures are considered to be outside the scope of our work.

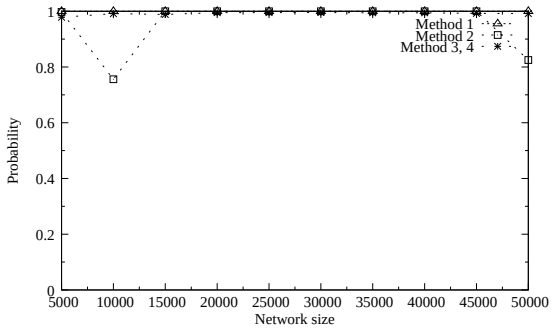
VI. Experimental Evaluation of SyMon

In this section, we assess the performance of our proposed sybil defense scheme through simulation results.

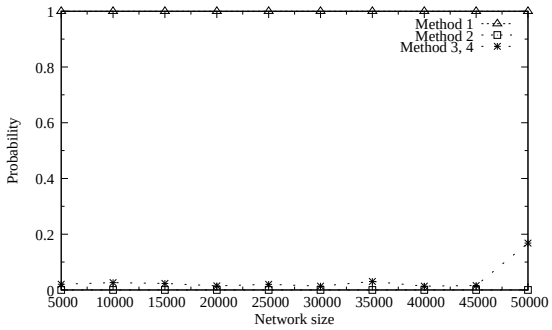
In our experiments, we consider a Pastry overlay of size varied between 5,000 and 50,000, simulated through FreePastry [34]. All our experiments were run on a Intel-P4 Quad Core



(a) Probability of considering honest *suspects* as honest in the absence of sybils



(b) Probability of considering honest *suspects* as honest when the fraction of sybils is 0.25



(c) Probability of considering sybil *suspects* as honest when the fraction of sybils is 0.25

4: Effectiveness of SyMon as a sybil defense system at different network sizes

system with 4GB memory and the results were averaged over five trials. In our experiments related to selection methods 1, 3 and 4, we set Φ_β to a value so that the probability of every peer finding its *SyMon* is 0.99 (see Equation (3)).

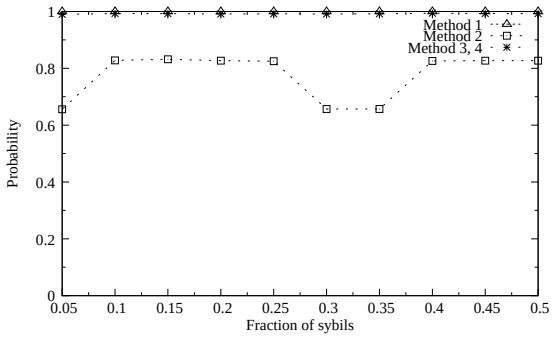
Aim: Our first goal is to check the effectiveness of the *SyMon* discovery protocol in discovering the most eligible *live* peer in the system as *SyMon*. We also show its efficiency in distributing the transaction monitoring load on peers chosen as *SyMon*. Our second goal is to study the effectiveness of our proposed sybil defense scheme. More specifically, we show i) the probability of success in considering honest peers as honest and ii)the probability of success in considering sybils as sybils by a randomly chosen honest *verifier* peer. We explore these results under various network sizes to study the scalability of our system.

A. SyMon Discovery Protocol

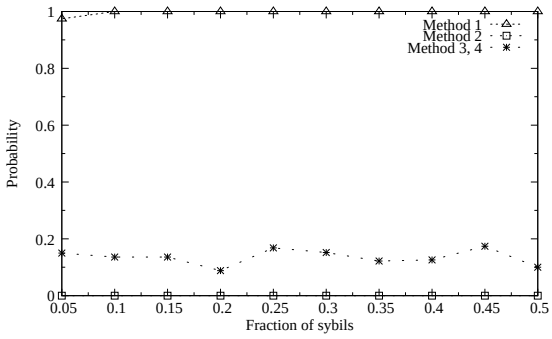
Network Size Estimation: Our proposed solution requires each peer to estimate the size of the network to set an appropriate value for Φ_α and Φ_β and hence verify the *SyMon* selection. Since the accuracy of this estimation influences our solution's accuracy, we measure the effectiveness of the network size estimation algorithm.

In this experiment, the size of the network was varied from 5,000 to 50,000. At different network sizes, every peer estimated the size of the network by checking the distance between the identities of each of its *LeafSet* peers, as proposed in [27]. Since we consider a static network, every peer estimated the network size when it joined the system. The root mean square error(RMSE) between the actual network size and the average size estimated by all peers is shown in Fig. 3(a). We observe that the error in estimation is small and does not vary much with an increase in the network size.

SyMon Selection methods: In this set of experiments, the size of the network was varied from 5,000 to 50,000. At different network sizes, every peer *discovered* its *SyMon* as per each of our selection methods (refer section IV-C). In each case, the number of digits(Φ_d) that matched between the given peer's identity (or its *transient identity*) and the discovered *SyMon's* identity was computed. We then handpicked the most eligible peer(s) in the system as per each of our selection methods. The number of digits(Φ_a) that matched between the given peer's identity (or its *transient identity*) and the handpicked peer's identity was computed. The RMSE between Φ_d and Φ_a was averaged. From Fig. 3(b), we observe that method 1 always finds the most eligible *SyMon*. Method 2 does not always pick the most eligible *SyMon* but it performs better than method 3 and 4.



(a) Probability of considering honest *suspects* as honest when the network size is 50k



(b) Probability of considering sybil *suspects* as honest when the network size is 50k

5: Effectiveness of SyMon as a sybil defense system at different fractions of sybils

Load Distribution on SyMon peers: In this experiment, the number of peers in the system was 50,000. Among them, 10 peers were chosen to represent a set of transacting peers in a typical P2P system. Each of the randomly chosen 10 peers performed 100 transactions with other randomly chosen peers in the system of size 50,000. For each transaction, the *SyMon* was *discovered* as per our protocol. At the end of 1000 transactions, the transaction monitoring load distribution was aggregated. Refer Fig. 3(c). In selection method 1, it is distributed over 10 peers since each transacting peer had a different *SyMon*. In selection method 2, the load is levied on only two peers since they were the only eligible *SyMon* pair in the system. In selection method 3 and 4, *SyMon* is chosen based on the randomly generated *transient identity* of a peer. Hence, the load is uniformly distributed on all peers in the system and is represented as a thick line at the bottom of the graph.

In our experiments, selection method 2 suffers from highly skewed load distribution since we have considered the best case of choosing only two peers (a single *SyMon* peer set) through a *partial collision search*. This peer set was chosen to monitor all the transactions in the system. The load can be balanced more evenly as discussed in section V-A.

In this experiment, the number of transacting peers was not varied as it does not vary the load distribution in the system.

B. SyMon Defense scheme

In this set of experiments, we study the effectiveness of our sybil defense scheme against Sybil attack.

In our experiments involving selection method 3 and 4, a *suspect* peer generated its *transient identity* randomly to mimic the puzzle computation process. Both honest peers and sybils perform this operation once in each of these experiments. In section V, we have already shown the expected number of attempts required for an *adversary* to break this defense.

Each experiment involves 1000 transactions between a randomly chosen honest *verifier* and a randomly chosen *suspect* peer. An honest *suspect* peer always chose its *SyMon* as per the discovery protocol whereas a sybil *suspect* peer chose one of its sybils, that best suited the selection criteria, as its *SyMon*. The *verifier* peer decided whether the given *suspect* and its *SyMon* were honest peers or sybils as per our verification strategy (see section IV-D). We then compared its decision against the actual type of peer chosen as the *suspect* peer in order to determine the false positives.

Before we analyze the experimental results, let us recall the significance of the accuracy of network size estimation on the correctness of the decisions taken by *verifier* peers. When the estimated network size is very high (or low), a *verifier* peer sets its Φ_α/Φ_β to a very high (or low) value. When Φ_α/Φ_β is set to a very high value, both honest as well as sybil *suspect* peers fail to satisfy the selection criteria and hence would be branded as sybils. On the other hand, when Φ_α/Φ_β is set to a very low value, both honest and sybil *suspect* peers can satisfy the selection criteria and hence, they would be considered as honest peers. Suppose, the network size estimated by the *verifier* peer is accurate, then the number of false positives in its decisions depends on the number of sybils in the system. More specifically, as long as the size of the network (*SyMon* search space as per the protocol) is significantly greater than the total number of sybils (sybil *SyMon* search space), our solution can detect sybils with greater accuracy.

Effectiveness in the absence of sybils: In these experiments, we considered a system free of sybils and evaluated the effectiveness of our scheme at different network sizes between 5,000 and 50,000. When only honest peers were chosen as *suspect*, each of our selection methods ensures that they were correctly identified as honest in most of the cases. Refer Fig. 4(a). This also implies that the chosen *verifier* peers were successful in estimating the network size accurately.

Effectiveness in the presence of sybils: In these experiments, we considered a network of size 50,000. Peers were randomly tagged as either honest or sybil such that sybils constituted some fraction of the system. The fraction of sybils was varied between 0.05 and 0.5. When only honest peers were chosen as *suspect*, our selection methods 1, 3 and 4 ensure that they

were correctly identified as honest in almost all transactions. Though selection method 2 does not perform as well as the other methods, it ensures that honest peers are considered as honest with high probability (see Fig. 5(a)). This implies that selection method 2 is more sensitive to the the accuracy of network size estimation as compared to the other methods. This is due to the fact that only the most eligible *SyMon* pair was considered in our experiment.

When sybils were chosen as the *suspect* peers, our selection method 1 was unable to detect them since an *adversary* can break this defense easily. Selection methods 2, 3 and 4 identify sybils with very high probability (see Fig. 5(b)). These results corroborate our theoretical analysis. Even when the fraction of sybils in the system increases, there is very little variation in the performance of our selection methods. This is due to the fact that sybils choose only among themselves while honest peers choose *SyMon* as per the protocol from the entire system. Hence, honest *verifier* peers set their Φ_α/Φ_β to values corresponding to the total size of the network. Sybils fail to choose among themselves that can satisfy the selection criteria due to their search space being significantly smaller than the size of the network. For example, in Fig. 5(b), we observe that the maximum number of sybils is 25,000 while the size of the network is maintained at 50,000. This ensures that our solution identifies sybils accurately.

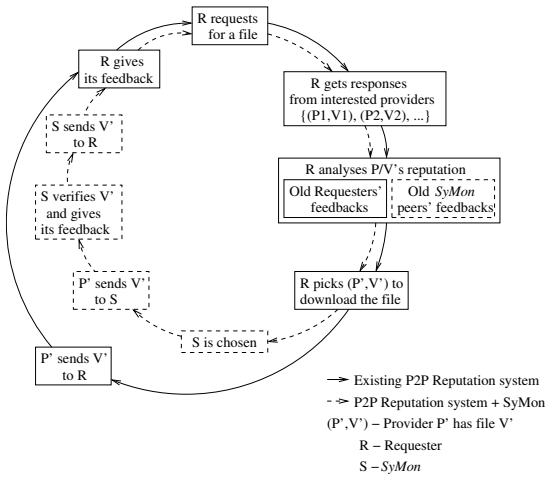
We repeated the above two experiments by varying the network size from 5,000 to 50,000 when the fraction of sybils was set to 0.25. Fig. 4(b) and Fig. 4(c) show that the effectiveness of our solution does not vary much with the variations in the size of the network.

VII. Case Studies on SyMon

SyMon, our sybil defense scheme proposed in section IV is a generic approach that can be integrated with a variety of P2P applications. In this section, we explore the application of *SyMon* in two such applications in order to analyze its robustness against Sybil attack in practice.

A. P2P reputation systems

Since its inception, P2P file sharing systems have been known to be a powerful medium for distribution of files. The growing popularity of these systems has attracted the unwanted attention of malicious users who try to spread their infected files in the system. In order to achieve this, they tamper with the contents of a popular file rendering it useless while still retaining the metadata of the original file. The files could even be infected with Viruses, Trojans, Worms, etc so as to spread *malware* in the system [35]. Such deliberately *poisoned* files, also known as *decoy files*, are injected into the system in massive numbers thereby decreasing its content availability [36], [37]. Measurement studies conducted on today's popular P2P file sharing applications confirm this disturbing trend. As per [36], 80% of popular files in Kazaa are polluted. This necessitates the need for a mechanism by which an innocent peer can differentiate between *decoy* files and good files *before* the file is actually downloaded.

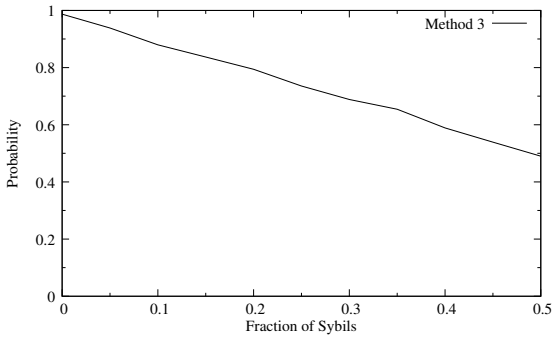


6: A typical transaction in an existing P2P reputation system and our new P2P reputation system for file sharing systems

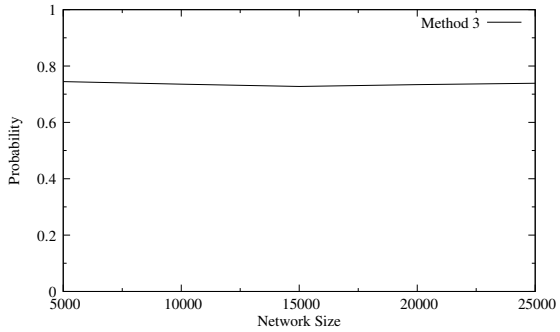
Recent research efforts have suggested reputation systems as a means for fighting deliberate content pollution and poisoning [38], [39], [40]. These systems aim to help a peer in making an informed decision about the nature of the transaction *before* actually downloading the file from other peers.

In a P2P reputation system, *Sybil Attack* is the most difficult and challenging problem to handle as it can be employed to launch all other types of security attacks. Sybils indulge in *ballot-stuffing* by spoofing transactions among themselves and giving good ratings thereby raising their reputation value. In the absence of centralized transaction monitoring systems, sybils can fake a massive number of transactions by logging transaction feedbacks. They need not even send a file to spoof a transaction. Once they attain a high reputation value, they can start *milking* their reputation to serve highly distorted/infected files to other innocent peers in the system. This necessitates the need for a mechanism by which sybils are prevented from artificially boosting their reputation through fake transactions. One solution is to get every file transfer transaction verified by a *non-sybil* peer, while it takes place, in order to differentiate between genuine and spurious transactions.

a) Our approach: We consider a typical P2P reputation system for a typical P2P based file sharing application. In such a system, we propose to employ a dynamically chosen *SyMon* to moderate every file sharing transaction so that sybils are discouraged from spoofing them (see Fig. 6). More specifically, this verification is performed while the transaction takes place, in order to differentiate between genuine and spurious transactions. As part of the verification process, *SyMon* checks whether the file being transferred from the provider peer to the requester peer is of good quality either manually or through automated methods [36], [41] and provide its feedback. Based on the old *SyMon* peers' feedbacks as well as the old requester peers' feedbacks, new requester peers can make an informed decision regarding the genuineness of past transactions while

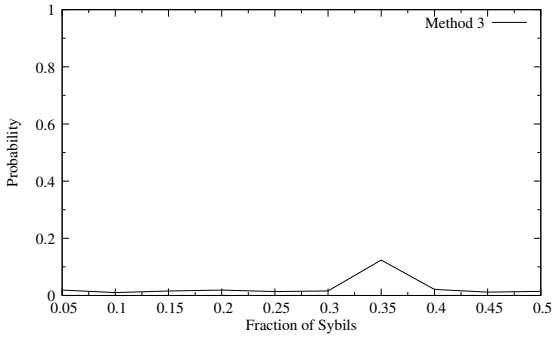


(a) Probability of considering good transactions as *legitimate* when the network size is 10k and the fraction of sybils is varied

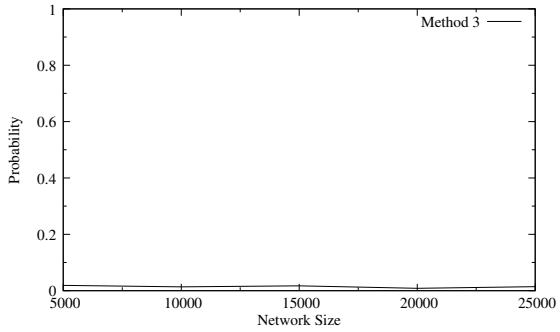


(b) Probability of considering good transactions as *legitimate* when the fraction of sybils is 0.25 and the network size is varied

7: Effectiveness of transaction legitimacy verification process against sybils when honest peers were chosen as *suspects*



(a) Probability of considering bad transactions as *legitimate* when the network size is 10k and the fraction of sybils is varied



(b) Probability of considering bad transactions as *legitimate* when the fraction of sybils is 0.25 and the network size is varied

8: Effectiveness of transaction legitimacy verification process against sybils when sybils were chosen as *suspects*

selecting reputed provider peers[3].

The transaction verification done by the chosen *SyMon* is similar to the verification done by the requester peer. However, our selection methods ensure that the chosen *SyMon* peers are non-sybil peers (with high probability). Thus, *SyMon* can be employed to help honest peers in identifying fake transactions and hence discourage sybils from lowering the content availability of the system through *ballot-stuffing*.

b) Evaluation: Our P2P reputation system (integrated with *SyMon*) can succeed only when a non-sybil peer is chosen as *SyMon* and the chosen *SyMon* peer can accurately monitor the transaction in the system. In section V, we have theoretically shown the robustness of our selection methods under various attack strategies of an *adversary*. Since *SyMon* is generic in nature and applicable to a wide variety of P2P applications, we had assumed that the transaction monitoring process is application dependent and robust to prevent sybils from exploiting it (refer section I). Hence , the simulation experiments discussed in VI focused on the impact of sybil attack on *SyMon* selection methods alone, given a robust transaction monitoring technique.

In this section, we relax our earlier assumption that the transaction monitoring process is resistant to sybils. Hence, the main goal of our current set of experiments (conducted on a testbed similar to the one described in section V) is to show the impact of Sybil attack on our reputation system, given a transaction monitoring technique that can potentially be compromised by sybils. More specifically, we consider the significance of incorrect feedbacks from sybils on the effectiveness of the entire system. In each of these experiments, unlike honest peers, sybils claim to have shared popular files while sending junk files among themselves. We aim to study the effectiveness of transaction monitoring process in exposing such fake transactions.

In the following experiments, the size of the network was

[3]The underlying P2P reputation system is assumed to i)manage the storage as well as the retrieval of the feedbacks from old *SyMon* peers as well as from old requester peers, for all transactions and ii)guide new requester peers in choosing reputed provider peers, based on these feedbacks.

maintained at 10,000 while the fraction of sybils was varied between 0.05 to 0.5. 1000 honest peers were randomly chosen as requester peers. Each of these requester peers randomly chose one *suspect* provider peer to perform a *single* transaction. In our experiments, the provider peers were chosen randomly since we have assumed that the underlying P2P reputation would guide requester peers in choosing reputed provider peers. If the chosen *suspect* provider peer is honest, then it selected *SyMon* as per the protocol and then sent a good file to the given requester peer via the chosen *SyMon*. If the chosen *suspect* provider is a sybil, then it chose another sybil as *SyMon* that best suited the selection criteria and then sent a junk file, via the chosen *SyMon*. In our experiments, we have employed selection method 3 to choose *SyMon* since it outperforms other methods[4]. A honest *SyMon* considered the transaction to be legitimate only when the file was of good quality and gave its feedback accordingly. On the other hand, a sybil *SyMon* gave incorrect feedback.

Once all the (1000) file transfer transactions were complete, 1000 honest *verifier* peers (i.e. new requester peers) were randomly chosen to determine the number of *legitimate* transactions. Each of these chosen *verifier* peers computed the number of legitimate transactions verified by a *SyMon* (whose selection is valid) and with positive feedback. This was compared against the actual number of good transactions to determine the total number of correct decisions[5].

When only honest peers were chosen as *suspect* provider peers by each of the 1000 requester peers, the effectiveness of the transaction legitimacy verification process suffered with the increase in the number of sybils in the system as shown in Fig. 7(a). This is because, an increase in the number of sybils increased the probability of the selection of sybils as *SyMon* to moderate good transactions. In addition to this, sybils can choose *SyMon* among themselves that can satisfy the selection criteria giving rise to a large number of incorrect decisions.

The same experiment was repeated by selecting only sybils as the *suspect* provider peers. From Fig. 8(a), we observe that the probability of considering bad transactions as *legitimate* is very low. This is because the selection of a sybil as *SyMon* can be easily identified by verifying its selection. This result is in sync with our earlier results shown in section VI-B.

The above two experiments were repeated at different network sizes ranging from 5,000 to 25,000 while controlling the fraction of sybils at 0.25. From Fig. 7(b) and Fig. 8(b), we observe that the process is scalable.

From these experiments, we conclude that the transaction verification, by the chosen *SyMon*, can help in identifying legitimate transactions in the system. This discourages sybils from lowering the content availability of the system through fake transactions.

[4]*SyMon* selection method 4 is very similar to method 3 except for the inclusion of *nonce* as an additional puzzle parameter. Hence, it is equally employable in these experiments.

[5]A decision is correct when a good transaction is also considered as *legitimate*.

B. DOS Attack

In P2P systems where peers offer services to others, sybils can launch DOS Attack by flooding the server with a large number of requests. In such a situation, the server can fail to respond to legitimate requests either temporarily or permanently depending on the severity of the attack.

Computational puzzles are widely adopted to discourage (sybil) clients from sending a large number of requests [6], [7]. In existing solutions, the responsiblity of forming the puzzle challenge rests with either the server itself or a central puzzle server (*bastion* [8]) or a randomly chosen server [9]. Each of these techniques can be broken by sybils with minimal efforts.

SyMon can help in preventing a DOS attack on a server through computational puzzles. In our approach, the process of forming the puzzle challenge is *outsourced* to the given client peer's *SyMon*. As each client peer is forced to solve the puzzle generated by a different *SyMon* peer, the puzzle generation load is distributed. Thus, our approach does not suffer from the inadvertent DOS attack problem that plague other existing solutions.

VIII. Related Work

Many of the existing P2P systems are prone to Sybil attack. As per [4], only a centralized peer identity generation scheme can prevent this attack completely. Distributed solutions either aim to prevent the generation of sybils by controlling the peer identities generated [6], [24], [10], [11] or detect their presence and hence protect the system.

Recently, solutions based on social network analysis have been proposed to detect the presence of sybils [42], [43]. However, these approaches are applicable to P2P systems which are aware of social connections between peers. Moreover, [43] assumes the knowledge about some fraction of honest users whereas [42] assumes that the entire (or at least a part of the) topology of the network is known. Our solution makes no such assumption and is applicable to any P2P system involving even unknown peers.

Choosing a *non-sybil* peer with high probability, in a decentralized P2P system, is a non-trivial task. In [24], any node closest to some point in the identity space is considered as *non-sybil* node and used to maintain a secure routing table. Our *SyMon* selection method 1 is similar to this approach. However, in section V, we have shown that the cost incurred by an *adversary* to break this defense is not very high. Though our other selection methods also adopt the *closeness* metric between peer identities as a measure of *non-sybilness*, they can inflict very high cost on the *adversary*.

Monitoring of a transaction by a *Trusted Third Party* (TTP), has been proposed earlier to protect honest peers from malicious peers. To overcome DOS attacks, [8] relies on a centralized secure *bastion* to generate puzzles for clients. Though [9] *harvests* online sources to randomly generate puzzles, it can still be manipulated by sybils. In our approach, *SyMon* is chosen in a fully decentralized manner to monitor the transactions and its *non-sybilness* is verifiable.

IX. Conclusion

In this paper, we have described *SyMon*, a fully decentralized solution to enable every honest peer to defend itself against Sybil attack in large structured P2P systems. We have introduced a novel concept of transaction monitoring as a way of protecting honest peers from sybils. This approach can succeed only when a non-sybil peer(*SyMon*) is chosen to monitor the transaction. We have proposed four different methods for choosing with high probability, a non-sybil peer as *SyMon*, for any given peer in the system. Through theoretical as well as experimental evaluation, we have shown the efficacy of each of our proposed *SyMon* selection methods. Among them, selection methods 3 and 4 outperform other methods in terms of inflicting very high cost on an adversary and incurring relatively lesser cost on honest peers in discovering eligible *SyMon*. Methods 3 and 4 also distribute the transaction monitoring load uniformly on all peers in the system compared to other methods. Our sybil defense approach is generic in nature and can be integrated with any of the existing peer admission control protocols to limit the impact of Sybil Attack on the system.

As part of our future work, we intend to build a full-fledged reputation framework for P2P based file sharing applications that utilizes *SyMon* to fight against sybils. We also intend to study the effectiveness of this P2P reputation system in defending against sybils.

References

[1] http://www.napster.com/.

[2] http://bitconjurer.org/BitTorrent/.

[3] http://www.kazaa.com/.

[4] J. R. Douceur, "The sybil attack," in *IPTPS '01: The First International Workshop on Peer-to-Peer Systems*, 2002.

[5] R. Bhattacharjee and A. Goel, "Avoiding ballot stuffing in ebay-like reputation systems," in *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, 2005.

[6] N. Borisov, "Computational puzzles as sybil defenses," in *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, 2006.

[7] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *ISOC '99: Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, 1999.

[8] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten, "New client puzzle outsourcing techniques for dos resistance," in *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, 2004.

[9] J. A. Halderman and B. Waters, "Harvesting verifiable challenges from oblivious online sources," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[10] J. Dinger and H. Hartenstein, "Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration," in *ARES '06: Proceedings of the First IEEE International Conference on Availability, Reliability and Security*, 2006.

[11] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta, "Limiting sybil attacks in structured p2p networks," in *INFOCOM 2007: 26th IEEE International Conference on Computer Communications*, 2007.

[12] N. Tran, B. Min, J. Li, and L. Submaranian, "Sybil-resilient online content voting," in *Proceedings of the 6th Symposium on Networked System Design and Implementation (NSDI)*, 2009.

[13] E. J. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, vol. 10, no. 2, pp. 173–199, 2001.

[14] Y. Yang, Q. Feng, Y. L. Sun, and Y. Dai, "Reptrap: a novel attack on feedback-based reputation systems," in *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, 2008.

[15] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing*, 2001.

[16] http://www.skype.com/.

[17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys and Tutorials, IEEE*, vol. 7, no. 2, pp. 72–93, Quarter 2005.

[18] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, 2001.

[19] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, 2003.

[20] A. Shamir and E. Tromer, "On the cost of factoring rsa-1024," *RSA CryptoBytes*, vol. 6, no. 2, pp. 10–19, 2003.

[21] "Secure hash algorithm," http://www.ietf.org/rfc/rfc3174.txt.

[22] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "Spki certificate theory," RFC 2693, 1999.

[23] S. Chokhani, W. Ford, R. V. Sabett, C. R. Merrill, and S. S. Wu, "Internet x.509 public key infrastructure certificate policy and certification practices framework," RFC 3647, 2003.

[24] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, 2002.

[25] A. Back, "Hashcash - a denial of service countermeasure," http://www.hashcash.org/hashcash.pdf, 2002.

[26] J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than 2^n work," in *Advances in Cryptology EUROCRYPT*, 2005.

[27] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *IPTPS '03: The Third International Workshop on Peer-to-Peer Systems*, 2003.

[28] L. Massoulié, E. L. Merrer, A.-M. Kermarrec, and A. Ganesh, "Peer counting and sampling in overlay networks: random walk methods," in *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, 2006.

[29] T. M. Shafaat, A. Ghodsi, and S. Haridi, "A practical approach to network size estimation for structured overlays," in *IWSOS '08: Proceedings of the 3rd International Workshop on Self-Organizing Systems*, 2008.

[30] E. M. Chan, C. A. Gunter, S. Jahid, E. Peryshkin, and D. Rebolledo, "Using rhythmic nonces for puzzle-based dos resistance," in *CSAW '08: Proceedings of the 2nd ACM workshop on Computer security architectures*, 2008.

[31] A. Dasgupta, "The matching, birthday and the strong birthday problem: a contemporary review," *Journal of Statistical Planning and Association*, no. 130, pp. 377–389, 2005.

[32] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full sha-1," in *Crypto-05*, 2005.

[33] K. Tiri, "Side-channel attack pitfalls," in *DAC '07: Proceedings of the 44th annual conference on Design automation*, 2007.

[34] http://www.freepastry.org/.

[35] S. Shin, J. Jung, and H. Balakrishnan, "Malware prevalence in the kazaa file-sharing network," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006.

[36] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in p2p file sharing systems," in *INFOCOM 2005: 24th IEEE International Conference on Computer Communications*, 2005.

[37] N. Christin, A. S. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, 2005.

[38] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*, 2003.

[39] K. Walsh and E. G. Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, 2005.
[40] C. Costa and J. Almeida, "Reputation systems for fighting pollution in peer-to-peer file sharing systems," in *P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, 2007.
[41] J. Liang, N. Naoumov, and K. W. Ross, "Efficient blacklisting and pollution-level estimation in p2p file-sharing systems," in *Proceedings of AINTEC 2005*, 2005.
[42] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks," in *NDSS '09: Proceedings of the 16th Annual Network and Distributed System Security Symposium*, 2009.
[43] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *SP '08: Proceedings of the IEEE Symposium on Security and Privacy*, 2008.