SyMon: Defending Large Structured P2P Systems Against Sybil Attack

Jyothi B S, Dharanipragada Janakiram
Dept. of CSE, IIT Madras, Chennai, India
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 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 propose 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.

1. Introduction

Among the security attacks that plague existing P2P systems, Sybil attack[8] 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* by exploiting the low or zero-cost entry barrier in P2P systems. These sybils are used to launch other types of security attacks thereby compromising the entire system.

Existing approaches: Researchers have attempted to defend against Sybil attack through various centralized as well as distributed approaches[2][7][8][18]. The main goal of these approaches is to limit the total number of sybils and hence 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. For example, in systems that rely on online voting schemes, it is important to prevent sybils from *out-voting* honest peers in the system.

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[11][25].

In Reputation systems, sybils can spoof transactions among themselves to raise their reputation through *ballot-stuffing*[1]. By logging fake feedbacks, they can spoof transactions 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. This necessitates the need for a mechanism by which every honest peer can protect itself from sybils.

Our approach: This paper proposes 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. To the best of our knowledge, ours is the first attempt to defend against Sybil attack through transaction monitoring process.

The process of monitoring a transaction is application dependent. We assume that it is effective in preventing i) a sybil from cheating an honest peer during the transaction and ii) a sybil *SyMon* from harming two honest peers during their transaction. For example, in P2P Reputation systems for file sharing applications, the *SyMon* of a provider peer can mandate that the file is transferred to a requester peer via itself. This process enables *SyMon* in establishing the legitimacy of the transaction and in verifying the quality of the file being transferred either manually or through automated methods[15][16]. The *SyMons'* feedbacks help new re-

978-1-4244-5067-1/09/$26.00 ©2009 IEEE

quester peers in verifying the past transaction history of provider peers and hence in identifying honest provider peers that serve good files. Thus, *SyMon* can help in preventing sybils from decreasing the content availability of the system.

SyMon can also help in preventing Denial-Of-Service(DOS) attack on a server. In DOS attack, sybils flood 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[2][13] can be employed to prevent this attack. The process of forming the puzzle challenge, when performed by the server itself or any other centralized entity[24], can unwittingly subject it to DOS attack. On the other hand, if any random peer is chosen as the puzzle server[12], then sybils can easily circumvent this. Hence, the process of forming the puzzle challenge can be *outsourced* to the client peer's *SyMon* thereby ensuring that the server is immune to DOS attack to a large extent.

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.

In this paper, we focus on the *SyMon* selection process. We propose four methods for *associating* any given peer with its *SyMon*. Each of our selection methods differs in its efficacy and the overhead incurred while choosing an eligible *SyMon* in the system. We propose a simple discovery protocol for finding *SyMon* in the system. We also provide a mechanism to verify that the chosen *SyMon* is not a sybil of the given peer with high probability. 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.

The rest of the paper is organized as follows. Section 2 describes our system model. In section 3, we present details about our system and then discuss its strengths and weaknesses against a range of attacks in section 4. We show the robustness of our proposed solution through simulation results in section 5. We compare our solution with existing research work in section 6. We conclude the paper in section 7 with a mention about the possible future research directions.

2. System Model

In our system model, we consider a fully decentralized large structured P2P overlay like Pastry[19] or Chord[21].

2.1. 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[23] to generate its peer identity. The peer generates its Identity Certificate (using SPKI[9]) 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 *uniformly* distributed over the identity space preventing sybils from choosing their own desired identities.

In order to prevent malicious users from creating a large number of sybils, every attempt to generate a peer identity is associated with some computational puzzle[2][3][8][12]. The puzzle result is embedded in the peer's Identity Certificate so that others can verify the same.

Every message exchange between peers is encrypted with the public/private key of the peers so as to prevent sybils from *masquerading* as other honest peers.

2.2. Threat Model

Peer model: The system includes honest users as well as malicious users. Each honest user follows the protocol 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* and attempt to subvert the system at all times.

Transaction model: We consider two transaction models. In the first model, 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*.

In the second model, 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[1]. Hence, the honest peer is considered as the *verifier* and the other two peers as *suspects*.

3. Proposed Solution Details

In our proposed system, every transacting peer is *associated* with another peer known as *SyMon*. We propose four methods of *associating* any given peer with its *SyMon*. These 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. Each of these methods has varying complexity and incurs varying cost on honest peers as well as an *adversary*. Depending on the application needs, any one of our selection methods can be employed to defend against Sybil attack.

We propose 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. 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*. *Alice*, an 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*. Hence, any peer can determine the legitimacy of the transaction between *Alice* and *Bob* even if it is not directly involved in the transaction by verifying the selection of *Carol*.

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
$\{\mathrm{ID}_A,\mathrm{ID}_B\}_\Phi$	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
$\mathrm{H}(\mathrm{N}_1) \rightarrow N_2$	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
$\mathrm{N}_1 \cdot N_2$	Concatenation of N_1 and N_2 numbers

3.1. SyMon Selection

In this section, we present different ways of choosing *SyMon* for a given peer. Each of our 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. 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 employing any of the approaches proposed in [3][17]. 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.

Selection Method 1: 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).

$$\{\mathrm{ID}_S \mid \{\mathrm{ID}_{Bob}, \mathrm{ID}_S\}_\Phi \text{ and } \Phi \geq \Phi_\beta\}$$
$$\text{where } 0 \leq \Phi < 40$$

Here, Φ_β is a constant dependent on the size of the network. When it is set to an appropriate value,

[1] In P2P Reputation systems, sybils can fake transactions among themselves to raise their reputation through *ballot-stuffing*[1]. New requester peers verify the past transactions of a provider peer while analyzing its reputation before selecting it for a transaction.

Bob's closest neighbor(s) can satisfy this criteria. To discover Bob's closest *live* neighbor, a *TraceRoute* message(M_{TR}) is sent to *Bob*'s neighbors (*LeafSet* peers in Pastry and *Predecessor/Successor* peers in Chord). 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.

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.

Selection Method 2: 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}).

$$\{\mathrm{ID}_{\mathrm{S}_1} \text{and } \mathrm{ID}_{\mathrm{S}_2} \mid \{\mathrm{ID}_{\mathrm{S}_1}, \mathrm{ID}_{\mathrm{S}_2}\}_\Phi \text{ and } \Phi \geq \Phi_\alpha\}$$
$$\text{where } 0 \leq \Phi < 40$$

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

Every peer finds its closest neighbor, as mentioned in selection method 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.

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 4.1.

Selection Method 3: 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 [3]. 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*.

The discovery protocol, in this method, involves finding a peer closest to *Bob's transient identity* in the system[2] and choosing *Bob's SyMon* from among the neighbors of this discovered peer as detailed in selection method 1.

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.

Selection Method 4: This method is similar to selection method 3. However, in this method, *Bob's tran-*

[2] In structured overlays like Pastry and Chord, a *live* peer whose identity is closest to the given *key*(the peer's *transient identity*) is found by routing a message with that key as the destination address.

sient 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 [4]. 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.

3.2. 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 6 fails.

4. 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.

4.1. Choosing SyMon

The formal model underlying our solution casts the problem of finding a *SyMon* for any given peer as the *classical birthday problem*[3] and its family of problems[6] with respect to SHA-1, the hash function used to generate peer identities (see section 2.1). 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*[6][4] on SHA-1. This is because only Φ digits collide between the peer identities where $0 \leq \Phi < 40$.

A *partial collision search* on SHA-1 can also be considered as a *collision search*[23][3] 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 [23], we can determine an approximate value of Φ_α as shown below[5]:

$$r = \sqrt{2^x} = \sqrt{2^{4\Phi_\alpha}} \qquad (1)$$

Selection methods 1, 3 and 4: In these methods, a peer whose identity matches with the given peer's iden-

[3] Collision search on a hash function is similar to the *classical birthday problem* which refers to the probability of finding *some* pair of people having the same birthday in a set of randomly chosen people.

[4] Partial collision search on a hash function is similar to the *almost birthday problem* which 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.

[5] The theoretical bound on collision search and preimage search holds good since the input(peer identity) cannot be modified arbitrarily.

tity (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*[6][6] on SHA-1.

A *partial preimage search* on SHA-1 can also be viewed as a *preimage search*[14][7] on H′ as mentioned above. 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 [14][5]:

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

It is important to note that 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*[6][8] on H′ or *strong partial preimage search* on SHA-1. 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 [6], we can determine an approximate value of Φ_β as shown below:

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

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 [6]:

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

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: $m = 2^{4\Phi_\alpha}$.

4.2. 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*[22] 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 4.1. 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 4.1). 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* (see section 3.1). Once the desired sybil pair is ob-

[6] Partial preimage search on a hash function is similar to the *almost same birthday as you* problem which 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.

[7] Preimage search on a hash function is similar to *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.

[8] *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.

Table 1. 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

Table 2. Security margin offered by SyMon selection methods

Ntw 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

tained, the *adversary* can also attempt to *reuse* them for future transactions with other honest peers.

4.3. 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 5.1).

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

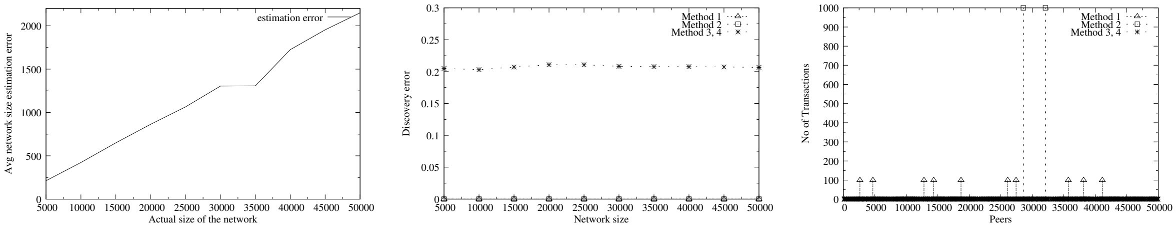
Table 1 and Table 2 compare each of our proposed *SyMon* selection methods under different attack strategies of an *adversary*. Table 2 also shows the approximate number of sybils required to subvert the system. The actual cost incurred by the *adversary*, in breaking our defense, is the product of the required number of sybils and the cost associated with generating each sybil (see section 2.1). 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.

5. 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 $\mathbb{N}$ simulated through FreePastry[10]. All our experiments were run on a Intel-P4 Quad Core 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. 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) 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

Figure 1. Effectiveness of the SyMon discovery protocol

5.1. SyMon Discovery Protocol

Network Size Estimation: In this experiment, every peer estimated the size of the network by checking the distance between the identities of each of its *LeafSet* peers, as proposed in [17]. 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. 1(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, every peer *discovered* its *SyMon* as per each of our selection methods (refer section 3.1). 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. 1(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.

Load Distribution on SyMon peers: In this experiment, each of the randomly chosen 10 peers performed 100 transactions with other randomly chosen peers in the system. 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. 1(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, the load is uniformly distributed on all peers in the system since *SyMon* is chosen based on the randomly generated *transient identity* of a peer.

In our experiments, selection method 2 suffers from highly skewed load distribution since we have considered the best case of choosing only two peers through a *partial collision search*. The load can be balanced more evenly as discussed in section 4.1.

5.2. SyMon Defense scheme

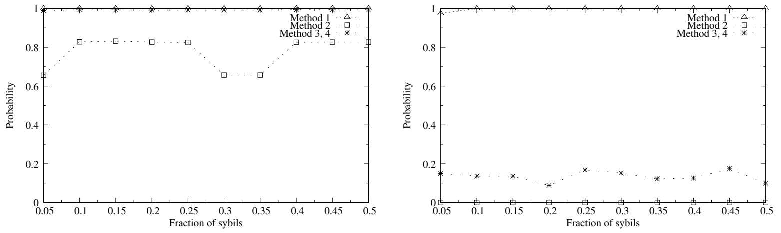
In this set of experiments, we study the effectiveness of our sybil defense scheme against Sybil attack.

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 3.2). We then compared its decision against the actual type of peer chosen as the *suspect* peer in order to determine the false positives. These false positives arise when the honest *verifier* peer sets its Φ_α/Φ_β to a very high or very low value.

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

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. 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. 3(a).

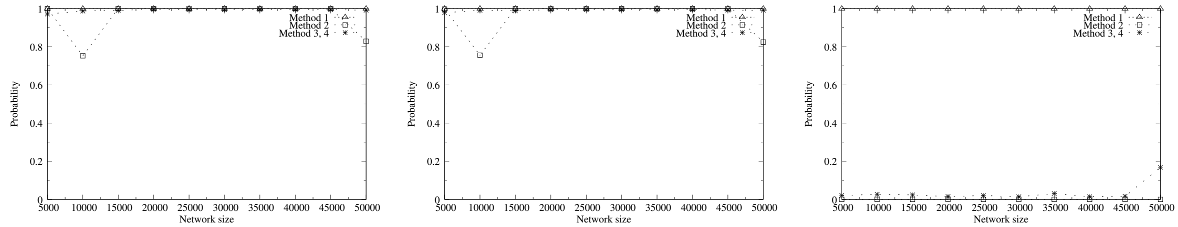
Effectiveness in the presence of sybils: In these experiments, we considered a network of size 50k and varied the percentage of sybils. When only honest peers were chosen as *suspect*, our selection methods 1, 3 and



(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

Figure 2. Effectiveness of SyMon as a sybil defense system at different fractions of sybils



(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

Figure 3. Effectiveness of SyMon as a sybil defense system at different network sizes

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. 2(a)).

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. 2(b)). These results corroborate our theoretical analysis.

We repeated the above two experiments by varying the network size when the fraction of sybils was set to 0.25. Fig. 3(b) and Fig. 3(c) show that the effectiveness of our solution does not vary much with the variations in the size of the network.

6. Related Work

Many of the existing P2P systems are prone to Sybil attack. As per [8], 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[2][3][7][18] 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[5][26]. However, these approaches are applicable to P2P systems which are aware of social connections between peers. Moreover, [26] assumes the knowledge about some fraction of honest users whereas [5] 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 [3], 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 4, 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, [24] relies on a centralized secure *bastion* to generate puzzles for clients. Though [12] *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.

7. Conclusion and Future Work

In this paper, we have introduced a novel concept of *transaction monitoring* by *SyMon* to protect honest peers from sybils. We have proposed four different methods for choosing with high probability, a *non-sybil SyMon* for any given peer. Through theoretical as well as experimental evaluation, we have shown the efficacy of each of our proposed *SyMon* selection methods in defending against Sybil attack.

As part of our future work, we intend to develop the incentive models to motivate peers chosen as *SyMon* to monitor the transactions. We also intend to further study the effectiveness of our proposed system in defending P2P Reputation systems against sybils.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] A. Dasgupta. The matching, birthday and the strong birthday problem: a contemporary review. *Journal of Statistical Planning and Association*, (130):377–389, 2005.

[7] 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.

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

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

[10] FreePastry. http://www.freepastry.org/.

[11] E. J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] A. Shamir and E. Tromer. On the cost of factoring rsa-1024. *RSA CryptoBytes*, 6(2):10–19, 2003.

[21] 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.*, 11(1), 2003.

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

[23] X. Wang, Y. Y. L, and Y. Hongbo. Finding collisions in the full sha-1. In *Crypto-05*, 2005.

[24] 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.

[25] 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.

[26] 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.