SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks

Haifeng Yu
National University of Singapore
haifeng@comp.nus.edu.sg

Phillip B. Gibbons
Intel Research Pittsburgh
phillip.b.gibbons@intel.com

Michael Kaminsky
Intel Research Pittsburgh
michael.e.kaminsky@intel.com

Feng Xiao
National University of Singapore
xiaof@comp.nus.edu.sg

Abstract

Decentralized distributed systems such as peer-to-peer systems are particularly vulnerable to sybil attacks*, where a malicious user pretends to have multiple identities (called* sybil nodes*). Without a trusted central authority, defending against sybil attacks is quite challenging. Among the small number of decentralized approaches, our recent SybilGuard protocol [43] leverages a key insight on social networks to bound the number of sybil nodes accepted. Although its direction is promising, SybilGuard can allow a large number of sybil nodes to be accepted. Furthermore, SybilGuard assumes that social networks are fast mixing, which has never been confirmed in the real world.*

This paper presents the novel SybilLimit protocol that leverages the same insight as SybilGuard but offers dramatically improved and near-optimal guarantees. The number of sybil nodes accepted is reduced by a factor of $\Theta(\sqrt{n})$, or around 200 times in our experiments for a million-node system. We further prove that SybilLimit's guarantee is at most a $\log n$ factor away from optimal, when considering approaches based on fast-mixing social networks. Finally, based on three large-scale real-world social networks, we provide the first evidence that real-world social networks are indeed fast mixing. This validates the fundamental assumption behind SybilLimit's and SybilGuard's approach.

1. Introduction

Decentralized distributed systems (such as peer-to-peer systems) are particularly vulnerable to *sybil attacks* [11], where a malicious user pretends to have multiple identities (called *sybil identities* or *sybil nodes*). In fact, such sybil attacks have already been observed in the real world [19, 40] in the Maze peer-to-peer system. Researchers have also demonstrated [35] that it is surprisingly easy to launch sybil attacks in the widely-used eMule system [12].

When a malicious user's sybil nodes comprise a large fraction of the nodes in the system, that one user is able to "out vote" the honest users in a wide scope of collaborative tasks. Examples of such collaborative tasks range from Byzantine consensus [18] and voting schemes for email spam [31] to implicit collaboration in redundant routing and data replication in Distributed Hash Tables (DHTs) [7]. The exact form of such collaboration and the exact fraction of sybil nodes these collaborative tasks can tolerate may differ from case to case. However, a generic requirement is that the number of sybil nodes (compared to the number of honest users) needs to be properly bounded.

To defend against sybil attacks, simply monitoring each node's historical behavior is often insufficient because sybil nodes can behave nicely initially, and then launch an attack. Although a trusted central authority can thwart such attacks by issuing credentials to actual human beings or requiring payment [22], finding such a single entity that every user worldwide is willing to trust can be difficult or impossible (especially if that entity requires users to provide sensitive information).

Without a trusted central authority, defending against sybil attacks is much harder. Among the small number of approaches, the simplest one perhaps is to bind identities to IP addresses or IP prefixes. Another approach is to require every identity to solve puzzles that require human effort, such as CAPTCHAs [36]. Both approaches can provide only limited protection—the adversary can readily steal IP addresses with different prefixes in today's Internet [32], while CAPTCHAs can be re-posted on an adversary's website to be solved by users seeking access to that site.

The SybilGuard approach. Recently, we proposed SybilGuard [43], a new protocol for defending against sybil attacks without relying on a trusted central authority. SybilGuard leverages a key insight regarding *social networks* (Figure 1). In a social network, the vertices (nodes) are iden-

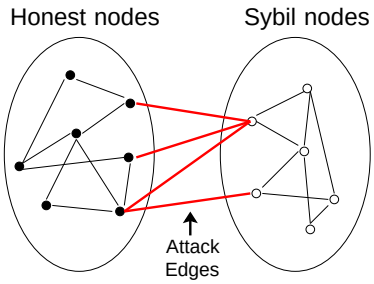

Figure 1. The social network.

tities in the distributed system and the (undirected) edges correspond to human-established trust relations in the real world. The edges connecting the honest region (i.e., the region containing all the honest nodes) and the sybil region (i.e., the region containing all the sybil identities created by malicious users) are called *attack edges*. SybilGuard ensures that the number of attack edges is independent of the number of sybil identities, and is limited by the number of trust relation pairs between malicious users and honest users. SybilGuard observes that if malicious users create too many sybil identities, the graph will have a small *quotient cut*—i.e., a small set of edges (the attack edges) whose removal disconnects a large number of nodes (all the sybil identities). On the other hand, "fast mixing" [26] social networks do not tend to have such cuts. SybilGuard leverages the small quotient cut to limit the size of sybil attacks.

SybilGuard is a completely decentralized protocol and enables any honest node V (called the *verifier*) to decide whether or not to *accept* another node S (called the *suspect*). "Accepting" means that V is willing to do collaborative tasks with S. SybilGuard's provable (probabilistic) guarantees hold for $(1-\epsilon)n$ verifiers out of the n honest nodes, where ϵ is some small constant close to 0. (The remaining nodes get degraded, not provable, protection.) Assuming fast-mixing social networks and assuming the number of attack edges is $o(\sqrt{n}/\log n)$, SybilGuard guarantees that any such verifier, with probability at least $1-\delta$ (δ being a small constant close to 0), will accept at most $O(\sqrt{n}\log n)$ sybil nodes per attack edge and at least $(1-\epsilon)n$ honest nodes.

While its direction is promising, SybilGuard suffers from two major limitations. First, although the end guarantees of SybilGuard are stronger than previous decentralized approaches, they are still rather weak in the absolute sense: Each attack edge allows $O(\sqrt{n}\log n)$ sybil nodes to be accepted. In a million-node synthetic social network, the number of sybil nodes accepted per attack edge is nearly 2000 [43]. The situation can get worse: When the number of attack edges $g = \Omega(\sqrt{n}/\log n)$ (or $g > 15,000$ in the million-node synthetic social network), SybilGuard can no longer bound the number of accepted sybil nodes *at all*. Second, SybilGuard critically relies on the assumption that social networks are fast mixing, an assumption that had never not been validated in the real world.

Number of attack edges g (unknown to protocol)	**SybilGuard accepts**	**SybilLimit accepts**
$o(\sqrt{n}/\log n)$	$O(\sqrt{n}\log n)$	$O(\log n)$
$\Omega(\sqrt{n}/\log n)$ to $o(n/\log n)$	unlimited	$O(\log n)$
below $\sim 15,000$	~ 2000	~ 10
above $\sim 15,000$ and below $\sim 100,000$	unlimited	~ 10

Table 1. Number of sybil nodes accepted per attack edge (out of an unlimited number of sybil nodes), both asymptotically for n honest nodes and experimentally for a million honest nodes. Smaller is better.

SybilLimit: A near-optimal protocol for real-world social networks. In this paper, we present a new protocol that leverages the same insight as SybilGuard but offers dramatically improved and near-optimal guarantees. We call the protocol *SybilLimit*, because i) it limits the number of sybil nodes accepted and ii) it is near-optimal and thus pushes the approach to the limit. For any $g = o(n/\log n)$, SybilLimit can bound the number of accepted sybil nodes per attack edge within $O(\log n)$ (see Table 1). This is a $\Theta(\sqrt{n})$ factor reduction from SybilGuard's $O(\sqrt{n}\log n)$ guarantee. In our experiments on the million-node synthetic social network used in [43], SybilLimit accepts on average around 10 sybil nodes per attack edge, yielding nearly 200 times improvement over SybilGuard. Putting it another way, with SybilLimit, the adversary needs to establish nearly 100,000 real-world social trust relations with honest users in order for the sybil nodes to out-number honest nodes, as compared to 500 trust relations in SybilGuard. We further prove that SybilLimit is at most a $\log n$ factor from optimal in the following sense: for any protocol based on the mixing time of a social network, there is a lower bound of $\Omega(1)$ on the number of sybil nodes accepted per attack edge. Finally, SybilLimit continues to provide the same guarantee even when g grows to $o(n/\log n)$, while SybilGuard's guarantee is voided once $g = \Omega(\sqrt{n}/\log n)$. Achieving these near-optimal improvements in SybilLimit is far from trivial and requires the combination of multiple novel techniques. SybilLimit achieves these improvements without compromising on other properties as compared to SybilGuard (e.g., guarantees on the fraction of honest nodes accepted).

Next, we consider whether real-world social networks are sufficiently fast mixing for protocols like SybilGuard and SybilLimit. Even though some simple synthetic social network models [17] have been shown [6, 14] to be fast mixing under specific parameters, whether real-world so-

cial networks are indeed fast mixing is controversial [2]. In fact, social networks are well-known [3, 15, 24, 38] to have groups or communities where intra-group edges are much denser than inter-group edges. Such characteristics, on the surface, could very well prevent fast mixing. To resolve this question, we experiment with three large-scale (up to nearly a million nodes) real-world social network datasets crawled from `www.friendster.com`, `www.livejournal.com`, and `dblp.uni-trier.de`. We find that despite the existence of social communities, even social networks of such large scales tend to mix well within a rather small number of hops (10 to 20 hops), and SybilLimit is quite effective at defending against sybil attacks based on such networks. These results provide the first evidence that real-world social networks are indeed fast mixing. As such, they validate the fundamental assumption behind the direction of leveraging social networks to limit sybil attacks.

2. Related work

The negative results in Douceur's initial paper on sybil attacks [11] showed that sybil attacks cannot be prevented unless special assumptions are made. Some researchers [9] proposed exploiting the *bootstrap graph* of DHTs. Here, the insight is that the large number of sybil nodes will all be introduced (directly or indirectly) into the DHT by a small number of malicious users. Bootstrap graphs may appear similar to our approach, but they have the drawback that an honest user may also indirectly introduce a large number of other honest users. Such possibility makes it difficult to distinguish malicious users from honest users. Instead of simply counting the number of nodes introduced directly and indirectly, SybilLimit distinguishes sybil nodes from honest nodes based on graph mixing time. It was shown [9] that the effectiveness of the bootstrap graph approach deteriorates as the adversary creates more and more sybil nodes, whereas SybilLimit's guarantees hold no matter how many sybil nodes are created. Some researchers [5] assume that the attacker has only one or small number of network positions in the Internet. If such assumption holds, then all sybil nodes created by the attacker will have similar network coordinates [29]. Unfortunately, once the attacker has more than a handful of network positions, the attacker can fabricate arbitrary network coordinates.

In reputation systems, colluding sybil nodes may artificially increase a (malicious) user's rating (e.g., in Ebay). Some systems such as Credence [37] rely on a trusted central authority to prevent this. There are existing distributed defenses [8, 13, 33] to prevent such artificial rating increases. These defenses, however, cannot bound the number of sybil nodes accepted, and in fact, all the sybil nodes can obtain the same rating as the malicious user. Sybil attacks and related problems have also been studied in sensor networks [28, 30], but the approaches and solutions usually rely on the unique properties of sensor networks (e.g., key predistribution). Margolin et al. [23] proposed using cash rewards to motivate one sybil node to reveal other sybil nodes, which is complimentary to bounding the number of sybil nodes accepted in the first place.

Social networks are one type of trust networks. There are other types of trust networks, e.g., based on historical interactions/transactions between users [8, 13, 37]. As in LOCKSS [21], Ostra [25], and SybilGuard [43], SybilLimit assumes a social network with a much stronger associated trust than these other types of trust networks [8, 13, 37]. LOCKSS uses social networks for digital library maintenance, and not as a general defense against sybil attacks. Ostra leverages social networks to prevent the adversary from sending excessive unwanted communication. In comparison, SybilLimit's functionality is more general: Because SybilLimit already bounds the number of sybil nodes, it can readily provide functionality equivalent to Ostra by allocating each node a communication quota. Furthermore, different from Ostra, SybilLimit has strong, provable end guarantees and has a complete design that is decentralized. The relationship between SybilGuard and SybilLimit is discussed in more detail in Sections 4 and 5.3. Unlike many other works [8, 13, 33, 37] on trust networks, SybilLimit does not use trust propagation in the social network.

Mislove et al. [24] also studied the graph properties of several online real-world social networks. But Mislove et al. did not focus on mixing time properties or their appropriateness for defending against sybil attacks. Finally, a preliminary version of this work appeared as [41].

3. System model and attack model

SybilLimit adopts a similar system model and attack model as SybilGuard [43]. The system has n honest human beings as *honest users*, each with one *honest identity/node*. Honest nodes obey the protocol. The system also has one or more malicious human beings as *malicious users*, each with one or more identities/nodes. To unify terminology, we call all identities created by malicious users as *sybil identities/nodes*. Sybil nodes are byzantine and may behave arbitrarily. All sybil nodes are colluding and are controlled by an *adversary*. A compromised honest node is completely controlled by the adversary and hence is considered as a sybil node and not as an honest node.

There is an undirected social network among all the nodes, where each undirected edge corresponds to human-established trust relations in the real world. The adversary may create arbitrary edges among sybil nodes in the social network. Each honest user knows her neighbors in the social network, while the adversary has full knowledge of the

entire social network. The honest nodes have m undirected edges among themselves in the social network. For expository purposes, we sometimes also consider the m undirected edges as $2m$ directed edges. The adversary may eavesdrop on any messages sent in the protocol.

Every node is simultaneously a suspect and a verifier. As in SybilGuard, we assume that each suspect S has a locally generated public/private key pair, which serves to prevent the adversary from "stealing" S's identity after S is accepted. When a verifier V accepts a suspect S, V actually accepts S's public key, which can be used later to authenticate S. We do not assume a public key infrastructure, and the protocol does not need to solve the public key distribution problem since the system is not concerned with binding public keys to human beings or computers. A malicious user may create multiple different key pairs for her different sybil nodes.

4. Background: SybilGuard

To better understand the improvements of SybilLimit over SybilGuard and the challenges involved, this section provides a concise review of SybilGuard.

Random walks and random routes. SybilGuard uses a special kind of random walk, called *random routes*, in the social network. In a random walk, at each hop, the current node flips a coin on-the-fly to select a uniformly random edge to direct the walk (the walk is allowed to turn back). For random routes, each node uses a pre-computed random permutation, "$x_1x_2...x_d$" where d is the degree of the node, as a *one-to-one mapping* from incoming edges to outgoing edges. A random route entering via edge i will always exit via edge x_i. This pre-computed permutation, or *routing table*, serves to introduce *external correlation* across multiple random routes. Namely, once two random routes traverse the same directed edge, they will merge and stay merged (i.e., they *converge*). Furthermore, the outgoing edge uniquely determines the incoming edge as well; thus the random routes can be *back-traced*. These two properties are key to SybilGuard's guarantees. As a side effect, such routing tables also introduce *internal correlation* within a single random route. Namely, if a random route visits the same node more than once, the exiting edges will be correlated. We showed [43] that such correlation tends to be negligible, and moreover, in theory it can be removed entirely using a more complex design. Thus, we ignore internal correlation from now on.

Without internal correlation, the behavior of a single random route is exactly the same as a random walk. In connected and non-bipartite graphs, as the length of a random walk goes toward infinity, the distribution of the last node (or edge) traversed becomes independent of the starting node of the walk. Intuitively, this means when the walk is sufficiently long, it "forgets" where it started. This final distribution of the last node (or edge) traversed is called the node (or edge) *stationary distribution* [26] of the graph. The edge stationary distribution (of any graph) is always a uniform distribution, while the node stationary distribution may not be. *Mixing time* [26] describes how fast we approach the stationary distribution as the length of the walk increases. More precisely, mixing time is the walk length needed to achieve a certain *variation distance* [26], Δ, to the stationary distribution. Variation distance is a value in $[0, 1]$ that describes the "distance" between two distributions—see [26] for the precise definition. A small variation distance means that the two distributions are similar. For a graph (family) with n nodes, we say that it is *fast mixing* if its mixing time is $O(\log n + \log \frac{1}{\Delta})$. In this paper, we only care about $\Delta = \Theta(\frac{1}{n})$, and we will simply say that a fast mixing graph has $O(\log n)$ mixing time. The following known result follows directly from the definition of mixing time and a useful interpretation of variation distance (Theorem 5.2 in [20]). This result is all we need in this paper about mixing time:

Theorem 1 *Consider any fast mixing graph with n nodes. A random walk of length $\Theta(\log n)$ is sufficiently long such that with probability at least $1 - \frac{1}{n}$, the last node/edge traversed is drawn from the node/edge stationary distribution of the graph.*

In SybilGuard, a random walk starting from an honest node in the social network is called *escaping* if it ever crosses any attack edge.

Theorem 2 *(from [43]) In any connected social network with n nodes and g attack edges, the probability of a length-l random walk starting from a uniformly random honest node being escaping is at most gl/n.*

Accepting honest nodes. In SybilGuard, each node performs a random route of length $l = \Theta(\sqrt{n} \log n)$. A verifier V only accepts a suspect S if S's random route intersects with V's. Theorem 2 tells us that V's random route will stay in the honest region with probability at least $1 - gl/n = 1 - o(1)$ for $g = o(\sqrt{n}/\log n)$. Theorem 1 further implies that with high probability, a random route $\Theta(\sqrt{n} \log n)$ long will include $\Theta(\sqrt{n})$ independent random nodes drawn from the node stationary distribution. It then follows from the generalized Birthday Paradox [1, 27] that an honest suspect S will have a random route that intersects with V's random route with probability $1 - \delta$ for any given (small) constant $\delta > 0$.

Bounding the number of sybil nodes accepted. To intersect with V's non-escaping random route, a sybil suspect's random route must traverse one of the attack edges. Consider Figure 2 where there is only a single attack edge. Because of the convergence property, all the random routes

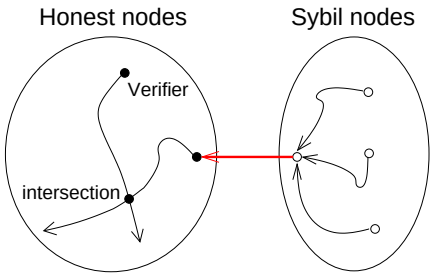

Figure 2. Routes over the same edge merge.

from all sybil suspects must merge completely once they traverse the attack edge. All these routes differ only in how many hops of the route remain after crossing the attack edge (between 1 and $l-1$ hops for a length-l route). Because the remaining parts of these routes are entirely in the honest region, they are controlled by honest nodes. Thus, there will be fewer than $l = O(\sqrt{n}\log n)$ random routes that emerge from the sybil region. In general, the number of such routes will be $O(g\sqrt{n}\log n)$ for g attack edges. SybilGuard is designed such that only one public key can be *registered* at the nodes on each random route. This means that the adversary can register only $O(g\sqrt{n}\log n)$ public keys for all the sybil nodes combined. In order to accept a suspect S, V must find an intersection between its random route and S's random route and then confirm that S is properly registered at the intersecting node. As a result, only $O(\sqrt{n}\log n)$ sybil nodes will be accepted per attack edge. For $g = o(\sqrt{n}/\log n)$, the total number of sybil nodes accepted is $o(n)$.

Estimating the needed length of random routes. While the length of the random routes is $\Theta(\sqrt{n}\log n)$, the value of n is unknown. In SybilGuard, nodes locally determine the needed length of the random routes via sampling. Each node is assumed to know a rough upper bound Z on the mixing time. To obtain a sample, a node A first performs a random walk of length Z, ending at some node B. Next A and B each perform random routes to determine how long the routes need to be to intersect. A sample is *bad* (i.e., potentially influenced by the adversary) if any of the three random walks/routes in the process is escaping. Applying Theorem 2 shows that the probability of a sample being bad is at most $3gl/n = o(1)$ for $g = o(\sqrt{n}/\log n)$.

5. SybilLimit protocol

As summarized in Table 1, SybilGuard accepts $O(\sqrt{n}\log n)$ sybil nodes per attack edge and further requires g to be $o(\sqrt{n}/\log n)$. SybilLimit, in contrast, aims to reduce the number of sybil nodes accepted per attack edge to $O(\log n)$ and further to allow for $g = o(n\log n)$. This is challenging, because SybilGuard's requirement on $g = o(\sqrt{n}/\log n)$ is fundamental in its design and is simultaneously needed to ensure:

- **Sybil nodes accepted by SybilGuard.** The total number of sybil nodes accepted, $O(g\sqrt{n}\log n)$, is $o(n)$.
- **Escaping probability in SybilGuard.** The escaping probability of the verifier's random route, $O(g\sqrt{n}\log n/n)$, is $o(1)$.
- **Bad sample probability in SybilGuard.** When estimating the random route length, the probability of a bad sample, $O(g\sqrt{n}\log n/n)$, is $o(1)$.

Thus to allow for larger g, SybilLimit needs to resolve all three issues above. Being more "robust" in only one aspect will not help.

SybilLimit has two component protocols, a *secure random route protocol* (Section 5.1) and a *verification protocol* (Section 5.2). The first protocol runs in the background and maintains information used by the second protocol. Some parts of these protocols are adopted from SybilGuard, and we will indicate so when describing those parts. To highlight the major novel ideas in SybilLimit (as compared to SybilGuard), we will summarize these ideas in Section 5.3. Later, Section 6 will present SybilLimit's end-to-end guarantees.

5.1. Secure random route protocol

Protocol description. We first focus on all the suspects in SybilLimit, i.e., nodes seeking to be accepted. Figure 3 presents the pseudo-code for how they perform random routes—this protocol is adapted from SybilGuard with little modification. In the protocol, each node has a public/private key pair, and communicates *only* with its neighbors in the social network. Every pair of neighbors share a unique symmetric secret key (the *edge key*, established out-of-band [43]) for authenticating each other. A sybil node M_1 may disclose its edge key with some honest node A to another sybil node M_2. But because all neighbors are authenticated via the edge key, when M_2 sends a message to A, A will still route the message as if it comes from M_1. In the protocol, every node has a pre-computed random permutation $x_1x_2...x_d$ (d being the node's degree) as its routing table. The routing table never changes unless the node adds new neighbors or deletes old neighbors. A random route entering via edge i always exits via edge x_i. A suspect S starts a random route by propagating along the route its public key K_S together with a counter initialized to 1. Every node along the route increments the counter and forwards the message until the counter reaches w, the length of a random route. In SybilLimit, w is chosen to be the mixing time of the social network; given a fast-mixing social network, $w = O(\log n)$.

Let "$A \rightarrow B$" be the last (directed) edge traversed by S's random route. We call this edge the *tail* of the random route. Node B will see the counter having a value of w and thus

Executed by each suspect S:
1. S picks a uniformly random neighbor Y;
2. S sends to Y: $\langle 1, S$'s public key K_S, MAC$(1||K_S)\rangle$ with the MAC generated using the edge key between S and Y;

Executed by each node B upon receiving a message $\langle i, K_S, \text{MAC}\rangle$ from some neighbor A:
1. discard the message if the MAC does not verify or $i < 1$ or $i > w$;
2. if $(i = w)$ { record K_S under the edge name "$K_A \rightarrow K_B$" where K_A and K_B are A's and B's public key, respectively;}
   else {
3. look up the routing table and determine to which neighbor (C) the random route should be directed;
4. B sends to C: $\langle i+1, K_S, \text{MAC}((i+1)||K_S)\rangle$ with the MAC generated using the edge key between B and C;
   }

Figure 3. Protocol for suspects to do random routes and register their public keys.

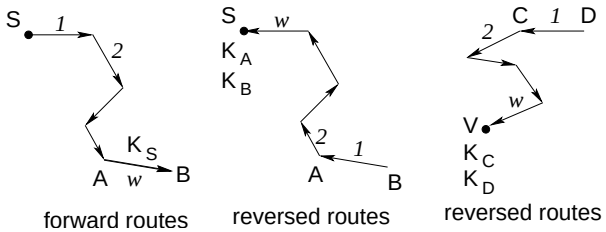

Figure 4. (i) Suspect S propagates K_S for w hops in an s-instance. (ii) K_A and K_B propagated back to suspect S in an s-instance. (iii) K_C and K_D propagated back to a verifier V in a v-instance.

record K_S under the name of that tail (more specifically, under the name of "$K_A \rightarrow K_B$" where K_A and K_B are A's and B's public key, respectively). Notice that B may potentially overwrite any previously recorded key under the name of that tail. When B records K_S, we say that S *registers* its public key with that tail. Our verification protocol, described later, requires that S know A's and B's public keys and IP addresses. To do so, similar to SybilGuard, SybilLimit invokes the protocol in Figure 3 a second time, where every node uses a "reversed" routing table (i.e., a random route entering via edge x_i will exit via edge i). This enables A and B to propagate their public keys and IP addresses backward along the route, so that S can learn about them (Figure 4).

Different from SybilGuard, SybilLimit invokes r independent instances (called *s-instances*) of the previous protocol for the suspects. The value of r should be $\Theta(\sqrt{m})$, and later we will explain how nodes can automatically pick the appropriate r. In every s-instance, each suspect uses the protocol in Figure 3 to perform one random route and to register its public key with the tail. Across all s-instances, a suspect will thus register its public key with r tails. Additionally in every s-instance, SybilLimit invokes the protocol a second time for each suspect using reversed routing tables, so that the suspects know their tails. The routing tables used in different s-instances are completely independent. Note, however, that all suspects share the same r s-instances—this is critical to preserve the desirable convergence/back-traceability property among their random routes in the same s-instance.

Similarly, every verifier performs r random routes. To avoid undesirable correlation between the verifiers' random routes and the suspects' random routes, SybilLimit uses another r independent instances (called *v-instances*) for all verifiers. Verifiers do not need to register their public keys—they only need to know their tails. Thus in each v-instance, SybilLimit invokes the protocol in Figure 3 once for each verifier, with reversed routing tables (Figure 4).

Performance overheads. While SybilLimit uses the same technique as SybilGuard to do random routes, the overhead incurred is different because SybilLimit uses multiple instances of the protocol with a shorter route length. Interestingly, using $\Theta(\sqrt{m})$ instances of the random route protocol does not incur extra storage or communication overhead by itself. First, a node does not need to store $\Theta(\sqrt{m})$ routing tables, since it can keep a single random seed and then generate any routing table on the fly as needed. Second, messages in different instances can be readily combined to reduce the number of messages. Remember that in all $\Theta(\sqrt{m})$ instances, a node communicates only with its neighbors. Given that the number of neighbors d is usually quite small on average (e.g., 20), a node needs to send only d messages instead of $\Theta(\sqrt{m})$ messages. Finally, the total number of bits a node needs to send in the protocol is linear with the number of random routes times the length of the routes. Thus, the total number of bits sent in the d messages in SybilLimit is $\Theta(\sqrt{m}\log n)$, as compared to $\Theta(\sqrt{n}\log n)$ in SybilGuard.

All these random routes need to be performed only one time (until the social network changes) and the relevant information will be recorded. Further aggressive optimizations are possible (e.g., propagating hashes of public keys instead of public keys themselves). We showed [43] that in a million-node system with average node degree be-

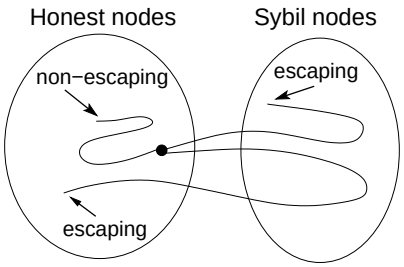

Figure 5. Escaping and non-escaping tails.

ing 10, an average node using SybilGuard needs to send 400KBs of data every few days. Under the same parameters, an average node using SybilLimit would send around $400 \times \sqrt{10} \approx 1300$KB of data every few days, which is still quite acceptable. We refer the reader to [43] for further details.

Basic security properties. The secure random route protocol provides some interesting basic security guarantees. We first formalize some notions. An honest suspect S has one *tail* in every s-instance, defined as the tail of its random route in that s-instance. We similarly define the r tails of a verifier. A random route starting from an honest node is called *escaping* if it ever traverses any attack edge. The tail of an escaping random route is called an *escaping tail* (Figure 5), even if the escaping random route eventually comes back to the honest region. By directing the random route in specific ways, the adversary can control/influence to which directed edge an escaping tail corresponds. But the adversary has no influence over non-escaping tails.

In any given s-instance, for every attack edge connecting honest node A and sybil node M, imagine that we perform a random route starting from the edge "$M \rightarrow A$", until either a subsequent hop traverses an attack edge or the length of the route reaches w. Because the adversary can fake a series of routes that each end on one of the edges on this route, these edges are called *tainted* tails. Intuitively, the adversary may register arbitrary public keys with these tails. In a given s-instance, one can easily see that the set of tainted tails is disjoint from the set of non-escaping tails from honest suspects. The reason is that random routes are back-traceable and starting from a non-escaping tail, one can always trace back to the starting node of the random route, encountering only honest nodes. This means that an honest suspect will never need to compete with the sybil nodes for a tail, as long as its random route is non-escaping.

After the secure random route protocol stabilizes (i.e., all propagations have completed), the following properties are guaranteed to hold:

- In every s-instance, each directed edge in the honest region allows only one public key to be registered.
- In every s-instance, an honest suspect S can always register its public key with its non-escaping tail (if any) in that s-instance.
- In every s-instance, among all the directed edges in the honest region, sybil nodes can register their public keys only with tainted tails. This is because nodes communicate with only their neighbors (together with proper authentication) and also because the counter in the registration message is incremented at each hop.
- In every s-instance (v-instance), if an honest suspect S (an honest verifier V) has a non-escaping tail "$A \rightarrow B$", then S (V) knows A's and B's public keys.

User and node dynamics. Most of our discussion so far assumes that the social network is static and all nodes are online. All techniques in SybilGuard to efficiently deal with user/node dynamics, as well as techniques to properly overwrite stale registration information for preventing certain attacks [43], apply to SybilLimit without modification. We do not elaborate on these due to space limitations.

5.2. Verification protocol

Protocol description. After the secure random route protocol stabilizes, a verifier V can invoke the verification protocol in Figure 6 to determine whether to accept a suspect S. S must satisfy both the *intersection condition* (Step 2–4 in Figure 6) and the *balance condition* (Step 5–7) to be accepted.

The intersection condition requires that S's tails and V's tails must intersect (instance number is ignored when determining intersection), with S being registered at the intersecting tail. In contrast, SybilGuard has an intersection condition on nodes (instead of on edges or tails). For the balance condition, V maintains r counters corresponding to its r tails (Figure 7). Every accepted suspect increments the "load" of some tail. The balance condition requires that accepting S should not result in a large "load spike" and cause the load on any tail to exceed $h \cdot \max(\log r, a)$. Here a is the current average load across all V's tails and $h > 1$ is some universal constant that is not too small (we use $h = 4$ in our experiments). In comparison, SybilGuard does not have any balance condition.

Performance overheads. The verification protocol can be made highly efficient. Except for Steps 1 and 3, all steps in the protocol involve only local computation. Instead of directly sending $\Theta(r)$ public keys in Step 1, S can readily use a Bloom Filter [26] to summarize the set of keys. In Step 3, for every intersecting tail in X, V needs to contact one node. On average, the number of intersections between a verifier V and an honest suspect S in the honest region is $O(1)$ with $r = \Theta(\sqrt{m})$, resulting in $O(1)$ messages. The adversary may intentionally introduce additional intersections in the sybil region between V's and S's escaping tails. However, if those extra intersecting nodes (introduced by the adversary) do not reply, V can blacklist them. If they do reply and

1. S sends to V its public key K_S and S's set of tails $\{(j, K_A, K_B) \mid S$'s tail in the jth s-instance is the edge "$A \rightarrow B$" and K_A (K_B) is A's (B's) public key$\}$;
// *Apply the* **intersection condition** *(the instance number is ignored when determining intersection)*
2. V computes the set of intersecting tails $X = \{(i, K_A, K_B) \mid (i, K_A, K_B)$ is V's tail and (j, K_A, K_B) is S's tail$\}$;
3. For every $(i, K_A, K_B) \in X$, V authenticates B using K_B and asks B whether S is registered under "$K_A \rightarrow K_B$"
   If not, remove (i, K_A, K_B) from X;
4. If X is empty then reject S and return;
// *Apply the* **balance condition** *(c_i is the counter for V's tail in the ith v-instance)*
5. Let $a = (1 + \sum_{i=1}^{r} c_i)/r$ and $b = h \cdot \max(\log r, a)$; // see text for description of h
6. Let c_{min} be the smallest counter among those c_i's corresponding to (i, K_A, K_B) that still remain in X (with tie-breaking favoring smaller i);
7. If $(c_{min} + 1 > b)$ then reject S; otherwise, increment c_{min} and accept S;

Figure 6. Protocol for V to verify S. V has r counters $c_1, ...c_r$ initialized to zero at start-up time.

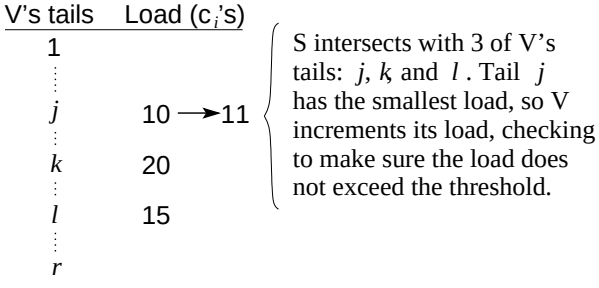

Figure 7. Balance condition example.

if V is overwhelmed by the overhead of such replies, then the adversary is effectively launching a DoS attack. Notice that the adversary can launch such a DoS attack against V even if V were not running SybilLimit. Thus such attacks are orthogonal to SybilLimit.

5.3. Key ideas in SybilLimit, vis-à-vis SybilGuard

This section highlights the key novel ideas in SybilLimit that eventually lead to the substantial end-to-end improvements over SybilGuard.

Intersection condition. To help convey the intuition, we will assume $g = 1$ in the following. In SybilLimit, each node uses $r = \Theta(\sqrt{m})$ random routes of length $w = \Theta(\log n)$ instead of a single random route of length $l = \Theta(\sqrt{n} \log n)$ as in SybilGuard.[1] In SybilGuard, each node along a random route corresponds to a "slot" for registering the public key of some node. The adversary can fake l distinct random routes of length l that cross the attack edge and enter the honest region. This means that the adversary will have $1 + 2 + ... + l = \Theta(l^2) = \Theta(n \log^2 n)$ slots for the sybil nodes in SybilGuard.

In SybilLimit, the tail of each random route corresponds to a "slot" for registration. In any given s-instance, the adversary can fake w distinct random routes of length w that cross the attack edge and enter the honest region. Notice that here SybilLimit reduces the number of such routes by using a w that is much smaller than l. Further, because we are concerned only with tails now, in the given s-instance, the adversary will have only w slots. With r s-instances, the adversary will have $r \cdot w = \Theta(\sqrt{m} \log n)$ such slots total, for all the sybil nodes. This reduction from $\Theta(n \log^2 n)$ slots to $\Theta(\sqrt{m} \log n)$ slots is the first key step in SybilLimit.

But doing r random routes introduces two problems. The first is that it is impossible for a degree-d node to have more that d distinct random routes, if we directly use SybilGuard's approach. SybilLimit observes that one can use many independent instances of the random route protocol, while still preserving the desired convergence/back-traceability property. The second problem is more serious. SybilGuard relies on the simple fact that the number of distinct routes from the adversary is l. All slots on the same route must have the same public key registered. This ensures that the total number of sybil nodes registered is l. In SybilLimit, there are $r \cdot w$ distinct routes from the adversary. Thus, a naive design may end up accepting $r \cdot w = \Theta(\sqrt{m} \log n)$ sybil nodes, which is even worse than SybilGuard. SybilLimit's key idea here is to perform intersections on edges instead of on nodes. Because the stationary distribution on edges is always uniform in any graph, it ensures that the *flip-side* of the Birthday Paradox holds. Namely, $\Theta(\sqrt{m})$ slots are both sufficient and *necessary* for intersection to happen (with high probability). Together with earlier arguments on the number of slots in SybilLimit, this will eventually allow us to prove that the number of sybil nodes with tails intersecting with V's non-escaping tails (more precisely, V's *uniform* non-escaping tails—see later) is $O(\log n)$ per attack edge.

Balance condition. In SybilGuard, the verifier's random route is either escaping or non-escaping, resulting in an "all-or-nothing" effect. For SybilGuard to work, this single random route must be non-escaping. Because of the large l of

[1] As an engineering optimization, a degree-d node in SybilGuard can perform d random routes of length $\Theta(\sqrt{n} \log n)$, but this does not improve SybilGuard's asymptotic guarantees.

$\Theta(\sqrt{n}\log n)$, the escaping probability will be $\Omega(1)$ once g reaches $\Omega(\sqrt{n}/\log n)$. Using much shorter random routes of length w in SybilLimit decreases such escaping probability. But on the other hand, because a verifier in SybilLimit needs to do r such routes, it remains quite likely that *some* of them are escaping. In fact, with $r = \Theta(\sqrt{m})$ and $w = \Theta(\log n)$, the probability of at least one of the r routes being escaping in SybilLimit is even larger than the probability of the single length-l random route being escaping in SybilGuard. Thus, so far we have only made the "all-or-nothing" effect in SybilGuard fractional.

SybilLimit relies on its (new) balance condition to address this fraction of escaping routes. To obtain some intuition, let us imagine the verifier V's tails as bins that can accommodate up to a certain load. When V accepts a suspect S, out of all of V's tails that intersect with S's tails, S conceptually increments the load of the least loaded tail/bin. Because of the randomness in the system, one would conjecture that all of V's tails should have similar load. If this is indeed true, then we can enforce a quota on the load of each tail, which will in turn bound the number of sybil nodes accepted by V's escaping tails. Later, we will show that the balance condition bounds the number within $O(g\log n)$.

Benchmarking technique. The SybilLimit protocol in Figures 3 and 6 assumes that $r = \Theta(\sqrt{m})$ is known. Obviously, without global knowledge, every node in SybilLimit needs to estimate r locally. Recall that SybilGuard also needs to estimate some system parameter (more specifically, the length of the walk). SybilGuard uses the sampling technique to do so, which only works for $g = o(\sqrt{n}/\log n)$. To allow any $g = o(n/\log n)$, SybilLimit avoids sampling completely. Instead, it use a novel and perhaps counter-intuitive *benchmarking technique* that mixes the real suspects with some random *benchmark suspects* that are already known to be mostly honest. The technique guarantees that a node will never over-estimate r regardless of the adversary's behavior. If the adversary causes an under-estimation for r, somewhat counter-intuitively, the technique can ensure that SybilLimit still achieves its end guarantees despite the under-estimated r. We will leave the detailed discussion to Section 7.

6. Provable guarantees of SybilLimit

While the intersection and balance conditions are simple at the protocol/implementation level, it is far from obvious why the designs provide the desired guarantees. We adopt the philosophy that all guarantees of SybilLimit must be proved mathematically, since experimental methods can cover only a subset of the adversary's strategies. Our proofs pay special attention to the correlation among various events, which turns out to be a key challenge. We cannot assume independence for simplicity because after all, SybilLimit exactly leverages external correlation among random routes. The following is the main theorem on SybilLimit's guarantee:

Theorem 3 *Assume that the social network's honest region is fast mixing and $g = o(n/\log n)$. For any given constants (potentially close to zero) $\epsilon > 0$ and $\delta > 0$, there is a set of $(1-\epsilon)n$ honest verifiers and universal constants w_0 and r_0, such that using $w = w_0 \log n$ and $r = r_0\sqrt{m}$ in SybilLimit will guarantee that for any given verifier V in the set, with probability at least $1-\delta$, V accepts at most $O(\log n)$ sybil nodes per attack edge and at least $(1-\epsilon)n$ honest nodes.*

For the remaining small fraction of ϵn honest verifiers, SybilLimit provides a degraded guarantee that is not provable. Because of space limitations, we will provide mostly intuitions in the following and leave formal/complete proofs to our technical report [42].

6.1. Intersection condition

Preliminaries: Classifying tails and nodes. As preparation, we first carefully classify tails and nodes. Table 2 summarizes the key definitions we will use. Consider a given verifier V (or suspect S) and a given v-instance (or s-instance). We classify its tail into 3 possibilities: i) the tail is an *escaping tail* (recall Section 5.1), ii) the tail is not escaping and is drawn from the (uniform) edge stationary distribution (i.e., a *uniform tail*), or iii) the tail is not escaping and is drawn from some unknown distribution on the edges (i.e., a *non-uniform tail*).[2] In a given v-instance, the routing tables of all honest nodes will entirely determine whether V's tail is escaping and in the case of a non-escaping tail, which edge is the tail. Thus, the adversary has no influence over non-escaping tails.

Because we do not know the distribution of the non-uniform tails, few probabilistic properties can be derived for them. Escaping tails are worse because their distribution is controlled by the adversary. Assuming that the honest region of the social network is fast mixing, our technical report [42] proves the following:

Lemma 4 *Consider any given constant (potentially close to zero) $\epsilon > 0$. We can always find a universal constant $w_0 > 0$, such that there exists a set H of at least $(1-\epsilon)n$ honest nodes (called* non-escaping nodes*) satisfying the following property: If we perform a length-w random walk starting from any non-escaping node with $w = w_0 \log n$, then the tail is a uniform tail (i.e., a uniformly random directed edge in the honest region) with probability at least $1 - O(\frac{g\log n}{n})$.*

[2] A finite-length random walk can only approach but never reach the stationary distribution. Thus a small fraction of tails will be non-uniform (also see Theorem 1).

Table 2. Terminology used in proofs (see text for precise definitions)

escaping route	random route from an honest node that traverses an attack edge
escaping tail	tail of an escaping route
tainted tail	any edge in the honest region on a length-w random route starting from an attack edge
uniform tail	non-escaping tail from the uniform edge distribution
non-uniform tail	non-escaping tail that is not a uniform tail
non-escaping node	honest node such that a length-w random walk has a uniform tail with $1 - o(1)$ probability
escaping node	honest node that is not a non-escaping node
uniform tail set	the set of all uniform tails of a given honest node
tainted tail set	set of all tainted tails

As a reminder, the probability in the above lemma is defined over the domain of all possible routing table states—obviously, if all routing tables are already determined, the tail will be some fixed edge.

It is still possible for the tail of a non-escaping node to be escaping or non-uniform—it is just that such probability is $O(\frac{g \log n}{n}) = o(1)$ for $g = o(n/\log n)$. An honest node that is not non-escaping is called an *escaping node*. By Lemma 4, we have at most ϵn escaping nodes; such nodes are usually near the attack edges. Notice that given the topology of the honest region and the location of the attack edges, we can fully determine the probability of the tail of a length-w random walk starting from a given node V being a uniform tail. In turn, this means whether a node V is escaping is not affected by the adversary. In the remainder of this paper, unless specifically mentioned, when we say "honest node/verifier/suspect", we mean "non-escaping (honest) node/verifier/suspect". We will not, however, ignore escaping nodes in the arguments since they may potentially disrupt the guarantees for non-escaping nodes.

For each verifier V, define its *tail set* as: $\{(i, e) \mid e$ is V's tail in the ith v-instance$\}$. V's *uniform tail set* $\mathcal{U}(V)$ is defined as:

$$\mathcal{U}(V) = \{(i, e) \mid e \text{ is } V\text{'s tail in the } i\text{th v-instance and } e \text{ is a uniform tail}\}$$

Notice that the distribution of $\mathcal{U}(V)$ is not affected by the adversary's strategy. We similarly define the tail set and uniform tail set for every suspect S. We define the *tainted tail set* ∇ as: $\nabla = \cup_{i=1}^{r} \nabla_i$, where

$$\nabla_i = \{(i, e) \mid e \text{ is a tainted tail in the } i\text{th s-instance}\}$$

Again, the definition of ∇ is not affected by the behavior of the adversary, as all these tails are in the honest region. Further notice that in a given s-instance for each attack edge, we can have at most w tainted tails. Thus $|\nabla_i| \leq g \times w$ and $|\nabla| \leq rgw = O(rg \log n)$.

With slight abuse of notation, we say that a tail set *intersects* with a tail e as long as the tail set contains an element (i, e) for some i. The *number of intersections* with e is defined to be the number of elements of the form (i, e). We double count e in different instances because for every element (i, e), an arbitrary public key can be registered under the name of e in the ith s-instance. For two tail sets T_1 and T_2, we define the *number of intersections* between them as: $\sum_{(j,e)\in T_2}$ (# intersections between e and T_1). For example, $\{(1, e_1), (2, e_1)\}$ and $\{(2, e_1), (3, e_1)\}$ have 4 intersections. T_1 and T_2 *intersect* if and only if the number of intersection between them is larger than 0.

Tail intersection between the verifier and honest suspects. The *intersection condition* requires that for a verifier V to accept a suspect S, V's tail set and S's tail set must intersect with S being registered at some intersecting tail. We claim that for any given constant $\delta > 0$, a verifier V and an honest suspect S will satisfy the intersection condition with probability $1 - \delta$ when $r = r_0\sqrt{m}$, with r_0 being an appropriately chosen constant. This is true because with $1 - \frac{\delta}{2}$ probability, they will both have $(1 - O(\frac{g \log n}{n})) \cdot r = (1 - o(1))r > 0.5r$ uniform tails when $g = o(n/\log n)$. A straight-forward application of the Birthday Paradox will then complete the argument. Notice that we are not able to make arguments on the distribution of non-uniform tails and escaping tails, but uniform tails by themselves are sufficient for intersection to happen.

Tail intersection between the verifier and sybil suspects. By definition, all uniform tails of V are in the honest region. From the secure random route property, the tainted tail set ∇ contains all tails that the sybil nodes can possibly have in the honest region. We would like to bound the number of sybil nodes with (tainted) tails intersecting with V's uniform tails. V's non-uniform tails and escaping tails will be taken care of later by the balance condition.

Each tail in ∇ allows the adversary to potentially register a public key for some sybil node. The adversary has complete freedom on how to "allocate" these tails. For example, in one extreme, it may create $|\nabla|$ sybil nodes each with one tainted tail. In such a case, most likely not all these $|\nabla|$ sybil nodes will be accepted because each has only one tainted tail. In the other extreme, it can create one sybil node and register its public key with all tails in ∇.

We need to understand what is the adversary's optimal strategy for such an allocation. Interestingly, we can prove that regardless of what $\mathcal{U}(V)$ is, to maximize the number of sybil nodes with tails intersecting with $\mathcal{U}(V)$, the adversary should always create $|\nabla|$ sybil nodes and allocate one tail for each sybil node. To understand why, let random variable X be the number of intersections between ∇ and $\mathcal{U}(V)$. It is obviously impossible for more than X sybil nodes to have tails intersecting with $\mathcal{U}(V)$. On the other hand, with the previous strategy, the adversary can always create X sybil nodes with tails intersecting with $\mathcal{U}(V)$.

With this optimal strategy (of the adversary), we know that it suffices to focus on the probabilistic property of X. A tricky part in reasoning about X is that those tails in ∇ are neither uniformly random nor independent. For example, they are more likely to concentrate in the region near the attack edges. However, each tail in $\mathcal{U}(V)$ is still uniformly random. From linearity of expectation, we know that each tail in $\mathcal{U}(V)$ has on expectation $\frac{|\nabla|}{2m} = O(\frac{rg\log n}{m})$ intersections with ∇. This in turn means:

$$E[X] \leq r \cdot O(\frac{rg\log n}{m}) = O(g\log n), \text{ for any } r = O(\sqrt{m})$$

A Markov inequality [26] can then show that for any given constant $\delta > 0$, with probability at least $1-\delta$, X is $O(g\log n)$.

6.2. Balance condition

In this section, for any verifier V, we treat all of its non-uniform tails as escaping tails. Obviously, this only increases the adversary's power and makes our arguments pessimistic. The goal of the balance condition is to bound the number of sybil nodes accepted by V's escaping tails, without significantly hurting honest suspects (who are subject to the same balance condition). While the condition is simple, rigorously reasoning about it turns out to be quite tricky due to the external correlation among random routes and also adversarial disruption that may intentionally cause load imbalance. This introduces challenges particularly for proving why most honest suspects will satisfy the balance condition despite all these disruptions.

Effects on sybil suspects. We first study how the bar of $b = h \cdot \max(\log r, a)$ (Steps 5–7 in Figure 6) successfully bounds the number of sybil nodes accepted by V's escaping tails. The argument is complicated by the fact that when $a > \log r$, the bar b is a floating one. Namely, as more suspects are accepted, a and thus b will increase, allowing further suspects to be accepted. If all n honest suspects are accepted, the bar may rise to $\Theta(\frac{n}{r})$. We use such a floating bar because n is unknown (otherwise we could directly set the bar to be $\Theta(\frac{n}{r})$).

But on the other hand, it may also appear that as the escaping tails accept sybil nodes, the rising bar will allow further sybil nodes to be accepted. The key observation here is that, as shown by the previous section, the number of sybil nodes accepted by V's uniform tails is always properly bounded (by the intersection condition). The fraction of escaping tails is $o(1) < \frac{1}{h}$. Thus, if the load on all these escaping tails increases by some value x while the load on all uniform tails remain unchanged, the bar will only rise $o(1) \cdot x$. Following such argument, we will see that the amount by which the bar rises each time is upper bounded by a geometric sequence with a ratio of $o(1)$. The sum of this geometric sequence obviously converges, and in fact is dominated by the very first term in the sequence. This prevents undesirable cascading/unbounded rising of the bar. Our technical report [42] formally proves that under any constant h, V's escaping tails will accept only $O(g\log n)$ sybil nodes despite the floating bar.

Effects on honest suspects. Next, we briefly sketch our proof [42] that most non-escaping honest suspects will satisfy the balance condition for a sufficiently large constant h. We first consider the load on V's uniform tails. By definition, these tails are in the honest region. The load of a uniform tail may increase when it intersects with:

1. **Uniform tails of non-escaping honest suspects.**
2. **Non-uniform tails of non-escaping honest suspects.** For $g = o(n/\log n)$, a tail of a non-escaping node is non-uniform with $O(\frac{g\log n}{n}) = o(1)$ probability. Thus, with r s-instances and at most n non-escaping nodes, the expected number of such tails is $o(rn)$. By applying a Markov's inequality, we obtain that there are $o(rn)$ such tails with probability at least $1-\delta$ for any given constant $\delta > 0$.
3. **Uniform or non-uniform tails of escaping honest suspects.** By Lemma 4, there are at most ϵrn such tails, where ϵ is a constant that can be made close to 0.
4. **Tainted tails.** As explained in Section 6.1, there are $O(rg\log n) = o(rn)$ such tails for $g = o(n/\log n)$.

Considering first the load imposed by only the first type of tails in this list, we are able to prove [42] that with $1-\delta$ probability, most non-escaping suspects will satisfy both the intersection condition and the balance condition and thus will be accepted. This proof is fairly tricky/involved due to the external correlation among random routes. Harder still is taking into account the load imposed by the last 3 types of tails. In particular, the adversary has many different strategies for when to increase the load of which of V's tail, and finding the optimal strategy of the adversary is challenging. Fortunately, as argued above, the total number of tails from suspects in the last 3 tail types is $\epsilon' rn$ for some small ϵ'. We can apply a similar argument as in Section 6.1 to show

that with probability of $1-\delta$, the number of intersections between these $\epsilon' rn$ tails and $\mathcal{U}(V)$ is at most $\epsilon'' n$ for some small ϵ''. This means that the total load imposed in the last 3 tail types is at most $\epsilon'' n$. Finally, we prove that after doubling the constant h obtained earlier, even if the adversary completely controls where and when to impose the $\epsilon'' n$ load, the adversary can cause only $\epsilon'' n$ honest suspects to be rejected. Because ϵ'' can be made small and close to 0, this ensures that most non-escaping honest suspects will remain accepted.

7. Estimating the number of routes needed

We have shown that in SybilLimit, a verifier V will accept $(1-\epsilon)n$ honest suspects with probability $1-\delta$ if $r = r_0\sqrt{m}$. The constant r_0 can be directly calculated from the Birthday Paradox and the desired end probabilistic guarantees. On the other hand, m is unknown to individual nodes.[3] Adapting the sampling approach from SybilGuard (as reviewed in Section 4) is not possible, because that approach is fundamentally limited to $g = o(\sqrt{n}/\log n)$.

Benchmarking technique. SybilLimit uses a novel and perhaps counter-intuitive *benchmarking technique* to address the previous problem, by mixing the real suspects with some random *benchmark nodes* that are already known to be mostly honest. Every verifier V maintains two sets of suspects, the *benchmark set* K and the *test set* T. The *benchmark set* K is constructed by repeatedly performing random routes of length w and then adding the ending node (called the *benchmark node*) to K. Let K^+ and K^- be the set of honest and sybil suspects in K, respectively. SybilLimit does not know which nodes in K belong to K^+. But a key property here is that because the escaping probability of such random routes is $o(1)$, even without invoking SybilLimit, we are assured that $|K^-|/|K| = o(1)$. The *test set* T contains the real suspects that V wants to verify, which may or may not happen to belong to K. We similarly define T^+ and T^-. Our technique will hinge upon the adversary not knowing K^+ or T^+ (see later for how to ensure this), even though it may know $K^+ \cup T^+$ and $K^- \cup T^-$.

To estimate r, a verifier V starts from $r = 1$ and then repeatedly doubles r. For every r value, V verifies all suspects in K and T. It stops doubling r when most of the nodes in K (e.g., 95%) are accepted, and then makes a final determination for each suspect in T.

No over-estimation. Once r reaches $r_0\sqrt{m}$, most of the suspects in K^+ will indeed be accepted, regardless of the behavior of the adversary. Further, because $|K^+|/|K| = 1-o(1)$, having an r of $r_0\sqrt{m}$ will enable us to reach the threshold (e.g., 95%) and stop doubling r further. Thus, V will never over-estimate r (within a factor of 2).

Under-estimation will not compromise SybilLimit's guarantees. It is possible for the adversary to cause an under-estimation of r by introducing artificial intersections between the escaping tails of V and the escaping tails of suspects in K^+. This may cause the threshold to be reached before r reaches $r_0\sqrt{m}$.

What if SybilLimit operates under an $r < r_0\sqrt{m}$? Interestingly, SybilLimit can bound the number of sybil nodes accepted within $O(\log n)$ per attack edge not only when $r = r_0\sqrt{m}$, but also for $r < r_0\sqrt{m}$ (see [42] for proofs). To obtain some intuition, first notice that the number of sybil nodes with tails intersecting with V's uniform tails (Section 6.1) can only decrease when r is smaller. Second, the arguments regarding the number of sybil nodes accepted by V's escaping tails and non-uniform tails (Section 6.2) hinges only upon the *fraction* of those tails, and not the value of r.

Using $r < r_0\sqrt{m}$, however, will decrease the probability of tail intersection between the verifier and an honest suspect. Here, we leverage a second important property of the benchmark set. Namely, conditioned upon the random routes for picking benchmark nodes being non-escaping, the adversary will not know which nodes are picked as benchmark nodes. (If the adversary may eavesdrop messages, we can readily encrypt messages using edge keys.) As a result, given an honest suspect, the adversary cannot tell whether it belongs to K^+ or T^+. If most (e.g., 95%) of the suspects in K are accepted, then most suspects in K^+ must be accepted as well, since $|K^+|/|K| = 1-o(1)$. If most suspects in K^+ are accepted under $r < r_0\sqrt{m}$, the adversary must have intentionally caused intersection between V and the suspects in K^+. Because the adversary cannot tell whether an honest suspect belongs to K^+ or T^+, it cannot introduce intersections *only* for suspects in K^+; it must introduce intersections for suspects in T^+ as well. Thus, most suspects in T^+ will be accepted as well under the given r.

Further discussions. The benchmarking technique may appear counter-intuitive in two aspects. First, if SybilLimit uses an under-estimated r, it will be the adversary that helps it to accept most of the honest nodes. While this is true, SybilLimit is still needed to bound the number of sybil nodes accepted and also to prevent r from growing beyond $r_0\sqrt{m}$. Second, the benchmark set K is itself a set with $o(1)$ fraction of sybil nodes. Thus, it may appear that an application can just as well use the nodes in K directly, and avoid the full SybilLimit protocol. However, the set K is constructed randomly and may not contain some specific suspects that V wants to verify.

For a more rigorous understanding of the benchmarking

[3] SybilLimit also requires that the random route length w be the mixing time of the graph, which is also unknown. However, as in SybilGuard [43], SybilLimit assumes that the nodes know a rough upper bound on the graph's mixing time. Such an assumption is reasonable because the mixing time should be $O(\log n)$, which is rather insensitive to n.

1. V starts with two sets of suspects, K and T;
2. Let set $A = \emptyset$ and $r = 1$;
3. While ($|A \cap K|/|K| < 95\%$) {
4. For every suspect $S \in ((K \cup T) \setminus A)$, verify S using the protocol in Figure 6;
5. If S is accepted, $A = A \cup \{S\}$;
6. Double r;
   }
7. V accepts all suspects in $A \cap T$, and rejects all suspects in $T \setminus A$;

Figure 8. Pseudo-code for the benchmarking technique.

technique, we can view the process as a sampling algorithm for estimating the fraction of the suspects accepted in the set $T^+ \cup K^+$. We take $|K^+|$ samples from the set and observe that fraction f of the samples are accepted. Classic estimation theory [4] tells us that if $|K^+| = \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$, then the fraction of the accepted suspects in T^+ is within $f \pm \epsilon$ with probability of at least $1 - \delta$. It is important to see that the needed size of K^+ (and thus K) is independent of the size of T. Simple simulation experiments show that $|K| = 30$ gives us an average ϵ of 0.0322.

Care must be taken when implementing the benchmarking technique. The technique hinges on the fact that the adversary cannot distinguish suspects in K^+ from suspects in T^+. A naive implementation would gradually increase r and invoke the verification protocol from Figure 6 multiple times (under different r) for each suspect. This will leak (probabilistic) information to the adversary. Namely, if the adversary notices that V still increases r even after a certain honest suspect S is accepted, then the conditional probability that S belongs to T^+ increases. Under the increased r, the adversary may then favor other suspects in $K^+ \cup T^+$ and cause S to be rejected. This will then violate the assumption that K^+ is a uniform sample of $K^+ \cup T^+$.

To ensure that K^+ is a uniform sample of $K^+ \cup T^+$, we automatically consider a suspect S that is accepted under a certain r to be accepted under larger r values, without re-verifying this. Figure 8 presents the pseudo-code, which maintains a set A including all suspects accepted so far. Now imagine that the adversary notices that V still increases r despite those suspects in A being accepted. This tells the adversary that the suspects in A are less likely to belong to K^+ than those suspects not in A. However, the adversary can no longer reverse the determinations already made for those suspects in A. The adversary can still influence future determinations on those suspects not in A. But all these suspects have the same probability of being in K^+. So it does not help the adversary to favor some over others.

8. Lower bound

SybilLimit bounds the number of sybil nodes accepted within $O(\log n)$ per attack edge. A natural question is whether we can further improve the guarantees. For example, it may appear that SybilLimit does not currently have any mechanism to limit the routing behavior of sybil nodes. One could imagine requiring nodes to commit (cryptographically) to their routing tables, so that sybil nodes could not perform random routes in an inconsistent fashion. We will show, however, that such techniques or similar techniques can provide at most a $\log n$ factor of improvement, because the total number of sybil nodes accepted is lower bounded by $\Omega(1)$ per attack edge.

SybilLimit entirely relies on the observation that if the adversary creates too many sybil nodes, then the resulting social network will no longer have $O(\log n)$ mixing time. Our technical report [42] proves that for any given constant c, any $g \in [1, n]$, and any graph G with n honest nodes and $O(\log n)$ mixing time, it is always possible for the adversary to introduce $c \cdot g$ sybil nodes via g attack edges so that the augmented graph's mixing time is $O(\log n')$ where $n' = n + c \cdot g$. There are actually many ways to create such an augmented graph. One way (as in our proof) is to pick g nodes arbitrarily from G and attach to each of them (using a single attack edge) a group of c sybil nodes. It does not matter how the c sybil nodes in a group are connected with each other, as long as they are connected. Now because the augmented graph has the same mixing time (i.e., $O(\log n')$) as a "normal" social network with n' nodes, as long as the protocol solely relies on mixing time, we cannot distinguish these sybil nodes from honest nodes. In other words, all protocols based on mixing time will end up accepting $\Omega(1)$ sybil nodes per attack edge.

9. Experiments with online social networks

Goal of experiments. We have proved that SybilLimit can bound the number of sybil nodes accepted within $O(\log n)$ per attack edge, which improved upon SybilGuard's guarantee of $O(\sqrt{n} \log n)$. However, these provable guarantees of SybilLimit (and SybilGuard as well) critically rely on the assumption that social networks have small (i.e., $O(\log n)$) mixing time. Our experiments thus mainly serve to validate such an assumption, based on real-world social networks. Such validation has a more general implication beyond SybilLimit—these results will tell us whether the approach of leveraging social networks to combat sybil attacks is valid. A second goal of our experiments is to gain better understanding of the hidden constant in SybilLimit's $O(\log n)$ guarantee. Finally, we will also provide some example numerical comparisons between SybilGuard and

SybilLimit. However, it is *not* our goal to perform a detailed experimental comparison, because SybilLimit's improvement over SybilGuard is already rigorously proved.

Social network data sets. We use three crawled online social network data sets in our experiments: Friendster, LiveJournal, and DBLP (Table 3). They are crawls of `http://www.friendster.com`, `http://www.livejournal.com`, and `http://dblp.uni-trier.de`, respectively. The DBLP data set is publicly available, but the other two are not. We also experiment with Kleinberg's synthetic social network [17], which we used [43] to evaluate SybilGuard.

Strictly speaking, DBLP is a bibliography database and not a social network. To derive the "social network" from DBLP, we consider two people having an edge between them if they have ever co-authored a paper. Because of the closely clustered co-authoring relationships among researchers, we expect such a social network to be more slowly mixing than standard social networks. Thus, we use DBLP as a bound on the worst-case scenario. Obviously, DBLP is guaranteed to be free of sybil nodes. Although it is theoretically possible for Friendster and LiveJournal to be polluted with sybil nodes already, we expect such pollution to be limited because of the lack of motivation to launch large-scale sybil attacks in Friendster and LiveJournal.

Preprocessing of data sets. We preprocess Friendster, LiveJournal, and DBLP in the following way before using them. (Kleinberg does not need preprocessing.) First, the original Friendster and LiveJournal data sets have directed edges between users instead of undirected edges, while SybilLimit operates on an undirected graph. During the crawl to obtain Friendster and LiveJournal, a directed edge $A \rightarrow B$ is added to the graph if A lists B as its friend. For LiveJournal, we consider that there is an undirected edge between A and B if and only if there are two directed edges $A \rightarrow B$ and $B \rightarrow A$ in the original data. For Friendster, we find on `www.friendster.com` that if A lists B as A's friend, then B must also lists A as B's friend. In other words, the friendship relation is always mutual. This does not necessarily mean that the original Friendster data set must contain both $A \rightarrow B$ and $B \rightarrow A$, since it is possible that the crawl stops after crawling A's friend list but before crawling B's friend list. Thus for Friendster, if there is a directed edge $A \rightarrow B$, or $B \rightarrow A$, or both, we consider that there is an undirected edge between A and B.

The second step of our preprocessing limits the degree of all nodes in the graph to be 100 or fewer. SybilLimit inherits the idea from SybilGuard that an honest node should not have an excessive number of neighbors. This restriction helps bound the number of additional attack edges the adversary gets when an honest node is compromised. We pick the limit 100 because this appears to be reasonable in the real world: a typical human being is likely to have strong social relationships with fewer than 100 people. We limit the degree of the nodes by removing random edges from a node if its degree is above 100.

Next we remove all nodes in the graph with degree smaller than 5. This decision reflects our expectation that a new user of SybilLimit will establish a minimum number of edges (e.g., 5) with existing users. This requirement ensures that the new user has at least some reasonable connectivity to the social network. Before the user establishes these edges, the user can still use other nodes as proxies to verify suspects, except that it cannot be verified by other nodes. The final preprocessing step is to select the largest connected component in the resulting graph. This largest connected component is what we use in our experiments. Table 3 presents the basic statistics of the four social networks after preprocessing (if needed).

Experimental methodology. We choose to use simulation in all of our experiments for two important reasons. First, we are mainly concerned with how the graph properties of these real-world social networks affect SybilLimit's end security guarantees. Performance is not the focus of our evaluation; as explained earlier, it is unlikely for SybilLimit to incur excessive performance overheads. Second, simulation allows us to study large (i.e., million-node) social networks. All results are obtained after the secure random route protocol in SybilLimit has stabilized.

Exactly as in [43], we place the g attack edges in the social networks in two different ways. In `rand`, we repeatedly pick uniformly random nodes in the graph and *mark* them. In `cluster`, we start from a uniformly random node and then perform a breadth-first search from that node. All nodes encountered are *marked*. In both `rand` and `cluster`, those edges between marked nodes and unmarked nodes are considered attack edges. We keep marking nodes until the total number of attack edges reaches our target g. We find that the results using `cluster` placement are usually slightly better than using `rand`, under the same g. Thus, all results presented below are the pessimistic results using `rand`.

Results: Mixing time of real-world social networks. In SybilLimit, the only parameter affected by mixing time is the length of the random routes (w). Namely, w should be at least as large as the mixing time. It is not possible to directly show that our data sets have $O(\log n)$ mixing time, since $O(\log n)$ is asymptotic behavior. It is not necessary to do so either, since all we need to confirm is that rather small w values are already sufficient for SybilLimit to work well.

For Friendster and LiveJournal, we use $w = 10$ (see Table 3). Random routes do not seem to reach good enough mixing for SybilLimit with w values much smaller than 10 (e.g., 5) in these two social networks. We use $w = 15$ for DBLP. As expected, DBLP has a worse mixing property than the other social networks. Our results will show that

Data set	Friendster	LiveJournal	DBLP	Kleinberg
Data set source	[34]	[39]	[10]	[17]
Date crawled	Nov-Dec 2005	May 2005	April 2006	not applicable
# nodes	932,512	900,822	106,002	1,000,000
# undirected edges	7,835,974	8,737,636	625,932	10,935,294
w used in SybilLimit	10	10	15	10
r used in SybilLimit	8,000	12,000	3,000	10,000

Table 3. Social network data sets.

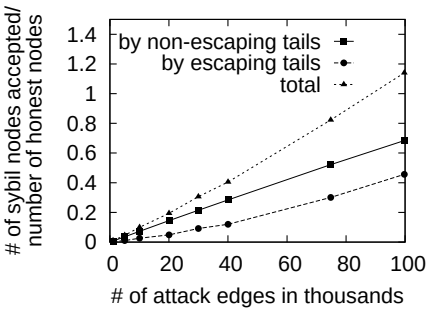

Figure 9. Friendster

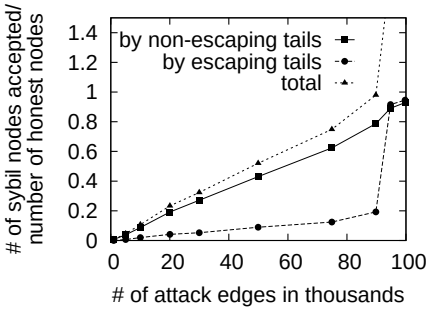

Figure 10. LiveJournal

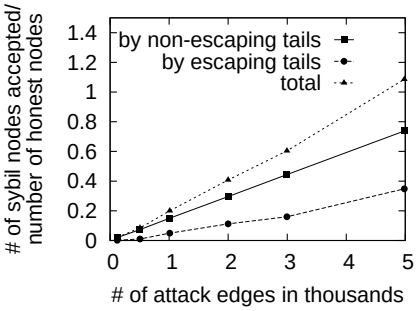

Figure 11. DBLP

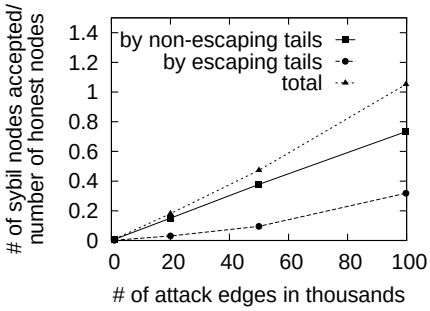

Figure 12. Kleinberg

these small w values are already sufficient to enable good enough mixing in our large-scale social networks (with 10^5 to around 10^6 nodes) for SybilLimit to work well.

It is worth noting that social networks are well-known to have groups or communities where intra-group edges are much denser than inter-group edges [3, 15, 24, 38]. In fact, there are explicitly-defined communities in LiveJournal for users to join, while people in DBLP by definition form research communities. Our results thus show that somewhat counter-intuitively and despite such groups, the sparse inter-group edges in these real-world social networks are sufficient to provide good mixing properties.

Results: SybilLimit's end guarantees. We use the w values from Table 3 to simulate SybilLimit and determine the number of sybil nodes accepted. Our simulator does not implement the estimation process for r. Rather, we directly use the r values from Table 3, which are obtained based on the value of m and the Birthday Paradox. We use 4 for the universal constant h in all our experiments. We have observed (results not included) that $h = 2.5$ is already sufficient in most cases, while excessively large h (e.g., 10) can unnecessarily weaken the guarantees (though not asymptotically). We always simulate the adversary's optimal strategy (i.e., worst-case for SybilLimit).

Figures 9 to 12 present the number of sybil nodes accepted by a randomly chosen verifier V (as a fraction of the number of honest nodes n), in each social network. We present a fraction to allow comparison across social networks with different n. We have repeated the experiments from a number of verifiers, yielding similar results. For all cases, we experiment with g up to the point where the number of sybil nodes accepted reaches n. The figures further break down the sybil nodes accepted into those accepted by V's non-escaping tails versus those accepted by V's escaping tails. The first component is bounded by the intersection condition while the second is bounded by the balance condition. In all figures, the number of sybil nodes accepted grows roughly linearly with g. The asymptotic guarantee of SybilLimit is $O(\log n)$ sybil nodes accepted per attack edge. Figures 9 to 12 show that this $O(\log n)$ asymptotic term translates to around between 10 (in Friendster, LiveJournal, and Kleinberg) to 20 (in DBLP). As a concrete numerical comparison with SybilGuard, SybilGuard [43] uses random routes of length $l = 1906$ in the million-node Kleinberg graph. Because SybilGuard accepts l sybil nodes per attack edge, this translates to 1906 sybil nodes accepted per attack edge for Kleinberg. Thus numerically in Kleinberg, SybilLimit reduces the number of sybil nodes accepted by nearly 200-fold over SybilGuard.

One can also view Figures 9 to 12 from another perspective. The three data sets Friendster, LiveJournal, and Kleinberg all have roughly one million nodes. Therefore, in order for the number of sybil nodes accepted to reach n, the number of attack edges needs to be around 100,000. Put it another way, the adversary needs to establish 100,000 social trust relationships with honest users in the system. As a quick comparison under Kleinberg, SybilGuard will accept n sybil nodes once g reaches around 500 (since

$l = 1906$). Some simple experiments further show that with $g \geq 15,000$, the escaping probability of the random routes in SybilGuard will be above 0.5 and SybilGuard can no longer provide any guarantees at all. Finally, DBLP is much smaller (with 100,000 nodes) and because of the slightly larger w needed for DBLP, the number of sybil nodes accepted will reach n roughly when g is 5,000.

Finally, we have also performed experiments to investigate SybilLimit's guarantees on much smaller social networks with only 100 nodes. To do so, we extract 100-node subgraphs from our social network data sets. As a concise summary, we observe that the number of sybil nodes accepted per attack edge is still around 10 to 20.

10. Conclusion

This paper presented SybilLimit, a near-optimal defense against sybil attacks using social networks. Compared to our previous SybilGuard protocol [43] that accepted $O(\sqrt{n}\log n)$ sybil nodes per attack edge, SybilLimit accepts only $O(\log n)$ sybil nodes per attack edge. Furthermore, SybilLimit provides this guarantee even when the number of attack edges grows to $o(n/\log n)$. SybilLimit's improvement derives from the combination of multiple novel techniques: i) leveraging multiple independent instances of the random route protocol to perform many short random routes, ii) exploiting intersections on edges instead of nodes, iii) using the novel balance condition to deal with escaping tails of the verifier, and iv) using the novel benchmarking technique to safely estimate r. Finally, our results on real-world social networks confirmed their fast mixing property, and thus validated the fundamental assumption behind SybilLimit's (and SybilGuard's) approach. As future work, we intend to implement SybilLimit within the context of some real-world applications and demonstrate its utility.

Acknowledgments

We thank Jelle Roozenburg for allowing us to use his Friendster data set and Rita Wouhaybi for allowing us to use her LiveJournal data set. We thank Chris Lesniewski-Laas and the anonymous reviewers for many helpful comments on the paper. This work is partly supported by NUS grants R-252-050-284-101 and R-252-050-284-133.

References

[1] I. Abraham and D. Malkhi. Probabilistic quorums for dynamic systems. In *DISC*, 2003.

[2] T. Anderson. SIGCOMM'06 public review on 'SybilGuard: Defending against sybil attacks via social networks'. `http://www.sigcomm.org/sigcomm2006/discussion/`, 2006.

[3] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *ACM KDD*, 2006.

[4] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. In *ACM STOC*, 2001.

[5] R. Bazzi and G. Konjevod. On the establishment of distinct identities in overlay networks. In *ACM PODC*, 2005.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *IEEE INFOCOM*, 2005.

[7] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *USENIX OSDI*, 2002.

[8] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *ACM P2PEcon*, 2005.

[9] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-resistant DHT routing. In *ESORICS*, 2005. Springer-Verlag LNCS 3679.

[10] `http://kdl.cs.umass.edu/data/dblp/dblp-info.html`.

[11] J. Douceur. The Sybil attack. In *IPTPS*, 2002.

[12] E-Mule. `http://www.emule-project.net`.

[13] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM Electronic Commerce*, 2004.

[14] A. Flaxman. Expansion and lack thereof in randomly perturbed graphs. Technical report, Microsoft Research, 2006. `ftp://ftp.research.microsoft.com/pub/tr/TR-2006-118.pdf`.

[15] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12), 2002.

[16] V. Guruswami. Rapidly mixing markov chains: A comparison of techniques. Technical report, MIT Laboratory for Computer Science, May 2000. Available at `http://www.cs.washington.edu/homes/venkat/pubs/papers/markov-survey.ps`.

[17] J. Kleinberg. The small-world phenomenon: An algorithm perspective. In *ACM STOC*, 2000.

[18] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM TOPLAS*, 4(3), 1982.

[19] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the Maze P2P file-sharing system. In *IEEE ICDCS*, 2007.

[20] T. Lindvall. *Lectures on the Coupling Method*. Dover Publications, 2002.

[21] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker. The LOCKSS peer-to-peer digital preservation system. *ACM TOCS*, 23(1), 2005.

[22] N. B. Margolin and B. N. Levine. Quantifying and discouraging sybil attacks. Technical report, U. Mass. Amherst, Computer Science, 2005.

[23] N. B. Margolin and B. N. Levine. Informant: Detecting sybils using incentives. In *Financial Cryptography*, 2007.

[24] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *ACM/USENIX IMC*, 2007.

[25] A. Mislove, A. Post, K. Gummadi, and P. Druschel. Ostra: Leveraging trust to thwart unwanted communication. In *USENIX NSDI*, 2008.

[26] M. Mitzenmacher and E. Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[27] R. Morselli, B. Bhattacharjee, A. Srinivasan, and M. Marsh. Efficient lookup on unstructured topologies. In *ACM PODC*, 2005.

[28] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: Analysis & defenses. In *ACM/IEEE IPSN*, 2004.

[29] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *IEEE INFOCOM*, 2002.

[30] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE S & P*, 2005.

[31] V. Prakash. Razor. `http://razor.sourceforge.net`.

[32] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *ACM SIGCOMM*, 2006.

[33] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *SWSA ISWC*, 2003.

[34] J. Roozenburg. A literature survey on bloom filters. Unpublished Research Assignment, Delft Univ. of Technology, NL, 2005.

[35] M. Steiner, T. En-Najjary, and E. W. Biersack. Exploiting KAD: Possible uses and misuses. *ACM SIGCOMM CCR*, 37(5), 2007.

[36] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Telling humans and computers apart. In *IACR Eurocrypt*, 2003.

[37] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *USENIX NSDI*, 2006.

[38] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, Cambridge, 1994.

[39] R. H. Wouhaybi. Trends and behavior in online social communities. Unpublished Research, Intel Corporation, Hillsboro, OR, USA, 2007.

[40] M. Yang, Z. Zhang, X. Li, and Y. Dai. An empirical study of free-riding behavior in the Maze P2P file-sharing system. In *IPTPS*, 2005.

[41] H. Yu, P. B. Gibbons, and M. Kaminsky. Brief announcement: Toward an optimal social network defense against sybil attacks. In *ACM PODC*, 2007.

[42] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. Technical Report TRA2/08, National Univ. of Singapore, School of Computing, Mar. 2008. `http://www.comp.nus.edu.sg/~yuhf/sybillimit-tr.pdf`.

[43] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against sybil attacks via social networks. In *ACM SIGCOMM*, 2006.

A. Proofs

The proofs in this appendix establish the asymptotic guarantees of SybilLimit. For all constants used or derived in the proofs, we aim for simplicity instead of pursuing the optimal. All results are for n sufficiently large, and assuming that the honest region of the social network has $O(\log n)$ mixing time.

A.1. Preliminaries: Classifying Tails and Nodes

Lemma 5 *Consider any given constant $\epsilon > 0$. We can always find a universal constant $w_0 > 0$, such that there exists a set H of at least $(1-\epsilon)n$ honest nodes (called* non-escaping *nodes) where if we perform a length-w random walk starting from any non-escaping node with $w = w_0 \log n$, then*

- *The tail is non-escaping with probability of at least $1 - O(\frac{g \log n}{n})$.*
- *The tail is a uniformly random directed edge in the honest region of the social network with probability of at least $1 - O(\frac{g \log n}{n})$.*

Proof: We let G denote the entire social network with all honest nodes and sybil nodes. The sybil nodes in G may deviate from the protocol in arbitrary way. We define G' to be the social network with only honest nodes and edges between honest nodes. G' is not known by SybilLimit. According to our system model (Section 3), G' has n nodes and m undirected edges. We will later draw connections between random walks in G to random walks in G'.

To prove the lemma, it suffices to show that we can find two universal constants $c_1 > 0$ and $c_2 > 0$ such that the probability of the tail being non-escaping is at least $1 - c_1 \frac{g \log n}{n}$ and the probability of the tail being uniformly random is at least $1 - c_2 \frac{g \log n}{n}$. According to our assumption, G' is fast mixing with $O(\log n)$ mixing time. This means that we can find a universal constant w_0 such that random walks of length $w = w_0 \log n$ in G' is sufficient to achieve a variation distance of $\frac{1}{n}$ or lower. We let $c_1 = w_0/\epsilon_1$ and $c_2 = 2c_1 + 1$.

We first intend to find a set of $(1-\epsilon_1)n$ nodes such that starting from any of them, a length-w random walk in G is non-escaping with probability of at least $1 - \frac{gw}{\epsilon_1 n}$. Let p_i be the probability of a length-w random walk being escaping if we start the random walk from honest node i, for $1 \leq i \leq n$. It has been proved [43] that $p_1 + p_2 + \ldots + p_n \leq gw$. Among all these p_i's, We claim that there must be at least $(1-\epsilon_1)n$ of them that are at most $\frac{gw}{\epsilon_1 n}$. This is true because otherwise there must be $\epsilon_1 n$ values that are larger than $\frac{gw}{\epsilon_1 n}$, which will make the summation larger than gw. Without loss of generality, we can thus assume $p_i \leq \frac{gw}{\epsilon_1 n}$ for $1 \leq i \leq (1-\epsilon)n$. The set H is then constructed as the set containing node 1 through node $(1-\epsilon)n$. Obvious $|H| \geq (1-\epsilon_1)n$ and the probability of a length-w random walk starting from any node in H being escaping is at most $\frac{gw}{\epsilon_1 n} = c_1 \frac{g \log n}{n}$.

Consider any node $V \in H$, and we will draw a connection between the random walk starting from V in G and the random walk starting from V in G'. In G', let a'_i be the probability of V's tail being directed edge i for $1 \leq i \leq 2m$. Given how we picked w_0 earlier, we know that the distribution a'_i has a variation distance of at most $\frac{1}{n}$ from the stationary distribution (i.e., the uniform distribution on the directed edges).

Now consider the graph G. For now let us assume that the adversary never allows any escaping random walk to return to the honest region. Let a_i be the probability of the tail being directed edge i for $1 \leq i \leq 2m$. Notice that we will have $\sum_{i=1} 2ma_i < 1$ by definition. It is not difficult to see that $a_i \leq a'_i$ for all i and $\sum_{i=1}^{2m}(a'_i - a_i) \leq c_1 g \log n/n$. Now if the adversary does direct some escaping random walks back to the honest region, the probabilities may increase from a_i to b_i. However, we must also have $\sum_{i=1}^{2m}(b_i - a_i) \leq c_1 g \log n/n$. Notice that there can still be random walks that do not return to the honest region. We use $b_{2m+1} \leq c_1 g \log n/n$ to denote such probability—this will make $\sum_{i=1}^{2m+1} b_i = 1$.

We would like to eventually reason about the variation distance between the distribution of b_i and the uniform distribution. To do so, we define $a'_{2m+1} = 0$. The variation distance between a'_i and b_i is:

$$\begin{aligned} 0.5 \sum_{i=1}^{2m+1} |a'_i - b_i| &\leq 0.5(c_1 \frac{g \log n}{n} + \sum_{i=1}^{2m} |a'_i - b_i|) \\ &\leq 0.5(c_1 \frac{g \log n}{n} + \sum_{i=1}^{2m} (|a'_i - a_i| + |a_i - b_i|)) \\ &< 2c_1 \frac{g \log n}{n} \end{aligned}$$

Finally, because the distribution a_i' has a variation distance of at most $\frac{1}{n}$ from the uniform distribution, the variation distance between b_i and the uniform distribution is at most $2c_1\frac{g\log n}{n} + \frac{1}{n} < c_2\frac{g\log n}{n}$. From the property of variation distance [20], we know that if we pick an edge from distribution b_i, with at least probability of $1 - c_2\frac{g\log n}{n}$, the edge is a uniformly random directed edge in the honest region. □

Comment. It is important to notice that the honest region of the social network, together with the location of the attack edges, uniquely determine which nodes are escaping and non-escaping. In other words, this is not affected by the routing tables.

Lemma 6 *Consider any given non-escaping verifier V (or suspect S) and its r tails in the r v-instances (or s-instances). For any given constant $\delta > 0$, with probability of at least $1 - \delta$:*

- *The number of escaping tails and non-uniform tails is $O(\frac{g\log n}{n}) \cdot r$.*
- *The number of uniform tails is $(1 - O(\frac{g\log n}{n})) \cdot r$.*

Proof: The second claim directly follows from the first claim. Let X be the total number of escaping and non-uniform tails in all r instances. To prove the first claim, it suffices to show that there exists some universal constant c_1, $Pr[X > c_1\frac{g\log n}{n} \cdot r] \leq \delta$. Lemma 5 tells us that in any given instance, the probability of V's tail being escaping or non-uniform is at most $O(\frac{g\log n}{n}) < c_2\frac{g\log n}{n}$ for some universal constant c_2. Let $c_1 = c_2/\delta$. We thus have $E[X] < c_2\frac{g\log n}{n}r = \delta c_1\frac{g\log n}{n}r$. Invoking a Markov inequality on X will yield that $Pr[X > c_1\frac{g\log n}{n}r] \leq \delta$. □

Comment. We use Markov inequality instead of a Chernoff bound in the proof because we want the result to hold even for small r values.

A.2. Why the Number of Accepted Sybil Nodes Is Properly Bounded

We would like to prove these result for any $r \leq r_0\sqrt{m}$, instead of only for $r = r_0\sqrt{m}$. This will allow us to use the benchmarking technique described in Section 7 to estimate r.

Lemma 7 *Consider any given constant $r_0 > 0$, $\delta > 0$, and any given honest verifier V. Let w_0 be from Lemma 5, $w = w_0\log n$, and $r \leq r_0\sqrt{m}$. Then with probability of at least $1 - \delta$, the number of sybil nodes with tails intersecting with $\mathcal{U}(V)$ is $O(g\log n)$.*

Proof: Let X denote the number of intersections between $\mathcal{U}(V)$ and the tainted tail set ∇. It suffices to show that there exists some universal constant c_1, such that $Pr[X > c_1 g\log n] \leq \delta$.

Consider any tail in $\mathcal{U}(V)$. In each s-instance, there are at most gw tainted tails. Because the tail from $\mathcal{U}(V)$ is a uniformly random edge, the probability of it intersecting with those tainted tails in the given s-instance is at most $gw/(2m)$. With total r s-instances, the expected number of intersections will be at most $rgw/(2m)$. Finally, we trivially have $|\mathcal{U}(V)| \leq r$ and thus $E[X] \leq r^2gw/(2m) = r_0^2w_0g\log n/2$. Let $c_1 = r_0^2w_0/(2\delta)$ and then invoke a Markov inequality on X. We will then have $Pr[X > c_1 g\log n] \leq \delta$. □

Comment. Notice that Lemma 7 holds for both non-escaping verifiers and escaping verifiers.

Lemma 8 *Consider any given constant $r_0 > 0$, $h > 0$, $\delta > 0$, and any given honest verifier V. Let w_0 be from Lemma 5, $w = w_0\log n$, and $r \leq r_0\sqrt{m}$. Assume that $g = o(\log n/n)$. Then with probability of at least $1 - 2\delta$, the number of sybil nodes accepted by V's non-uniform tails and escaping tails is $O(g\log n)$.*

Proof: Define random variable Q to be the number of non-uniform tails and escaping tails that V has. From the balance condition, we know that the number of sybil nodes accepted by V's non-uniform tails and escaping tails is at most $Q \cdot h \cdot max(\log r, a)$, where a is the average number of suspects accepted by all tails of the verifier. Lemma 6 tells us that with probability of at least $1 - \delta$, Q is $O(\frac{g\log n}{n}) \cdot r$. Conditioned upon that $Q = O(\frac{g\log n}{n}) \cdot r$, we consider two cases:

- $a \leq \log r$: We have $Q \cdot h \cdot max(\log r, a) = Qh\log r = O((g\log n)\frac{r\log r}{n}) = O((g\log n)\frac{r_0\sqrt{m}\log(r_0\sqrt{m})}{n}) = O(g\log n)$ (assuming $\sqrt{m}\log m = O(n)$).

- $a > \log r$: Let X be the number of suspects accepted by V's non-uniform tails and escaping tails. Let Y be the number of suspects accepted by V's uniform tails. There are n honest suspects that can be accepted. Furthermore, Lemma 7 tells us that with probability of at least $1-\delta$, the number of sybil nodes accepted by these uniform tails is $O(g \log n) = o(n) < n$ when $g = o(n/\log n)$. Thus with probability of at least $1-\delta$, $Y \leq 2n$. The load on the non-uniform tails and escaping tails must satisfy the balance condition, which implies:

$$\begin{aligned} & \frac{X}{Q} \leq h \cdot \frac{X+Y}{r} \leq h \cdot \frac{X+2n}{r} \\ \Rightarrow \quad & \frac{X}{O(\frac{g \log n}{n}) \cdot r} \leq h \cdot \frac{X+2n}{r} \\ \Rightarrow \quad & X \leq (X+2n) \cdot O(\frac{g \log n}{n}) \\ \Rightarrow \quad & X \leq O(g \log n)/(1 - O(\frac{g \log n}{n})) = O(g \log n)/(1-o(1)) = O(g \log n) \end{aligned}$$

□

A.3. Why Honest Suspects will Satisfy the Intersection Condition

Lemma 9 *Assume that $g = o(\log n/n)$ and consider any give constant $\delta > 0$. We can always find a universal constant $r_0 \geq 1$ such that if $r = r_0\sqrt{m}$, then for any given non-escaping verifier V and any given non-escaping suspect S, $\mathcal{U}(V)$ and $\mathcal{U}(S)$ intersect with probability of at least $1-\delta$.*

Proof: Directly follows from Lemma 6 and the Birthday Paradox. □

A.4. Why Honest Suspects Will Satisfy the Balance Condition

In this section, we prove why most honest suspects will satisfy the balance condition. Together with Lemma 9, this will complete the proof for SybilLimit's guarantee on the fraction of honest nodes accepted. The proof in this section turns out to be the most tricky among all our proofs. For better explanation, we will construct our proofs based on connections among the following 4 cases on how the load of the verifier's tails are incremented. In all cases, we still require the load on every tail to be no larger than $h \cdot max(\log r, a)$.

$\mathcal{A}$ Every accepted non-escaping honest suspect S increments the load of every tail in $\mathcal{U}(V)$ that intersects with S's uniform tails. The load of V's tails does not increase in other cases. In other words, i) no additional load is imposed on tails in $\mathcal{U}(V)$, and ii) those tails not in $\mathcal{U}(V)$ (i.e., non-uniform tails and escaping tails) always have a load of 0.

$\mathcal{B}$ Every accepted non-escaping honest suspect S increments the load of one tail, out of all tails in $\mathcal{U}(V)$ that intersect with S's uniform tails. We allow the adversary to determine the tail picked. The load of V's tails does not increase in other cases.

$\mathcal{C}$ Every accepted non-escaping honest suspect S increments the load of the least loaded tail (with tie breaking based on instance number), out of all tails in $\mathcal{U}(V)$ that intersect with S's uniform tails. The load of V's tails does not increase in other cases.

$\mathcal{D}$ Every accepted non-escaping honest suspect S increments the load of the least loaded tail (with tie breaking based on instance number), among all tails in $\mathcal{U}(V)$ that intersect with S's tails (notice that it is "S's tails" instead of "S's uniform tails"). Additionally, for tails in $\mathcal{U}(V)$, the adversary may increase their load arbitrarily at any point of time, subject to the condition that the total increase load (for all tails in $\mathcal{U}(V)$ combined) is ϵn for some small $\epsilon > 0$. For tails not in $\mathcal{U}(V)$, the adversary may increase their load arbitrarily at any point of time with no restrictions.

The last case $\mathcal{D}$ captures SybilLimit's behavior (pessimistically). The first three cases are not "implementable", but they serve as stepping stones in our proofs and arguments.

A.4.1 Proofs for $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$

Lemma 10 *Suppose $g = o(n/\log n)$. In $\mathcal{A}$, consider any given set of k honest non-escaping suspects and any given directed edge in the honest region of the social network. Let Z denote the number of intersections that edge has with all the uniform tail sets of the k suspects. Then with probability of at least $1 - 2^{-r/2}$, Z is lower bounded by some binomial distribution with mean of $rk/(16m)$ and upper bounded by some other binomial distribution with mean of $rk/(2m)$.*

Proof: Consider the ith s-instance and let random variable T_i to be the number of uniform tails that the k suspects have in that s-instance. Each of these T_i tails is a uniformly random edge, but they are potentially correlated. Despite such correlation, let X_j (for $1 \leq j \leq T_i$) be indicator random variable denoting th event that the (X_j)th tail intersect with the given directed edge. Further define X to be the event that some tail from the T_i tails intersect with the given directed edge. Because of the backtracability of the random routes within any given s-instance, at most one of all these X_i's can be 1. Thus we have $X = X_1 + X_2 + ...X_{T_i}$. From linearity of expectation, we know that $E[X] = E[X_1] + E[X_2] + ... + E[X_{T_i}] = \frac{T_i}{2m}$. On the other hand, because X is an indicator random variable, $Pr[X] = E[X] = \frac{T_i}{2m}$.

Next we would like to study the distribution of T_i. Define indicator random variable Y_j to denote the event that the tail from the jth suspect in the given s-instance is a uniform tail, for $1 \leq j \leq k$. Obviously, we have $T_i = \sum_{j=1}^{k} Y_j = k - \sum_{j=1}^{k} \bar{Y}_j$. Since all the k suspects are non-escaping, we know from Lemma 5 that $Pr[\bar{Y}_j] = o(1) < \frac{1}{8e}$ when $g = o(n/\log n)$. Thus $E[\sum_{j=1}^{k} \bar{Y}_j] < \frac{k}{8e}$. Invoking a Markov inequality and we have $Pr[\sum_{j=1}^{k} \bar{Y}_j \geq \frac{k}{2}] \leq \frac{1}{4e}$. This in turn means that $Pr[T_i < \frac{k}{2}] \leq \frac{1}{4e}$.

We obviously have $0 \leq T_i \leq k$ for $1 \leq i \leq r$. Define indicator random variable W_i to denote the event that $T_i < \frac{k}{2}$ for $1 \leq i \leq r$. Obviously, we have $Pr[W_i] \leq \frac{1}{4e}$ for any i. Define $W = \sum_{i=1}^{r} W_i$, where $E[W] < \frac{r}{4e}$. Because all r s-instances are independent, we can invoke a strong Chernoff bound on W, which will show that $Pr[W \geq \frac{r}{2}] = Pr[W \geq (2e) \cdot E[W]] < 2^{-r/2}$. This means that with probability of at least $1 - 2^{-r/2}$, there will be at least $(\frac{r}{2} - 1)$ s-instances with $T_i \geq \frac{k}{2}$. Thus the binomial distribution with mean of $(\frac{r}{2} - 1) \cdot \frac{k/2}{2m} > rk/(16m)$ (when n, and thus m and r, sufficiently large) is a lower bound on Z.

To upper bound Z, notice that we have r s-instances and that $T_i \leq k$ for $1 \leq i \leq r$. Thus the binomial distribution with mean of $r \cdot \frac{k}{2m} = rk/(2m)$ upper bounds Z. □

Comment. In the proofs below, we will invoke Chernoff bounds on the binomial distributions to bound the tail distribution of Z. This is why we only care about the means of the binomial distributions.

Lemma 11 *In $\mathcal{B}$, assume $g = o(n/\log n)$. For any given constant $\delta > 0$, we can find a universal constant r_0 such that let $r = r_0 \cdot \sqrt{m}$ will give us the following property. If we pick an arbitrary non-escaping verifier V and an arbitrary non-escaping suspect S, construct a uniformly random permutation of all suspects (including honest and sybil suspects), and let V verify the sequence one by one, then the probability of V accepting S is at least $1 - \delta$.*

Proof: In $\mathcal{B}$, only non-escaping honest suspects may affect the load on V's tails. Thus to simplify discussion in the following, we delete all other nodes from the sequence. Based on Lemma 9, we can pick appropriate r_0 such that for V and any non-escaping suspect S', $\mathcal{U}(V)$ and $\mathcal{U}(S')$ intersect with probability of at least $1 - \delta^2/64$. We will prove that such r_0 will satisfy the requirement of the lemma.

Consider the set H of all non-escaping suspects. Let Y be the number of suspects in H whose uniform tail set does not intersect with V's uniform tail set:

$$Y \quad = \quad |\{S \mid S \in H \text{ and } \mathcal{U}(S) \text{ does not intersect with } \mathcal{U}(V)\}|$$

Obviously we have $E[Y] \leq \delta^2|H|/64$. Invoke a Markov inequality and we have $Pr[Y \geq \delta|H|/8] \leq \delta/8$. This means that probability at least $1 - \delta/8$, the total number of non-escaping suspects with their uniform tail sets intersecting with $\mathcal{U}(V)$ is at least $(1 - \delta/8)|H|$. Consider a uniformly random permutation of the $|H|$ suspects. Let the ith suspect in the sequence be S_i for $1 \leq i \leq |H|$. Suppose that S is the kth suspect in the sequence, where $1 \leq k \leq |H|$. Obviously, with probability $1 - \delta/4$, $k \leq (1 - \delta/4)|H|$.

For $1 \leq i \leq k$, define indicator random variable X_i to be the event that the combined load imposed by the first i suspects in the sequence satisfies the balance condition. Obviously, X_1 means that the load imposed by the first suspect satisfies the balance condition. If the first suspect also satisfy the intersection condition, it will be accepted. On the other hand, since the verifier verifies the suspects sequentially, X_2 by itself does not necessarily mean that the load imposed by the second

suspect (after the first suspect) will satisfy the balance condition, since the first suspect could have been rejected if X_1 is not true. Thus X_2 only means that the combined load of the first and the second suspect satisfies the balance condition. However, $X_1X_2...X_k$ do indeed imply that all k suspects, when verified sequentially, satisfy the balance condition.

Considered upon $Y \leq \delta|H|/8$ and $k \leq (1-\delta/4)|H|$, Lemma 12 below will prove that $Pr[X_1X_2...X_k] \geq 1-\delta/2$. Notice that $\mathcal{U}(V)$ and $\mathcal{U}(S)$ intersects with probability of at least $1-\delta^2/64$. A union bound then shows that S is accepted with probability at least $1-\delta$. □

Lemma 12 *In $\mathcal{B}$, assume $g = o(n/\log n)$. Let δ, r_0, H, S_i (for $1 \leq i \leq |H|$), S, k, Y, and X_i (for $1 \leq i \leq k$) be the same as in the proof of Lemma 11. If $Y \leq \delta|H|/8$ and $k \leq (1-\delta/4)|H|$, then using any constant $h \geq max(24, 4.8r_0^2)$ in the balance condition is sufficient to ensure $Pr[X_1X_2...X_k] \geq 1-\delta/2$.*

Proof: For $1 \leq i \leq k$, we will prove that $Pr[X_i] \geq 1-\delta/(2n)$. Let $\mu = ri/(2m)$:

- $\mu \leq 10\log r$**:** Notice that the load of a tail in $\mathcal{B}$ will never be larger than in $\mathcal{A}$. Since X_i is about the combined load imposed by the first i suspects that are non-escaping, we can invoke Lemma 10 to reason about the distribution for the load. Notice that we are not yet discussing whether individual suspect will be accepted. Lemma 10 shows that in $\mathcal{A}$, the load of a tail in $\mathcal{U}(V)$ is upper bounded by a binomial distribution with mean of μ. A Chernoff bound on the binomial distribution will show that the probability of the load being larger than $24\log r$ is at most $e^{-(19.6\log r)/4} < 1/r^4$. The load on different tails are correlated. But a union bound can still show that the probability that all tails in $\mathcal{U}(V)$ have a load smaller than $24\log r$ is at least $1-r\cdot 1/r^4 > 1-1/r^3 = 1-1/(r_0^3 m^{1.5}) > 1-1/n^{1.5} = 1-o(1/n)$. On the other hand, the bar is at least $h\log r > 24\log r$ which means that the balance condition must be satisfied with probability of at least $1-o(1/n) > 1-\delta/(2n)$.

- $\mu > 10\log r$**:** Again the load of a tail in $\mathcal{B}$ will never be larger than in $\mathcal{A}$. Lemma 10 shows that in $\mathcal{A}$, the load of a tail in $\mathcal{U}(V)$ is upper bounded by a binomial distribution with mean of μ. A Chernoff bound will show that the probability of the load being larger than 2.4μ is at most $e^{-(1.96\mu)/4} < e^{-(19.6\log r)/4} < 1/r^4$. A same union bound as before will then show that the probability that all tails in $\mathcal{U}(V)$ have a load smaller than 2.4μ is at least $1-o(1/n)$.

  On the other hand, define random variable Z to denote the number of non-escaping suspects before S_i whose uniform tail sets intersect with V's uniform tail set:

$$Z \quad = \quad |\{S_j | 1 \leq j \leq i \text{ and } \mathcal{U}(S_j) \text{ intersect with } \mathcal{U}(V)\}|$$

  Conditioned upon $Y \leq \delta|H|/8$ and $1 \leq j \leq i \leq k \leq \delta|H|/4$, Lemma 13 will prove:

$$Pr[Z \geq \frac{i}{4}] \quad \geq \quad 1-exp(-\frac{i}{16})$$

  From the conditions $\mu > 10\log r$ and $\mu = \frac{ri}{2m}$, we have $i > \frac{20m\log r}{r} = \frac{20\sqrt{m}\log\left(r_0\sqrt{m}\right)}{r_0} > \frac{20\log r_0}{r_0}\sqrt{n}$. Therefore:

$$Pr[Z \geq \frac{i}{4}] \quad \geq \quad 1-exp\left(-\frac{i}{16}\right) > 1-exp\left(-\frac{20\log r_0}{16r_0}\sqrt{n}\right) = 1-o(\frac{1}{n})$$

  Thus with at least $1-o(1/n)$ probability, the total load across all tails is at least $i/4$, and the bar is at least:

$$h\cdot a \geq 4.8r_0^2 \cdot \frac{i}{4r} = \frac{4.8ir}{4m} = 2.4\mu$$

  A union bound then shows $Pr[X_i] \geq 1-\delta/(2n)$.

Finally, $k \leq n$ and a simple union bound across all X_i's finishes the proof. □

Lemma 13 *In $\mathcal{B}$, assume $g = o(n/\log n)$. Let δ, r_0, H, S_i (for $1 \leq i \leq |H|$), S, k, Y, X_i (for $1 \leq i \leq k$), and Z be the same as in the proof of Lemma 12. Conditioned upon $Y \leq \delta|H|/8$ and $1 \leq i \leq k \leq \delta|H|/4$, we have:*

$$Pr[Z \geq \frac{i}{4}] \geq 1-exp(-\frac{i}{16})$$

Proof: Assume that all routing table contents are already known. Conditioned upon $Y \leq \delta|H|/8$, define indicator random variable Z_j to denote the event that $\mathcal{U}(S_j)$ intersects with $\mathcal{U}(V)$, for $1 \leq j \leq i$. Notice that Z_j is define over the domain of all possible permutations (of the non-escaping suspects) instead of over the domain of all possible routing table contents. Obviously, we have $Z = Z_1 + Z_2 + ... + Z_i$. In the remainder of the proof for this lemma, when we say a suspect "intersect" with the verifier, we mean the suspect's uniform tail set intersects with the verifier's uniform tail set.

We already know that out of the $|H|$ non-escaping suspects, only $\delta/4$ fraction of them do not intersect with the verifier. A random permutation of the $|H|$ suspects can be constructed in the following way: We first pick a (uniformly) random suspect out of H (without replacement) as S_1. Next we pick a (uniformly) random suspect out of the remaining nodes (without replacement) as S_2, and so on. Let H_j be the set of remaining suspects immediately before we pick S_j. For any $1 \leq j \leq i \leq k \leq \delta|H|/4$, the fraction of suspects in H_j that intersect with the verifier is at least:

$$\frac{(1-\delta/8)\,|H| - (j-1)}{|H| - (j-1)} \quad = \quad 1 - \frac{\delta|H|/8}{|H| - j + 1} \geq 1 - \frac{\delta|H|/8}{|H| - (1-\delta/4)\,|H| + 1} = 1 - \frac{\delta|H|/8}{\delta|H|/4 + 1} > 0.5$$

This means that the event Z_j occurs with probability at least 0.5, regardless of whether events Z_1, Z_2, ... Z_{j-1} occur or not.

Define random variable Z' as the sum of i independent Bernoulli trials where each Bernoulli trial succeeds with probability of 0.5. A Chernoff bound of Z' tells us that

$$Pr[Z' \leq i/4] \quad = \quad Pr[Z' \leq (1-0.5)E[Z']] \leq exp\,(-E[Z']/8) = exp(-i/16)$$

Finally, it is obvious that $Pr[Z \geq i/4] \geq Pr[Z' \geq i/4] \geq 1 - Pr[Z' \leq i/4] \geq 1 - exp(-i/16)$. □

Comment. $\mathcal{C}$ is actually a special case of $\mathcal{B}$, thus Lemma 11, 12 and 13 for $\mathcal{B}$ directly carry over to $\mathcal{C}$.

A.4.2 Proofs for $\mathcal{D}$

$\mathcal{D}$ has two differences from $\mathcal{C}$. First, in $\mathcal{D}$, every accepted non-escaping honest suspect picks the least loaded tail out of all intersecting tails (instead of out of those intersecting tails in $\mathcal{U}(V)$). We define $\sigma(V)$ to denote all of V's tails not in $\mathcal{U}(V)$. Namely, these are V's non-uniform tails and escaping tails. Second, the adversary may now interfere to cause honest nodes to be rejected, by intentionally cause load imbalance.

We need to precisely model such interference from the adversary. To do so, we start from a sequence $S_1S_2...$ of non-escaping honest suspects that were all accepted under $\mathcal{C}$. We consider this sequence of honest suspects as a sequence of *white* balls, each of which goes into some bin (tail) in $\mathcal{U}(V)$ in $\mathcal{C}$. In $\mathcal{D}$, the adversary may interfere in the following two ways, with the goal of causing some of suspects previously accepted to be now rejected:

1. The adversary may introduce sybil nodes with tails intersecting with V's tails. If these sybil nodes are accepted, then the load of V's tails will increase, potentially causing some previously accepted non-escaping honest suspects to be now rejected. As a concrete example, imagine that previously in $\mathcal{C}$, S_5 only had a single intersection with all of V's tails and the intersection is on V's uniform tail #12. So S_5 conceptually placed a ball into bin 12. Now in $\mathcal{D}$, before S_5 is verified, the adversary introduces a sybil node with tails intersecting with tail 12. Further assume that after the sybil node is accepted, tail 12's load happens to reach the bar. Next when S_5 is verified in $\mathcal{D}$, S_5 will no longer satisfy the balance condition and thus be rejected.

   We model this kind of interference from the adversary by allowing the adversary in $\mathcal{D}$ to insert *red* balls at any place in the sequence of white balls. Each red ball corresponds to an accepted sybil node, and will increment the load of some tail of V's.

   Our later proof will prove that the "damage" caused by each red ball is limited, leveraging the following key observation in the above example: After S_5 is rejected, the load on tail 12 go back to "normal". More precisely, the sybil node caused tail 12 to have one extra load than before. However, S_5 should have incremented the load of tail 12. Now that S_5 is rejected, the load of tail 12 becomes the same as before (i.e., as in $\mathcal{C}$). In other words, after S_5 is rejected, the influence of the adversary "disappears". To cause the rejection of another honest suspect, the adversary has to introduce another red ball. The scenario can get more complex. Nevertheless, our proof later is based on a generalization of the above intuition.

2. The adversary may register with some of V's tails the public key of some honest suspect previously accepted under $\mathcal{C}$. This kind of interference is quite subtle and perhaps counter-intuitive. As a concrete example, imagine that previously

in $\mathcal{C}$, S_5 only had a single intersection with all of V's tails and the intersection is on V's uniform tail #12. Now in $\mathcal{D}$, the adversary intentionally registers the public key of S_5 with one of the tainted tails that intersect with V's tail #14. The adversary can further fool S_5 into believing that the tail is indeed one of S_5's tail, as long as S_5 has at least one escaping tail. After all such manipulation from the adversary, S_5's tails now have two intersections with V's tails. Assume that when S_5 is verified, tail 14 has a lower load than tail 12. As a result, S_5 will increment the load of tail 14 instead of tail 12. It may seem at this point that the adversary almost helped us to achieve better balancing. However, imagine that later tail 14 gets overloaded, and because the adversary caused tail 14 to have one extra load, some honest suspect (e.g., S_{16}) may be rejected.

We model this second kind of interference from the adversary by allowing the adversary in $\mathcal{D}$ to replace some white balls in the sequence with *green* balls. A green ball (in $\mathcal{D}$) will go into a different bin from the bin that the replaced white ball went into in $\mathcal{C}$. In our example above, the green ball (corresponding to S_5) goes into bin 14 in $\mathcal{D}$, while the corresponding white ball went into bin 12 in $\mathcal{C}$.

Later we intend to prove that the "damage" caused by each green ball is limited. The intuition here is trickier than the earlier intuition for red balls. Namely, here even after S_{16} is rejected, the load of the tails do not go back to "normal". For example, tail 12 still has one less load than before. Because our bar is floating with the average (and thus total load), this may cause the bar to be too low. The key insight here is that to cause the total load to drop by one, at least one honest suspect is accepted (S_5 in the example). Thus the total load in $\mathcal{D}$ can never drop below half of the total load in $\mathcal{C}$. This in turn means that doubling the constant h used in $\mathcal{C}$ is sufficient to prevent the bar from dropping and offset such disruptive effect.

Finally, if some white balls in the sequence are now rejected in $\mathcal{D}$ (e.g., due to interference from the adversary), we say that they are now *black*. By definition, it is impossible for a ball to be simultaneously black and green.

The total number of red balls and green balls that the adversary can use in bins (tails) in $\sigma(V)$ is potentially unlimited. But for bins (tails) in $\mathcal{U}(V)$, Section 6.2 explained that the total number of red balls and green balls the adversary can use is within $\epsilon'' n$ for some small ϵ''.

Lemma 14 *Consider a sequence of honest non-escaping honest suspects that were all accepted under $\mathcal{C}$ (i.e., a sequence of white balls). In $\mathcal{D}$, we double the constant h in the balance condition from $\mathcal{C}$. Suppose that in $\mathcal{D}$ by inserting red balls and replacing some white balls with green balls, the adversary manages to prevent K of those suspects from being accepted under $\mathcal{D}$. Then the total number of red and green balls used in bins in $\mathcal{U}(V)$ is at least K.*

Proof: To avoid notation confusion, we let the bar in $\mathcal{C}$ be $b = h \cdot max(\log r, a)$ and the bar in $\mathcal{D}$ be $b' = h' \cdot max(\log r, a')$, where $h' = 2h$. Let the K non-escaping honest suspected rejected in $\mathcal{D}$ be S_1, S_2, ..., S_K. We use induction to prove the following two claims for $1 \leq k \leq K$:

Claim 1 When verifying S_k, $b' \geq b$.

Claim 2 In $\mathcal{D}$, before S_k is verified, the total number of red balls and green balls used in bins in $\mathcal{U}(V)$ is at least k.

Induction base. For $k = 1$, notice that when verifying S_1, the total load of all the tails in $\mathcal{D}$ must be no smaller than in $\mathcal{C}$. This is because all white balls before S_1 in the sequence are still accepted in $\mathcal{D}$ (since S_1 is the very first black ball), and thus contribute to the total load. The total load in $\mathcal{D}$ can be larger because the adversary may insert additional red balls. As a result, we have $a' \geq a$ and thus $b' = h' \cdot max(\log r, a') > h \cdot max(\log r, a) = b$.

Next we prove **Claim 2** by showing that when verifying S_1, if the total number of red and green balls in bins in $\mathcal{U}(V)$ is 0, then S_1 must be accepted. Let the number of white balls before S_1 in $\mathcal{C}$ be z. We will prove via an induction on z that when verifying S_1, for any tail in $\mathcal{U}(V)$, its load in $\mathcal{D}$ is no larger than in $\mathcal{C}$. If this is indeed correct, then together with the fact that $b' \geq b$, we know that S_1 must be accepted in $\mathcal{D}$.

The case for $z = 0$ is trivial. Now assume that the previous argument holds for z and we consider $z + 1$. Let R be the $(z + 1)$th white ball in $\mathcal{C}$. Suppose that R goes into bin (tail) i in $\mathcal{C}$ and bin (tail) j in $\mathcal{D}$. By definition of $\mathcal{C}$, bin i must be in $\mathcal{U}(V)$. Obviously, we only need to consider the case where $i \neq j$ and where bin j is in $\mathcal{U}(V)$ as well. We first consider the case of $i < j$. Immediately before R is accepted, let the load of bin i be x and x' in $\mathcal{C}$ and $\mathcal{D}$, respectively. Similarly let the load of bin j be y and y'. Because there are no green balls in bin i and bin j, it means that both bin i and bin j are intersecting tails in both $\mathcal{C}$ and $\mathcal{D}$. Because R chooses bin i (over bin j) in $\mathcal{C}$ and bin j (over bin i) in $\mathcal{D}$, we know that $x \leq y$ and $x' > y'$. Induction hypothesis tells us that $x' \leq x$ and $y' \leq y$. If $y' < y$, then the claim still holds after R goes into bin j in $\mathcal{D}$. If $y' = y$, we must have $x' > y' = y \geq x \geq x'$, which is impossible. Finally, the case for $i > j$ is similar.

Inductive step. Assume now that the previous two claims hold up to k. Namely,

1. For any $1 \leq i \leq k$, when verifying S_k, $b' \geq b$.
2. In $\mathcal{D}$, before S_k is verified, the total number of red balls and green balls used in tails in $\mathcal{U}(V)$ is at least k.

To prove **Claim 1** for $k+1$, we compare the total load $load$ in $\mathcal{C}$ immediately after S_k is accepted and the total load $load'$ in $\mathcal{D}$ immediately after S_k is rejected. Assume that the adversary has used x red balls and y green balls in tails in $\mathcal{U}(V)$. We know that:

- $x + y \geq k$: From induction hypothesis.
- $y \leq load - k$: Since each green ball corresponds to some distinct white ball in $\mathcal{C}$ and there are at most $load - k$ white balls so far.
- $load' = load + x - k$: Each red ball increments the total load.

Given these relationships, Lemma 15 proves that $load'/load \geq 0.5$. Now notice that by definition, all white balls between S_k and S_{k+1} are all accepted in both $\mathcal{C}$ and $\mathcal{D}$. This means that when verifying S_{k+1}, we still have $load'/load \geq 0.5$ and thus $a' \geq a/2$. This in turn means that $b' = h' \cdot max(\log r, a') \geq 2h \cdot max(\log r, a/2) \geq h \cdot max(\log r, a) = b$.

To prove **Claim 2** for $k+1$, it suffices to show that if the adversary only uses k red and green balls in bins in $\mathcal{U}(V)$, it is impossible for S_{k+1} to be rejected. We define:

$$extra \quad = \quad \sum_{\text{bin } i \text{ is in } \mathcal{U}(V)} max(0, \text{bin } i\text{'s load in } \mathcal{D} - \text{ bin } i\text{'s load in } \mathcal{C})$$

Obviously, if $extra = 0$ immediately before verifying S_{k+1}, then together with $b' \geq b$, we immediately know that S_{k+1} will be accepted. To prove $extra = 0$, we will show that:

$$extra \quad \leq \quad \text{number of red balls and green balls} - \text{number of black balls} \tag{1}$$

If the above inequality holds, then using only k red and green balls will leave us with $extra \leq 0$, and S_{k+1} will be accepted. This will then complete the proof for the inductive step for **Claim 2**.

Consider the sequence of balls in $\mathcal{D}$ up to but not including S_{k+1}. We prove Inequality 1 via an induction on the length z of the sequence. The induction base for $z = 0$ is trivial, since both sides of the inequality is 0. Now assume the inequality holds for z and we will prove that it holds for $z + 1$. We consider the color of the last ball in the sequence:

Black This means that the ball is S_k. Suppose the ball goes to bin i in $\mathcal{C}$ and increments the load of bin i from x to $x + 1$. From Claim 1, $b' \geq b$. Thus if S_k is rejected in $\mathcal{D}$, it must be because that bin i has a load of at least $x + 1$ before verifying S_k in $\mathcal{D}$. Since S_k is accepted in $\mathcal{C}$ and is rejected in $\mathcal{D}$, it must decrease $extra$ by 1. Thus Inequality 1 still holds.

Red Obvious because $extra$ can increase by at most 1 while the right-hand side of Inequality 1 is guaranteed to increase by 1.

Green Obvious because $extra$ can increase by at most 1 while the right-hand side of Inequality 1 is guaranteed to increase by 1.

White Suppose the white ball goes into bin i in $\mathcal{C}$ and bin j in $\mathcal{D}$. By definition of $\mathcal{C}$, bin i must be in $\mathcal{U}(V)$. Obviously, we only need to consider the case of $i \neq j$. If bin j is in $\sigma(V)$, then this white ball will either decrement $extra$ or leave it unchanged. Thus Inequality 1 will continue to hold.

Next we consider the case where bin j is in $\mathcal{U}(V)$ and where $i < j$. Immediately before the white ball is accepted, let the load of bin i be x and x' in $\mathcal{C}$ and $\mathcal{D}$, respectively. Similarly let the load of bin j be y and y'. Because the ball is already know to be white (instead of green), it means that both bin i and bin j are intersecting tails in both $\mathcal{C}$ and $\mathcal{D}$. Because the white ball chooses bin i (over bin j) in $\mathcal{C}$ and bin j (over bin i) in $\mathcal{D}$, we know that $x \leq y$ and $x' > y'$. If $y' < y$, then $extra$ can never increase because of the white ball. On the other hand, if $y' \geq y$, we must have $x' > y' \geq y \geq x$. Thus the white ball will decrement $extra$ on the behalf of bin i and increment $extra$ on the behalf of bin j, again leaving $extra$ unchanged. Finally, the case where bin j is in $\mathcal{U}(V)$ and where $i > j$ is similar.

This finishes the proof for Inequality 1, which in turn, completes the inductive step for **Claim 2**. □

Lemma 15 *Consider arbitrary integers $k \geq 1$, $x \geq 0$, $y \geq 0$, $load \geq 0$, and $load' \geq 0$, where $x + y \geq k$, $y \leq load - k$, and $load' = load + x - k$. Then $load'/load \geq 0.5$.*

Proof:

$$\begin{aligned} & load'/load \geq 0.5 \\ \Leftrightarrow \quad & load \geq 2k - 2x \\ \Leftarrow \quad & y + k \geq 2k - 2x \\ \Leftrightarrow \quad & y + 2x \geq k \\ \Leftarrow \quad & y + x \geq k \end{aligned}$$

□

A.5. Proof for the main SybilLimit theorem—Theorem 3

Proof for Theorem 3:. To avoid notation collision, we will prove that "for any given constant $\epsilon' > 0$ and $\delta' > 0$, we can always find a set of $(1-\epsilon')n$ honest verifiers and universal constants w_0 and r_0, such that using $w = w_0 \log n$ and $r = r_0\sqrt{m}$ in SybilLimit will guarantee that for any given verifier V in the set, with probability of at least $1 - \delta'$, V accepts at most $O(g \log n)$ sybil nodes and at least $(1 - \epsilon')n$ honest nodes".

Lemma 5 tells us that there exist at least $(1 - \epsilon)n$ non-escaping verifiers. We let w_0 to be the w_0 as determined in Lemma 5 and r_0 to be the r_0 as determined in Lemma 11. For number of sybil nodes accepted, Lemma 7 and 8 tells us that a non-escaping verifier V will accept at most $O(\log n)$ sybil nodes with probability of at least $1 - 3\delta$.

Next consider the number Q of honest nodes accepted. There are at most ϵn escaping honest suspects. Consider any non-escaping honest suspects S and non-escaping verifier V. In $\mathcal{C}$ for any $\epsilon > 0$, we know from Lemma 11 that V will accept S with probability of at least $1 - \epsilon$. Let X be the number of non-escaping honest suspects rejected by V (out of the n honest suspects). We have $E[X] \leq \epsilon n$. Invoke a Markov inequality and we have $Pr[X \geq \frac{\epsilon}{\delta} n] \leq \delta$.

Next we consider the interference from the adversary on the balance condition. As from Section 6.2, the load on a uniform tail of V's may increase when it intersects with:

1. **Uniform tails of non-escaping honest suspects.**

2. **Non-uniform tails of non-escaping honest suspect.** In each s-instance, Lemma 5 tells us that there are on expectation $o(n) < \epsilon n$ such tails.

3. **Uniform or non-uniform tails of escaping honest suspects.** There are at most ϵn such tails in each s-instance.

4. **Tainted Tails.** There are $O(g \log n) = o(n) < \epsilon n$ such tails in each s-instance.

In each s-instance, the expected number of tails in the last three cases is thus at most $3\epsilon n$. We would like to prove in the next that the number of intersections (Y) between $\mathcal{U}(V)$ and these tails, in all s-instances, satisfies $Pr[Y > \frac{1.5 r_0^2 \epsilon}{\delta} n] < \delta$. The proof is somewhat similar to the proof of Lemma 7. In a given s-instance, let random variable Z denote the number of tails in the last three cases. Let $p_i = Pr[Z = i]$. We obviously have $\sum p_i \cdot i = E[Z] \leq 3\epsilon n$. Consider any tail in $\mathcal{U}(V)$, and because the tail is a uniform tail, we know that the expected number of intersections it has with those tails in the given s-instance is:

$$\sum p_i \cdot \frac{i}{2m} \leq \frac{3\epsilon n}{2m}$$

With total r s-instances, the expected number of intersections will be at most $\frac{3\epsilon r n}{2m}$. Finally, we trivially have $|\mathcal{U}(V)| \leq r$ and thus $E[Y] \leq \frac{r^2 \cdot 3\epsilon n}{2m} = 1.5 r_0^2 \epsilon n$. Invoke a Markov inequality on Y and we have $Pr[Y \geq \frac{1.5 r_0^2 \epsilon}{\delta} n] \leq \delta$.

Now from Lemma 14, we know that in $\mathcal{D}$, Y is the maximum number of honest suspects that can be rejected due to adversary's interference. Thus with probability of at least $1-2\delta$, we have:

$$\begin{aligned} n-Q &\leq \epsilon n \text{ (escaping suspects) } + \frac{\epsilon}{\delta}\cdot n \text{ (non-escaping suspects rejected under } \mathcal{C}) \\ &\quad + \frac{1.5r_0^2\epsilon}{\delta} n \text{ (additional non-escaping suspected rejected when going from } \mathcal{C} \text{ to } \mathcal{D}) \\ &= (\epsilon + \frac{\epsilon}{\delta} + \frac{1.5r_0^2\epsilon}{\delta})\cdot n \end{aligned}$$

To finish the proof, we only need to find constants $\epsilon > 0$ and $\delta > 0$ satisfying:

$$\begin{aligned} 5\delta &\leq \delta' \\ \epsilon &\leq \epsilon' \\ \epsilon + \frac{\epsilon}{\delta} + \frac{1.5r_0^2\epsilon}{\delta} &\leq \epsilon' \end{aligned}$$

One can easily verify the following constant ϵ and δ will satisfy the above inequalities:

$$\begin{aligned} \delta &= \min(1, \frac{\delta'}{5}) \\ \epsilon &= \min(\epsilon', \frac{\epsilon'\cdot\min(1,\frac{\delta'}{5})}{2+1.5r_0^2}) \end{aligned}$$

This then completes our proof. □

A.6. Proof for the $\Omega(g)$ Lower Bound

Theorem 16 *Consider any constant $c > 0$ and any $g \in [1, n]$. For any graph G with n nodes and $O(\log n)$ mixing time, we can always find another graph G' with $n' = (n + c\cdot g)$ nodes where i) G is a subgraph of G', ii) the number of edges between nodes in G and nodes in $G' \setminus G$ is g, and iii) G' has $O(\log n')$ mixing time.*

Proof: To prepare for the proof, we need to introduce the notion of *conductance*. Consider any given graph with vertex set of V. For any $S \subseteq V$, we define $\bar{S} = V \setminus S$. For any $X \subseteq V$ and $Y \subseteq V$, define function $e(X,Y) = |\{(x,y) \mid x \to y \text{ is a directed edge of graph } G \text{ and } x \in X \text{ and } y \in Y\}|$. The conductance Φ is defined as:

$$\begin{aligned} \Phi &= \min_{S \subseteq V \text{ and } e(S,S)+e(S,\bar{S}) \leq |E|} \left(\frac{e(S,\bar{S})}{e(S,S)+e(S,\bar{S})} \right) \\ &= \min_{S \subseteq V \text{ and } e(S,S)+e(S,\bar{S}) \leq |E|} \left(\frac{1}{\frac{e(S,S)}{e(S,\bar{S})}+1} \right) \end{aligned}$$

Classic theory [16] on graph mixing time tells us that Φ being lower bounded by a positive constant is a both sufficient and necessary condition for $O(\log n + \frac{1}{\Delta})$ mixing time under $\Delta = \frac{1}{n}$.

We obtain G' in the following way. We first pick an arbitrary connected graph F with c sybil nodes. Obviously, there are many such F's. Next we pick arbitrary g nodes from G. For each node picked, we attach F to that node using a single edge. We called that node as the *introducer* of all the nodes in F. It does not matter which node in F that edge is connected to.

Obviously, G' has $(n + c\cdot g)$ nodes and G is a subgraph of G'. Also, the number of edges between nodes in G and nodes in $G' \setminus G$ is exactly g. We only need to prove that G' has $O(\log n)$ mixing time as well. Let E be G's undirected edge set. Similarly define V' and E' for G'. Let Φ and Φ' be the conductance of G and G', respectively. We will prove that Φ' is lower bounded by a constant where

$$\Phi' = \min_{S' \subseteq V' \text{ and } e(S',S')+e(S',\bar{S}') \leq |E'|} \left(\frac{1}{\frac{e(S',S')}{e(S',\bar{S}')}+1} \right) \quad (2)$$

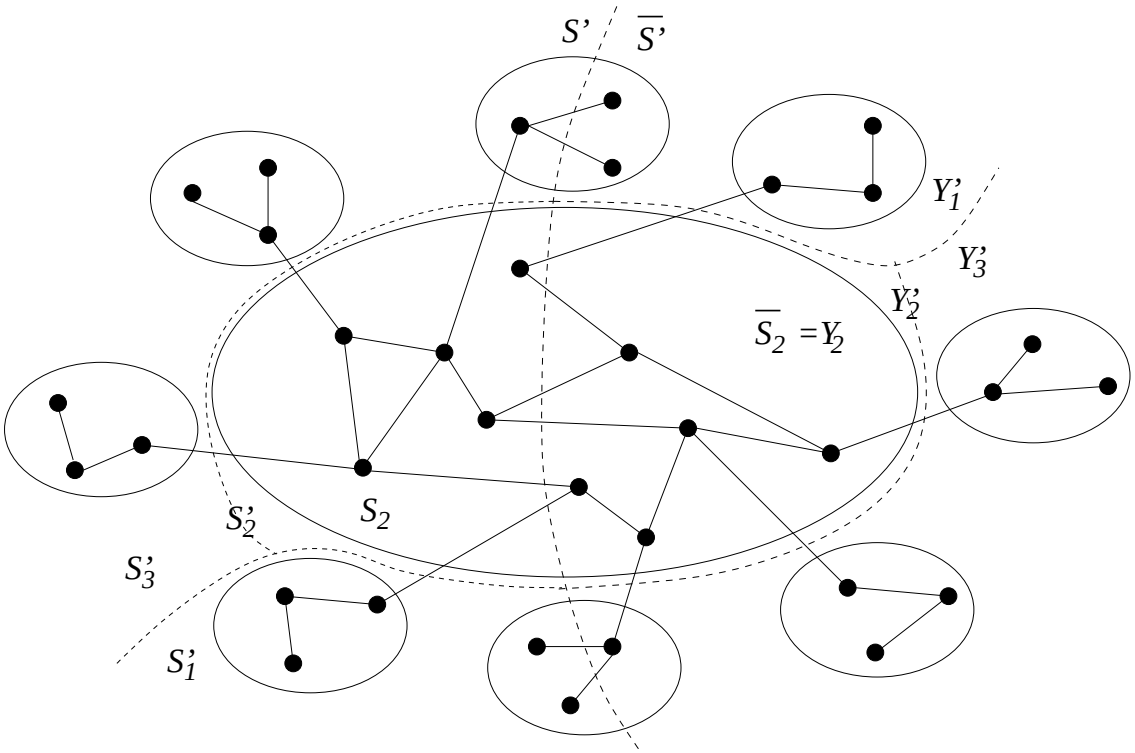

Figure 13. Definition of sets in G' in the proof for Theorem 16.

Consider any $S' \subseteq V'$ (see Figure 13), from the definition of conductance, we have the condition that $e\left(S', S'\right) + e\left(S', \bar{S}'\right) \leq |E'| = \frac{1}{2}\left(e\left(S', S'\right) + e\left(S', \bar{S}'\right) + e\left(\bar{S}', S'\right) + e\left(\bar{S}', \bar{S}'\right)\right)$, and since $e\left(S', \bar{S}'\right) = e\left(\bar{S}', S'\right)$, we get $e\left(S', S'\right) \leq e\left(\bar{S}', \bar{S}'\right)$. If we want to prove that Φ' is lower bounded by a positive constant, we only need to argue that $\frac{e\left(S', S'\right)}{e\left(S', \bar{S}'\right)}$ is upper bounded by a positive constant.

In S', define the set $S'_1 = \{x \mid x \in S' \text{ and } x \notin V \text{ and } x\text{'s introducer is not in } S'\}$, $S'_2 = S' \setminus S'_1$. Define the set $S'_3 = \{x | x \in S'_2 \text{ and } x \notin V\}$, $S_2 = S'_2 \setminus S'_3$ and $\bar{S}_2 = V \setminus S_2$. Similar to the definition of S'_1, S'_2, S'_3 and S_2 in S', define sets Y'_1, Y'_2, Y'_3 and Y_2 respectively in $\bar{S}'$. Define $Y'_1 = \{x \mid x \in \bar{S}' \text{ and } x \notin V \text{ and } x's \text{ introducer is not in } \bar{S}'\}$, $Y'_2 = \bar{S}' \setminus Y_1$, $Y'_3 = \{x \mid x \in Y'_2 \text{ and } x \notin V\}$, and $Y_2 = Y'_2 \setminus Y'_3$. It is apparent that Y_2 is the same set as $\bar{S}_2$. Figure 13 shows the definition of all these sets.

In order to prove that $\frac{e\left(S', S'\right)}{e\left(S', \bar{S}'\right)}$ is upper bounded by a positive constant, we can first consider the simple case that $e(\bar{S}', \bar{S}') + e(\bar{S}', S') \leq k(e(Y'_1, Y'_1) + e(Y'_1, S'))$, k is some positive constant which is larger than one. Lemma 17 has proved that in this case $\frac{e(S', S')}{e(\bar{S}', \bar{S}')}$ is upper bounded by a positive constant. Without loosing generality, we set k to be 2 and we also have that $\frac{e(S', S')}{e(S', \bar{S}')}$ is upper bounded by $2c^2 + 1$ which is a positive constant.

Next, consider the case that $e(\bar{S}', \bar{S}') + e(\bar{S}', S') \geq 2(e(Y'_1, Y'_1) + e(Y'_1, S'))$. Lemma 19 can prove that in this case, $\frac{e(S', S')}{e(S', \bar{S}')}$ is also upper bounded by a positive constant.

Finally, we have that $\frac{e(S', S')}{e(S', \bar{S}')}$ is upper bounded by a positive constant in either case and Φ' is lower bounded by a positive constant. □

Lemma 17 *Let G, G', Φ, Φ', V, S', $\bar{S}'$, c, Y'_1,$e\left(X, Y\right)$ and F be the same as in the proof of Theorem 16. Conditioned upon that $e(\bar{S}', \bar{S}') + e(\bar{S}', S') \leq k(e(Y'_1, Y'_1) + e(Y'_1, S'))$, k is a positive constant which is larger than one, we have that $\frac{e(S', S')}{e(\bar{S}', S')}$ is upper bounded by a positive constant.*

Proof: Since $e(\bar{S}', \bar{S}') + e(\bar{S}', S') \leq k(e(Y'_1, Y'_1) + e(Y'_1, S'))$, and it is shown in the proof of Theorem 16 that $e(S', S') \leq$

$e(\bar{S}', S')$, we have:

$$\frac{e(S', S')}{e(\bar{S}', S')} \leq \frac{e(\bar{S}', \bar{S}')}{e(\bar{S}', S')} \leq \frac{k(e(Y_1', Y_1') + e(Y_1', S'))}{e(\bar{S}', S')} - 1$$

From the definition of Y_1', we have $Y_1' \subseteq \bar{S}'$. Therefore, $e(Y_1', S') \leq e(\bar{S}', S')$ and we can get:

$$\begin{aligned} \frac{e(S', S')}{e(\bar{S}', S')} &\leq \frac{k(e(Y_1', Y_1') + e(Y_1', S'))}{e(\bar{S}', S')} - 1 \\ &\leq \frac{k(e(Y_1', Y_1') + e(Y_1', S'))}{e(Y_1', S')} - 1 \\ &= \frac{k(e(Y_1', Y_1')}{e(Y_1', S')} + k - 1 \end{aligned}$$

Lemma 18 has proved that $\frac{e(Y_1', Y_1')}{e(Y_1', S')} \leq c^2$. Finally, we have $\frac{e(S', S')}{e(S', S')} \leq kc^2 + k - 1$. Since $k > 1$, it is obvious that $kc^2 + k - 1$ is a positive constant. □

Lemma 18 *Let G, G', S', $\bar{S}'$, c, Y_1', $e(X, Y)$ and F be the same as in the proof of Theorem 16. Then we have $\frac{e(Y_1', Y_1')}{e(Y_1', S')} \leq c^2$*

Proof: Consider that Y_1' can be partitioned into l groups ω_1, ω_2, ω_3,..., ω_l, and all nodes in each group are in the same connected graph F. Obviously, there is no edge between any two groups and $|Y_1'| = |\omega_1| + |\omega_2| + |\omega_3| + ... + |\omega_l|$. For any $1 \leq i \leq l$, $|\omega_i| \leq c$.

$$\begin{aligned} e(Y_1', Y_1') &< |\omega_1|^2 + |\omega_2|^2 + |\omega_3|^2 + ...|\omega_l|^2 \\ &\leq c \cdot (|\omega_1| + |\omega_2| + |\omega_3| + ... + |\omega_l|) \\ &= c \cdot |Y_1'| \end{aligned}$$

Because for each group in Y_1', there is an edge connecting it to its nodes' introducer in S' or there are edges connecting nodes in this group to those nodes not in this group but in the same F as this group, the number of edges between Y_1' and S' is at least the number of groups *l*, so we have $e(Y_1', S') \geq l$, and since each group contains at most c nodes, it is obvious that $l \geq \frac{|Y_1'|}{c}$. Then we have :

$$\frac{e(Y_1', Y_1')}{e(Y_1', S')} \leq \frac{c \cdot |Y_1'|}{l} \leq \frac{c \cdot |Y_1'|}{\frac{|Y_1'|}{c}} = c^2$$

□

Lemma 19 *Let G, G', Φ, Φ', V, S', $\bar{S}'$, c, S_1', S_2', S_3', S_2, Y_1', Y_2', Y_3', Y_2, $e(X, Y)$ and F be the same as in the proof of Theorem 16. Conditioned upon that $e(\bar{S}', \bar{S}') + e(\bar{S}', S') \geq 2(e(Y_1', Y_1') + e(Y_1', S'))$, we have $\frac{e(S', S')}{e(S', S')}$ is upper bounded by a positive constant.*

Proof: From the definition of S_1' and S_2', we can easily get the following equation:

$$\frac{e(S', S')}{e(S', \bar{S}')} = \frac{e(S_1', S_1') + e(S_2', S_2') + e(S_1', S_2') + e(S_2', S_1')}{e(S_1', \bar{S}') + e(S_2', \bar{S}')}$$

Since nodes in S_1' are not in V and their introducers are not in S', there will be no edge in S' connecting them to V or to other nodes not in V but whose introducers are in S', thus there is no edge between S_1' and S_2'. So we have $e(S_2', S_1') = e(S_1', S_2') = 0$. Then, the equation becomes:

$$\frac{e(S', S')}{e(S', \bar{S}')} = \frac{e(S_1', S_1') + e(S_2', S_2')}{e(S_1', \bar{S}') + e(S_2', \bar{S}')} \tag{3}$$

In order to prove that $\frac{e(S',S')}{e(S',\bar{S}')}$ is upper bounded by a positive constant, we only need to argue that both $\frac{e(S'_1,S'_1)}{e(S'_1,\bar{S}')}$ and $\frac{e(S'_2,S'_2)}{e(S'_2,\bar{S}')}$ are upper bounded by some positive constant. Lemma 20 has proved that $\frac{e(S'_1,S'_1)}{e(S'_1,\bar{S}')}$ is upper bounded by c^2. Consider $\frac{e(S'_2,S'_2)}{e(S'_2,\bar{S}')}$. From the definition of S'_2, S_2, S'_3 we get:

$$e\left(S'_2,S'_2\right)=e\left(S'_3,S'_3\right)+e\left(S_2,S_2\right)+e\left(S_2,S'_3\right)+e\left(S'_3,S_2\right)$$

Using the similar argument in the proof of Lemma 20, we can consider that S'_3 can be partitioned into r groups ω_1, ω_2, ω_3,..., ω_r and we have $e\left(S'_3,S'_3\right)\leq c\cdot|S'_3|$. Since each group in S'_3 connects to some distinct node in S_2 and each group contains at most c nodes, we have $|S'_3|\leq c\cdot|S_2|$. Because the edges between S_2 and S'_3 are those connecting a group in S'_3 and a corresponding node, the introducer of all the nodes in that group, in S_2, we get $e\left(S_2,S'_3\right)=e\left(S'_3,S_2\right)\leq|S_2|$. Since S_2 is a subset of S'_2, we have $e\left(S_2,\bar{S}'\right)\leq e\left(S'_2,\bar{S}'\right)$. And from the definition of $\bar{S}_2$, it is obvious that $\bar{S}_2\subseteq\bar{S}'$, thus $e\left(S_2,\bar{S}_2\right)\leq e\left(S_2,\bar{S}'\right)$. Now we have:

$$\begin{aligned}\frac{e\left(S'_2,S'_2\right)}{e\left(S'_2,\bar{S}'\right)} &\leq \frac{e\left(S'_2,S'_2\right)}{e\left(S_2,\bar{S}'\right)}\leq\frac{e\left(S'_2,S'_2\right)}{e\left(S_2,\bar{S}_2\right)}\\ &= \frac{e\left(S'_3,S'_3\right)+e\left(S_2,S_2\right)+e\left(S_2,S'_3\right)+e\left(S'_3,S_2\right)}{e\left(S_2,\bar{S}_2\right)}\\ &\leq \frac{c^2|S_2|+e\left(S_2,S_2\right)+|S_2|+|S_2|}{e\left(S_2,\bar{S}_2\right)}\\ &= \frac{\left(c^2+2\right)|S_2|}{e\left(S_2,\bar{S}_2\right)}+\frac{e\left(S_2,S_2\right)}{e\left(S_2,\bar{S}_2\right)}\end{aligned}$$

Lemma 21 can prove $\frac{e(S_2,S_2)}{e(S_2,\bar{S}_2)}$ is upper bounded by some positive constant, say a_1. Then, we have $\frac{e(S_2,S_2)}{e(S_2,\bar{S}_2)}\leq a_1$. Since G is a connected graph and $S_2\subseteq V$, we have $e\left(S_2,\bar{S}_2\right)+e\left(S_2,S_2\right)\geq|S_2|$. Therefore, we can get:

$$\begin{aligned}\frac{e\left(S'_2,S'_2\right)}{e\left(S'_2,\bar{S}'\right)} &\leq \frac{\left(c^2+2\right)|S_2|}{e\left(S_2,\bar{S}_2\right)}+\frac{e\left(S_2,S_2\right)}{e\left(S_2,\bar{S}_2\right)}\\ &\leq \frac{\left(c^2+2\right)\left(1+a_1\right)e\left(S_2,\bar{S}_2\right)}{e\left(S_2,\bar{S}_2\right)}+a_1\\ &= \left(c^2+2\right)\left(1+a_1\right)+a_1\end{aligned}$$

Until now, we have that $\frac{e(S'_1,S'_1)}{e(S'_1,\bar{S}')}\leq c^2$ and $\frac{e(S'_2,S'_2)}{e(S'_2,\bar{S}')}\leq\left(c^2+2\right)\left(1+a_1\right)+a_1$. Then, we have:

$$\begin{aligned}\frac{e\left(S',\bar{S}'\right)}{e\left(S',S'\right)} &= \frac{e\left(S'_1,S'_1\right)+e\left(S'_2,S'_2\right)}{e\left(S'_1,\bar{S}'\right)+e\left(S'_2,\bar{S}'\right)}\\ &\leq max\{\frac{e\left(S'_1,S'_1\right)}{e\left(S'_1,\bar{S}'\right)},\frac{e\left(S'_2,S'_2\right)}{e\left(S'_2,\bar{S}'\right)}\}\\ &= \left(c^2+2\right)\left(1+a_1\right)+a_1\end{aligned}$$

Finally, we have that Φ' is lower bounded by $\frac{1}{(c^2+3)(1+a_1)}$, which is a positive constant. □

Lemma 20 *Let G, G', S', $\bar{S}'$, c, S'_1, $e\left(X,Y\right)$ and F be the same as in the proof of Theorem 16. Then we have $\frac{e(S'_1,S'_1)}{e(S'_1,\bar{S}')}\leq c^2$*

Proof: Using the similar argument in the proof of Lemma 18, we can easily get the conclusion that $\frac{e(S'_1,S'_1)}{e(S'_1,\bar{S}')}\leq c^2$ □

Lemma 21 *Let G, G', Φ, Φ', V, S', $\bar{S}'$, c, S'_1, S'_2, S'_3, S_2, Y'_1, Y'_2, Y'_3, Y_2, $e(X,Y)$ and F be the same as in the proof of Theorem 16. Conditioned upon that $e(\bar{S}',\bar{S}')+e(\bar{S}',S') \geq 2(e(Y'_1,Y'_1)+e(Y'_1,S'))$, we have $\frac{e(S_2,S_2)}{e\left(S_2,\bar{S}_2\right)}$ is upper bounded by some positive constant.*

Proof: For convenience, we define the number of nodes in $\bar{S}_2$ to be β and define $x_1 = e\left(\bar{S}_2,\bar{S}_2\right)+e\left(\bar{S}_2,S_2\right)$, $x_2 = e(S_2,S_2)+e\left(S_2,\bar{S}_2\right)$, $x'_1 = e\left(\bar{S}',\bar{S}'\right)+e\left(\bar{S}',S'\right)$, $x'_2 = e(S',S')+e\left(S',\bar{S}'\right)$, and $x'_3 = e(Y'_1,Y'_1)+e(Y'_1,S')$. As shown in the proof of Theorem 16, we have $e(S',S') \leq e(\bar{S}',\bar{S}')$. Then we can get $x'_1 \geq x'_2$. And from the condition that $e(\bar{S}',\bar{S}')+e(\bar{S}',S') \geq 2(e(Y'_1,Y'_1)+e(Y'_1,S'))$, we have $x'_3 \leq \frac{1}{2}x'_1$. From the definition of x'_1 and all sets we have the following equation:

$$\begin{aligned} x'_1 &= e(Y'_1,Y'_1)+e(Y'_1,S')+e(Y'_3,Y'_3)+e(Y'_3,S')+e(Y_2,Y_2)+e(Y_2,S') \\ &\quad +e(Y'_1,Y'_3)+e(Y'_3,Y'_1)+e(Y'_1,Y_2)+e(Y_2,Y'_1)+e(Y'_3,Y_2)+e(Y_2,Y'_3) \end{aligned}$$

Using the similar argument in the proof of Lemma 19 for S'_1 and S'_2, we can get that there is no edge between Y'_1 and Y'_2. Then we have $e(Y'_1,Y'_3) = e(Y'_3,Y'_1) = e(Y'_1,Y_2) = e(Y_2,Y'_1) = 0$. It is easy to get that $e(Y_2,S') = e\left(\bar{S}_2,S'\right) = e\left(\bar{S}_2,S_2\right)+e\left(\bar{S}_2,S'_1\right)$. And the number of edges between $\bar{S}_2$ and S'_1 should be less than the number of nodes in $\bar{S}_2$, thus $e\left(\bar{S}_2,S'_1\right) \leq \beta$. Using the similar reason mentioned above for S'_3 in the proof of Lemma 19 again, consider Y'_3 can be partitioned into p groups. It is obvious that p is less than β. For each group in Y'_3, there exists only one kind of edges between this group and S'. They are the edges that connect a node in the group and a node in the same F as this group but belongs to S'. Therefore, the sum of the number of the edges between this group and S' and the number of edges in each group is less than the number of edges in each corresponding F. And since there is no edge between any two groups, we have: $e(Y'_3,Y'_3)+e(Y'_3,S') \leq p \cdot c^2 \leq \beta \cdot c^2$. The number of edges between Y'_3 and Y_2 is also at most the number of nodes contained in Y_2, so we have $e(Y'_3,Y_2) = e(Y_2,Y'_3) \leq \beta$. Combined with all conditions above, we get:

$$\begin{aligned} x'_1 &= e(Y'_3,Y'_3)+e(Y'_3,S')+e(Y_2,Y_2)+e(Y_2,S')+e(Y_2,Y'_3)+e(Y'_3,Y_2)+x'_3 \\ &= e(Y'_3,Y'_3)+e(Y'_3,S')+e\left(\bar{S}_2,\bar{S}_2\right)+e\left(\bar{S}_2,S_2\right)+e\left(\bar{S}_2,S'_1\right)+e(Y_2,Y'_3)+e(Y'_3,Y_2)+x'_3 \\ &\leq e(Y'_3,Y'_3)+e(Y'_3,S')+x_1+\beta+e(Y_2,Y'_3)+e(Y'_3,Y_2)+x'_3 \\ &\leq \left(c^2+3\right)\beta+x_1+x'_3 \\ &\leq \left(c^2+3\right)\beta+x_1+x'_1/2 \end{aligned}$$

From the definition of x'_2 and x_2, we have $x_2 \leq x'_2$. Since G is a connected graph, we have $\beta \leq x_1$. Therefore, $x_2 \leq x'_2 \leq x'_1 \leq 2\left(c^2+3\right)\beta+2x_1 \leq \left(2c^2+8\right)x_1$.

Here, we need to use the condition that graph G has a $O(\log n)$ mixing time. Consider two different cases. In the first case $x_2 \leq x_1$, which means that $e(S_2,S_2)+e\left(S_2,\bar{S}_2\right) \leq |E|$. Since the conductance being lower bounded by a positive constant is a both sufficient and necessary condition for $O(\log n)$ mixing time and graph G has a $O(\log n)$ mixing time, we can get that $\frac{e(S_2,S_2)}{e\left(S_2,\bar{S}_2\right)}$ is upper bounded by some positive constant directly. In the second case, $x_1 \leq x_2 \leq \left(2c^2+8\right)x_1$. In this case we have $e\left(\bar{S}_2,\bar{S}_2\right)+e\left(\bar{S}_2,S_2\right) \leq |E|$. Using the same reason mentioned in the first case, we have $\frac{e\left(\bar{S}_2,\bar{S}_2\right)}{e\left(S_2,\bar{S}_2\right)}$ is upper bounded by some positive constant, say c_1. Then $x_2/x_1 = \frac{e(S_2,S_2)+e\left(S_2,\bar{S}_2\right)}{e\left(\bar{S}_2,\bar{S}_2\right)+e\left(\bar{S}_2,S_2\right)} \leq 2c^2+8$, and we get:

$$\begin{aligned} \frac{e(S_2,S_2)}{e\left(S_2,\bar{S}_2\right)} &\leq \left(2c^2+7\right)+\left(2c^2+8\right)\frac{e\left(\bar{S}_2,\bar{S}_2\right)}{e\left(S_2,\bar{S}_2\right)} \\ &\leq \left(2c^2+7\right)+\left(2c^2+8\right)c_1 \end{aligned}$$

Finally, we have that $\frac{e(S_2,S_2)}{e\left(S_2,\bar{S}_2\right)}$ is upper bounded by a positive constant in the second case. □