Improving Social Network-based Sybil Defenses by Rewiring and Augmenting Social Graphs

Abedelaziz Mohaisen and Scott Hollenbeck

Verisign Labs, Reston, VA 20190

Abstract. Recent solutions to defend against the Sybil attack, in which a node generates multiple identities, using social networks. In these solutions, social networks are assumed to be fast mixing, and Sybil nodes—which disrupt the fast mixing property of social networks—are detected. Little is known about the cause of the mixing quality in social graphs, and how to improve it in slow mixing ones. In this work we relate the mixing time of social graphs to graph degeneracy, which captures cohesiveness of the graph. We experimentally show that fast-mixing graphs tend to have a larger single core whereas slow-mixing graphs tend to have smaller multiple cores. We then propose several heuristics to improve the mixing of slow-mixing graphs using their topological structures by augmenting them. We show that our heuristics greatly improve Sybil defenses.

Key words: Sybil Attacks and Defenses, Social Networks, Mixing Time.

1 Introduction

The Sybil attack is a very challenging security threat in distributed systems. In this attack, a single malicious node claims multiple identities with the intention of disrupting the normal operation of the distributed system by acting as if she is multiple nodes [6]. To defend against this attack, there are two schools of thoughts: centralized [3, 4, 6, 30] and decentralized solutions [13, 24, 28, 29]. In the centralized solutions, a centralized authority is used to provide digital credentials, such as cryptographic keys, and to bind them to the identity of participating nodes in the system. These credentials are then used to hold nodes accountable to their actions. While effective, those solutions are expensive: 1) they require an online authority that is hard to bring in distributed systems, 2) the authority would rely on privacy-sensitive information, like an identification number, physical address, and the like, and that would scare users away from using the system, 3) the existence of the centralized authority is very challenging to the scalability and security of the distributed system in general; an adversary would target that authority with attacks making it a potential bottleneck. Accordingly, centralized solutions are impractical in largely distributed systems.

Decentralized solutions replace the centralized authority with decentralized mechanisms. Nodes in distributed systems oftentimes have physical resources that are limited in nature, such as processing capabilities, memory, addresses, and geographical location, and these can be verified by other nodes in the distributed system to establish the identity of that node. Solutions that rely on those credentials overcome several of the shortcomings of the decentralized solutions: no single point of failure, and (mostly) no privacy concerns associated with the solution. However, these solutions work effectively on the premise that the adversary has a user-level resources: a powerful adversary can easily surpass such assumptions and gain control over more resources, thus bypassing the detection mechanism and introducing more Sybil identities in the system.

Recent solutions to the problem use social networks and their fabric of trust [4,10,15,17,28,29]. In such solutions, nodes are limited by their resources; the number of edges they create with others. To make such identities well enmeshed into the social graph, the adversary needs to create many edges between himself and the rest of the social graph, collectively representing honest users, which is associated with a high cost. Informally, many social network-based designs to defend against the Sybil attack rely on the mixing characteristics of social graphs for their operation. These designs assume that social networks are fast mixing, meaning that a short random walk from any node in the graph after a small number of steps will end up on a node that is randomly selected from the entire graph. The introduction of large number of Sybil identities hidden behind a few nodes that are connected by a few edges with the rest of the graph would violate this property: the honest and dishonest parts of the graphs are slow mixing. More formally, the prior literature on defending against the Sybil attack makes the assumption that a random walk of $O(\log n)$ steps, where n is the number of nodes in the social graph, is enough to obtain a sample that is driven from a distribution close to the stationary distribution of the random walk.The theoretical guarantees of Sybil defenses and their practicality rely greatly on such parameters: the number of tolerable Sybil identities per an attack edge, an edge connects an honest node with a malicious node, is proportional to the random walk length that is considered the mixing time.

Researchers recently demonstrated that the mixing time of social graphs is slower mixing used in the literature, and showed several immediate findings [22]. First, the theoretical guarantees that make use of certain qualities of the mixing time of social graphs are inaccurate, since the property does not hold in these graphs as being assumed. Second, although the mixing time is larger than expected, Sybil defenses still work fairly reasonably on many of the graphs with the relatively large mixing time, indicating that a more relaxed property than the one used in the theoretical reasoning about the operation of Sybil defenses. Finally, different graphs have different quality of the mixing time, and in certain graphs—which are mostly the result of face-to-face interactions—the slow mixing prohibits the applicability of Sybil defenses on them [17,22].

The main intuition behind the quality of the mixing time is hypothesized to be the community structure in them: whereas face-to-face graphs have slower mixing characteristics because of their clear community structure, online social networks have faster mixing times because they are likely subject to noise and weak social ties, resulting in flat and less clear community structures. However, no prior work tested this intuition to show its validity in social networks used for building such applications. Yet more importantly, no prior work used the inherent properties of slower mixing social graphs to improve their mixing time, and make them more suitable for such applications as Sybil defenses. To this end, we set out to investigate the reasons why certain graphs are slower mixing than others. We use our findings on why certain social graphs are slower mixing to improve their mixing time, and thus improve the security of social network-based systems when operated on top of them.

▶ **Contributions.** Motivated by the lack of prior work on understanding the mixing time of social graphs, our contribution is two-fold. First, we explore understanding the mixing time of social graphs by identifying why some social graphs are fast mixing whereas others are slow mixing. We relate the quality of the mixing characteristics of social graphs to the degeneracy (coreness) of graphs: we find that whereas slow mixing graphs have multiple small cores, fast mixing graphs have a single large core. Second, we use this observation to propose several heuristics that utilize the structure of slow mixing social graphs to augment them and to improve their mixing characteristics. We show that the improvement in the mixing time affects Sybil defenses built on top of social graphs. In one particular heuristics, we are able to reduce the overhead of operating SybilLimit by more than 55% and the security by more than 70%.

▶ **Organization.** The rest of this paper is organized as follows. In Section 3, we review preliminaries used in this work; including the graph model and the formal definition of the mixing time, as long as the k-coreness, the main metric used for understanding the mixing time. In Section 4, we present measurements on relating the mixing time of social graphs to core structure followed by heuristics to improve the mixing time in section 5. Conclusion is in section 6.

2 Related Work

To the best of our knowledge, there is no prior work on understanding the mixing time, improving the mixing time of slow mixing social graphs, and studying the impact of that on the operation of Sybil defenses, except for our preliminary work in [19] which is limited to the first part. Concurrent to this work, rewiring of social graphs to improve the mixing time is proposed in [31], without identify reasons why some social graphs are slow mixing, and without considering the context of Sybil defenses for the improvement. The rest of the related work can be broadly classified into three veins: wok that investigated improving Sybil defenses, although in different ways than our approach, related work on measuring the mixing time, and work on building new Sybil defenses.

Our prior work in [17] improves the performance of Sybil defenses by accounting for trust, not the underlying honest social graphs: selection on nodes and edges in Sybil defenses are biased based on differential trust. Measuring the mixing time and testing whether assumptions widely used for building Sybil defenses are valid or not is studied in [22] and independently in [5], although findings and conclusions of both studies are at odd. Looking into building tools to measure the mixing time of directed graphs and the security implications on the operation of Sybil defenses, as well as anonymous communication systems built on top of social networks and using the mixing time, is done in [21]. Finally, mathematical tools and theoretical bounds that characterize the mixing time and are used for measuring it are proposed in [8, 25].

As pointed out in the introduction, there has been several works on the design of defenses that make use of the mixing time of social networks, some of which use it directly and require some strict quality for their theoretical guarantees to be achieved while others merely require it as a crude property to conceptually capture an identifying and distinguishing characteristic between Sybil and honest nodes. Those works include [3, 4, 11, 16, 17, 22, 28, 29], among others—an interesting survey of such works is provided in [27]. Our findings in this paper, although test on SybilLimit, are easily extendable to other designs. Indeed, our heuristics provided in this paper do not alter the operation of SybilLimit, but rather the underlying social graph to improve its mixing characteristics. To this end, we expect that one can easily extend the results and findings on improving the operation of other Sybil defenses using the same mechanisms.

3 Preliminaries

3.1 Graph model

Let $G = (V, E)$ be an undirected and unweighted graph over n vertices and m edges, where the set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and the set of edges $E = \{e_{ij}\}$ for every $v_i \sim v_j$ (v_i is adjacent to v_j). For G, let $P = [p_{ij}]^{n\times n}$ be a transition probability matrix s.t. $p_{ij} = 1/\deg(v_i)$ if $v_i \sim v_j$ (i.e., if $e_{ij} \in E$) and 0 otherwise. k_x denotes a fully connected graph on x vertices.

▶ **The mixing time.** Informally, the mixing time is the length of a random walk to reach a constant distance from the stationary distribution of that walk, when starting from any node in the graph. The stationary distribution is defined as a probability distribution for any node to be selected as the final node in a random walk after an infinite number of steps. For the same graph defined above, the stationary distribution is defined as $\pi = [\pi_i]^{1\times n}$, where $\pi_i = [\deg(v_i)/2m]$ for $1 \leq i \leq n$. Formally, the mixing time is defined as

$$T(\epsilon) = \max_j \min_t \{t : ||\pi - \pi^{(j)} P^t|| < \epsilon\}, \tag{1}$$

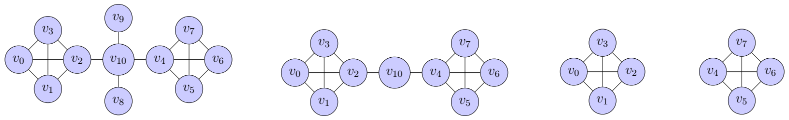
where $\pi^{(j)}$ is the delta distribution (also known as the Kronecker delta function) concentrated on the j-th position in that vector. This is, $\pi^{(j)}$ is defined as $\pi^{(j)} = \delta[x]$ where $\delta[x] = 0$ if $x \neq j$ and 1 otherwise, and the norm $||\cdot||$ in (1) is defined as $||\pi - \pi^{(j)}|| = \frac{1}{2}\sum_{i=1}^{n} |\pi_i - \pi_i^{(j)}|$.

In the literature, two methods are used for measuring the mixing time of social graphs. The first method uses the definition in (1); given that the mixing time converges as the sample of the starting distribution increases, and because the property of interest in many social network-based Sybil defenses is the distribution over ϵ, rather than a fixed ϵ as defined for the largest t in (1), one can start from a random set of nodes and obtain different values of ϵ as t increases. The different values of ϵ can be used to measure its distribution and characterize the mixing time of social networks. On the other hand, the second method for measuring the mixing time makes use of the second largest eigenvalue of the the matrix P defined above, and only provides an upper and lower bound on the mixing time as defined in (1).

Graphs are either fast or slow mixing depending on how quickly walks on them reach the stationary distribution [22] (i.e., how large is t for a given ϵ in the model in (1)). It has been claimed that the mixing time does not relate to any of the graph structural properties, making the mixing time interesting in its own right [5]. We re-examine this claim, and find that mixing characteristics of a graph are closely related to the core structure, which captures graph cohesiveness. We show that fast mixing graphs have large *single* core, whereas slower mixing graphs have multiple small cores and use that observation to propose several herustics to improve the mixing time.

3.2 *k*-coreness

For the undirected graph G we defined in § 3.1, let k be a parameter such that $k \geq 1$. We define the graph $G_k = (V_k, E_k)$, where $V_k = \{v_k^i, \ldots, v_k^{n_k}\}$, and $E_k = \{e_{ij}\}$ for all $v_k^i \sim v_k^j \in V_k$, to be a subgraph in G such that $|V_k| = n_k$, $\min\{\deg(v_k^i)\} \geq k$ for all $v_k^i \in V_k$. The subgraph G_k is said to be a k-core of G if it satisfies the above degree condition, it is maximal in size, and it is a connected graph. By relaxing the connectivity condition, we obtain a set of cores (potentially more than one), each of which satisfies the degree condition. For such k-core, we define the normalized size as $s_k = n_k/n$. Formally, G_k consists of $t_k \geq 1$ components denoted as $\{G_k^1, G_k^2, \ldots, G_k^{t_k}\}$. We denote nodes that are in G_k^i as $v_k^{i1}, \ldots, v_k^{i|V_k^i|}$. We refer to the largest connected G_k^i as the major core and others for a given k as minor cores.



(a) G_1 (b) G_2 (c) G_3

Fig. 1. An illustration of the $k-$core decomposition of the graph. The original graph $G = G_1$ is shown in 1(a). Notice that G_4 is an empty graph, which results from trimming nodes in G_3, which is shown in Figure 1(c), with degrees ≤ 3.

An example illustrating the definition stated above is shown in Figure 1 of a graph over 11 nodes and 14 edges, with a lowest degree of 1 and highest degree of 3, thus the $k-$core number of the graph is less than or equal to 3. By recursively omitting nodes in G with degree less than or equal to 1, we get G_2, shown in Figure 1(b). Similarly, omitting nodes with degree less than or equal to 2 in Figure 1(b) produces G_3 shown in Figure 1(c), which consists of two components, each of which is a fully connected graph defined over 4 nodes. Given that they are equal in size, either of the components can be considered as the major component, and the other is considered as the minor component. Given that the highest degree in G_3 is 3 as well, the original graph has a maximum k of 3, and this dissolves entirely when omitting nodes with degree less than or equal to 3. Computing the k-cores of a graph for any k is done efficiently using off-the-shelf algorithms. An efficient algorithm for decomposing a simple graph on m edges and n nodes to its $k-$cores by iteratively pruning nodes with degree less than k has the complexity of $O(m)$ [2]. To this end, the overall complexity of running algorithms described in the rest of this paper is linear in both the maximum k value and the number of edges in the graph G. Finally, notice that the definition of k-core [14] is related to k-coloring [7], and thus can be naturally connected to the connectivity of the graph.

4 Measurements and Results

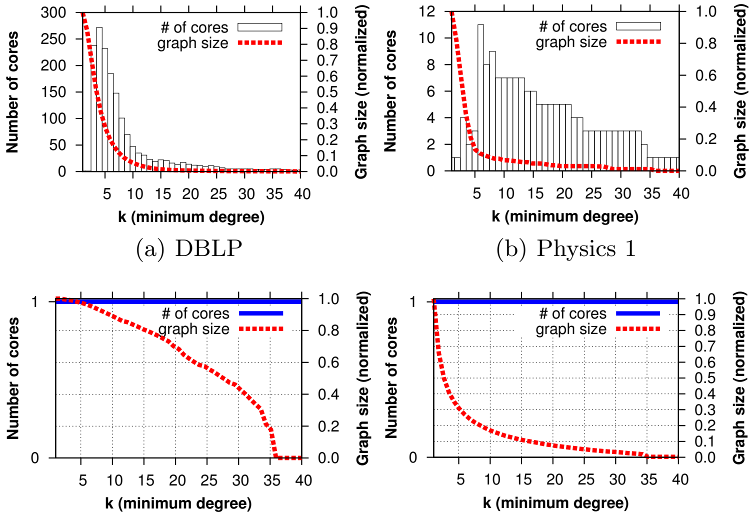
▶ **Datasets.** We use the datasets in Table 1 in our measurements. All of these datasets are widely used as benchmarking graphs in the literature [3,12,18,19,21]. The datasets DBLP, Physics 1, and Physics 2 are scientific collaboration graphs, and can be considered of the same type, while Slashdot is a blog following graph, and Wiki-vote is wikipedia's admin voting graph. In DBLP, nodes represent authors while edges indicate that two authors have a co-authored paper among them. The Slashdot dataset consists of users and as nodes and an edge between two nodes indicates that the first node follows an article by the second node (we omit directions as below). Finally, Wiki-vote is the wikipedia administrators voting dataset in which the first node has voted for the promotion of the second node to become an administrator. Some of these datasets are directed (i.e., Slashdot and Wiki-vote), so we follow the literature [3,12] and convert the directed graphs to undirected ones, by considering an edge between two nodes in the undirected graph if it exists in either direction in the directed one.

▶ **Measuring the mixing time.** We use the definition in (1) to compute ϵ for a varying t when starting walks from different nodes in the graph. For feasibility, we sample the initial distributions of the walks: we start from 1000 uniformly distributed nodes in each graph and compute the mixing time as per the definition in (1) and the average ϵ for each walk length. The mixing characteristics of these graphs are shown in Fig. 2—maximum in Fig. 2(a) and average in Fig. 2(b).

Table 1. Datasets used in experimentation and validation.

Dataset	# nodes	# edges
DBLP	769,641	3,051,127
Slashdot	70,355	459,620
Physics 2	11,204	117,619
Physics 1	4,158	13,422
Wiki-vote	1,300	36,529

For each graph in Table 1, we use an off-the-shelf linear-time algorithm [2] to compute the $k-$core. As k increases to its ultimate value at which the graph diminishes, we compute the following: (1) the number of cores in each k-core, (2) the normalized size of each k-core. Results are shown in Fig. 3. Notice that graphs in Fig. 3(a)-3(b) are slow mixing and graphs in Fig. 3(c)-3(d) are fast

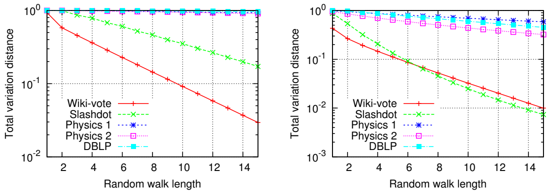


(a) DBLP (b) Physics 1

(c) Wiki-vote (d) Slashdot

Fig. 3. The core structure of slow mixing (3(a) and 3(b)) versus fast mixing (3(c) and 3(d)) graphs. The slow mixing graph dissolves into multiple cores as k, the minimum degree in the $k-$core algorithm increases, unlike fast mixing graphs which consist of a single large core.

mixing, as demonstrated in Fig. 2 for both the maximum and average mixing time cases.



(a) Max (b) Mean

Fig. 2. Mixing time measurement of graphs in Table 1.

By comparing Fig. 3(a), Fig. 3(c), and Fig. 3(d), we observe that slow mixing graphs are less cohesive whereas fast mixing graphs are more cohesive. This observation is reflected in the number of cores in the k-core of each graph as we increase k until the graph is dissolved entirely. Also, whereas slow mixing graphs—shown in Fig. 3(a) and Fig. 3(b)—are decomposed into multiple cores as we increase k, fast mixing graphs resist this decomposition and remain cohesive as k increases, even for larger k than in the slower mixing graphs.

Second, even though slow mixing graphs are decomposed into multiple cores, these cores are relatively small in size and the graph dissolves quickly as k increases. Fast mixing graphs on the other hand remain in a single core, which is relatively larger in size than the counterpart core in slow mixing graphs

5 Improving the Mixing Time and Sybil Defenses

With a different motivation, there has been several attempts in the literature to design algorithms that improve the mixing time of random walks on social graphs [1,9]. The main motivation of these designs is to provide a better method for sampling large graphs and to obtain representative samples of the large population in these graphs [9,18,23]. However, these solutions fall short in providing the desirable features for Sybil defenses. For example, existing solutions that improve the mixing characteristics of social graphs by providing uniform teleportation probability to any node in the graph at any step in the random walk are expensive [1], since they require each node to know the entire social graph. More importantly, these designs are impractical for Sybil defenses, which use the mixing time for their operation. This impracticality comes from the fact that these algorithms will ultimately improve the mixing characteristics of both honest and dishonest nodes arbitrarily, since the probability of choosing an honest and a Sybil node in the graph as a next step of the random walk due to the teleportation is equal, even when the algorithm is performed in a centralized fashion. Notice the latter shortcoming can be prevented if the label of destination is known in advance. However, deviation based on the label would of the nodes will reduce the effectiveness of the algorithm by not achieving the claimed improvement in the mixing time in aforementioned work.

As we have shown in the previous section, the mixing characteristics of social graphs, which influences the operation of Sybil defenses on top of social networks, depend on the core structure of these graphs. Slower mixing graphs tend to have multiple cores as the parameter k increases, whereas fast mixing graphs resist dissolution and consist of a single core as k increases. Using this observation, we proceed to describe several heuristics to improve the mixing time, and ultimately improve the operation of Sybil defenses on top social networks. The main goal of these heuristics is to prevent the dissolution of social graphs into multiple cores, thus improving its connectivity in a meaningful way. Our work is different in both objective and tools we use, and is tailored for random walks on social graphs.

5.1 Heuristics to Improve the Mixing Time

From our previous measurements we observe that as k increases the graph dissolves into multiple cores, particularly in slow mixing social graphs. Accordingly, we refer to the largest core for a given k value as the *main core*, and other cores as *minor cores*. In each of the following heuristics we aim to improve the mixing time by preventing the creation of multiple cores as k increases using auxiliary edges. We call this process of adding edges as *core wiring*. We introduce these heuristics with sybil defenses in mind as potential applications.

Heuristic X-1-C. The intuition here is to add edges so that only prevent the dissolution of the graph into multiple cores as k increases. Accordingly, for $G_k = \{G_k^1, \ldots, G_k^{t_k}\}$ ($k \geq 1$ and $t_k > 1$), where G_k^1 is the main core and G_k^i for $i \geq 2$ is a minor core, we add an edge between only one random node v_k^{ij} in the minor core G_k^i (where j is chosen at random) and v_k^{1l} in the major core G_k^1. We repeat that process as k increases to its ultimate value upon which the whole graph diminishes. The total number of added edges in the original graph G is $[(\sum_{k=1}^{k_{max}} t_k) - k_{max}]$, where k_{max} is the largest k of a core in G. **Heuristic X-A-C.** An illustration of the operation of the heuristic is highlighted in Figure 4.

As k increases, for $k = 1$ and $k = 2$, the resulting graph is a single component, so no edges are added to it. However, when $k = 3$, the graph dissolves into two components, as shown in Fig. 1(c). Thus, two nodes are randomly selected; one from each component (i.e., one is from the major and the other is the minor component, which these labels used interchangeably for the graph in G_3). Then an edge is created between the two selected nodes, which is the dotted edge in Figure 4, created between node v_5 and v_2.

Unlike in the previous heuristic where only dissolution prevent measure is taken to improve the connectivity of the graph, in this heuristic we aim to further improve the connectivity by adding multiple edges that would improve resilience of the graph to the removal of edges in between of different components. The heuristic accordingly adds multiple edges between each component in the k-core graph, as k increases. This is, for $G_k = \{G_k^1, \ldots, G_k^{t_k}\}$ ($k \geq 1$ and $t_k > 1$), where G_k^1 is the main core and G_k^i for $i \geq 2$ is a minor core, we add an edge between every node v_k^{ij} in the minor core G_k^i and random nodes v_k^{1l} in the major core G_k^1. We repeat that process as k increases to its ultimate value upon which the whole graph diminishes. The total number of added edges in the original graph G is $[(\sum_{k=1}^{k_{max}} t_k) - k]$.

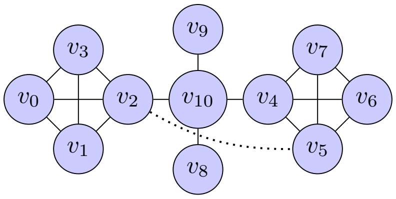


Fig. 4. An illustration of how the $k-$core decomposition of the graph is used to improve graph connectivity using X-1-C.

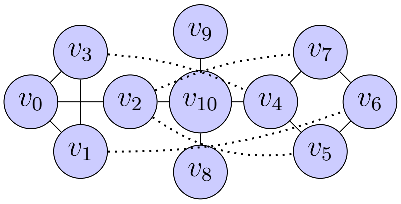


Fig. 5. An illustration of how the $k-$core decomposition of the graph is used to improve graph connectivity using X-A-C.

An illustration of this heuristic is applied to the graph in Figure 5. In this example, and without losing generality, recall that only G_3 dissolves into multiple components, and requires addition of edges to prevent such dissolution, as shown in Figure 1(c), according to the heuristic X-A-C. Also, without losing generality, let $G^1{}_3 = (V^1{}_3, E^1{}_3)$ where $V_3{}^1 = \{v_4, v_5, v_6, v_7\}$ be the minor core and let $G^2{}_3 = (V^2{}_3, E^2{}_3)$where $V_3{}^2 = \{v_0, v_1, v_2, v_3\}$ be the major core. In this heuristic, every node in G_3^2 is associated with a node in the major core G_3^2, where the latter node need not to be unique.

▶ **Heuristic X-A-A.** In this heuristic, we aim to further enmesh nodes in different cores together by adding edges across cores, not only between nodes in the major and the minor cores. To this end, we wire all nodes in a minor core to other cores in the graph, including both minor and major cores. The number of auxiliary edges is bounded by the *order* of the number of nodes in each k-core. However, to avoid undesirable complexity in the operation of the heuristic, we first sort all components in a given graph G_k, for any valid k, with respect to their size (the number of nodes each component has). Then, we wire nodes in the smaller component with nodes in the bigger component only.

▶ **Heuristic X-A-A$^+$.** As we have seen in the previous proposed heuristics, additional edges are added to the graph in order to prevent its dissolution as k increases. These added edges can be viewed as a cost associated with the operation of these heuristics, and it is desirable to reduce this cost. Indeed, one desirable modification to the previous heuristic is graph rewiring. At each time an edge is added between two nodes in two different components, an edge is removed from either component (for that, we remove edges from the minor component, or the component with the smaller size when the number of minor components is greater than one). Desirably, we remove edges that constitute triangles within that component, and stop the process of rewiring the graph when we exhaust all triangles in that component. This approach is similar to the concurrent work in [31], although the strategy used for rewiring edges is different. Notice that this heuristic preserves the number of edges in the original graph, and would rewire nodes instead of adding edges. This heuristic provides the highest improvement, which suggests that adapting this strategy to other earlier heuristics will also improve them.

5.2 Practical considerations

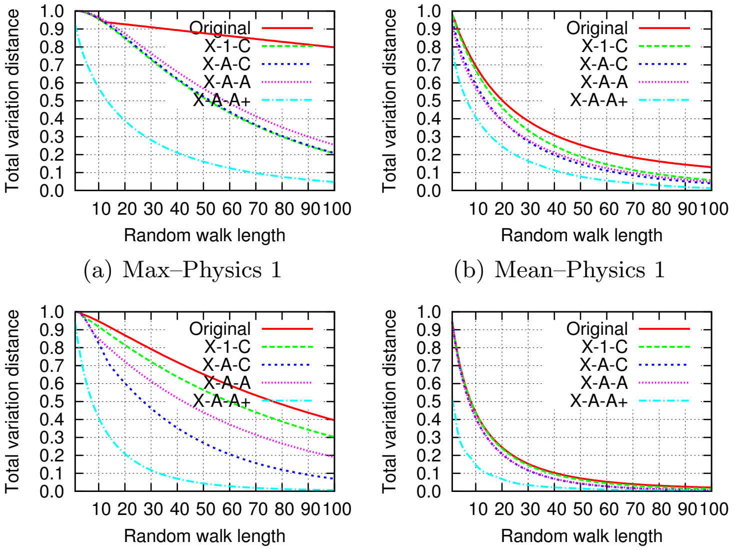
Two issues have a great influence on the operation of the proposed heuristics in this paper that could possibly limit their practicality. In the following, we raise these issues as questions and subsequently answer them. First, what is the rationale of using such edges, particularly when the number of edges is large? Second, what is the guarantee that edges are not going to be created between good nodes and Sybil nodes, thus improving the mixing time not only for the honest region of the graph, the region that includes honest nodes only, but also the dishonest region of the graph as well?

We address the first issue by pointing out two practical considerations. First, such edges can be made as a part of the natural evolution of the underlying social graph; by incorporating them into a link recommendation system where that is possible. Ultimately, not all links will be added to the graph, but some of them that would be created and such links would be of great importance to the connectivity of the graph. Second, since the operation of Sybil defenses on top of social networks does not require a real existence of links between nodes, but rather the flow of the walk on these links—which makes these edges virtual, we claim that such edges can be created virtually, but not in reality. This is, when a random walk is originated from a node on the graph, the random walk would deviate at that point from the one done on the original graph by assigning transition probability to the walk towards nodes connected via the virtual edges.

This part of practical consideration of our approach is in a sense similar to the prior work that adds a random teleportation probability of the random walk to improve the mixing time [1, 9]. However, our approach will limit the number of nodes this teleportation would be assigned to (bounded by the number of the added edges a node in the graph would be a part of), thus no prior knowledge of the entire graph is done. However, the probability assigned to each node that is not connected to a given node have to be given in advance to that node. These probabilities (practically, they will be identifiers of nodes to which random walks are then propagated) can be distributed in the initialization phase of the Sybil defense, which can be done in a centralized manner.

To address the second issue, we use the existing reasoning in the literature which considers pre-existing labels of nodes in social graph to operate social network-based Sybil defenses [3, 17, 26]. For example, some of the prior work in the literature has assumed a predetermined labels of honest and Sybil nodes to improve the operation of Sybil defenses by incorporating weights on existing edges between some nodes in a more favorable way than others [17]. On the other hand, some work has indeed used a pre-determined list of labeled honest nodes to start the operation of the Sybil defense and to rank other nodes as either honest or Sybil [3, 26]. To address the second issue, we claim that one can create edges, or add the transition probability as described previously for virtual edges, between only previously labeled honest nodes, thus improving the mixing time of the honest region of the graph but not the Sybil one.

In conclusion, auxiliary edges added in our heuristics can be made part of the evolution of the social graph through link recommendation. Alternatively, when centralized initialization is viable, these edges can be virtually created among honest nodes only if some of the nodes are labeled, as used previously.



(a) Max–Physics 1

(b) Mean–Physics 1

(c) Max–Physics 2

(d) Mean–Physics 2

Fig. 6. Mixing time measurement of Physics 1 and Physics 2.

5.3 Results and Discussion

We select Physics 1 and Physics 2, two of the slow-mixing and relatively small social graphs to explore the potential of our heuristics in improving the mixing time for slow-mixing social graphs. We emphasize that the main reason to choose those social networks is their size, which enabled us to compute the mixing time using the definition in (1) from all nodes in each of the graphs. The results of measuring the mixing time after applying the heuristics in section 5.1 for all possible initial distributions are in Fig. 6; Fig. 6(a) and 6(b) are for the Physics 1 dataset whereas Fig. 6(c) and 6(d) are for Physics 2. The total number of edges before and after wiring graphs using the different methods explained earlier is shown in Table 2. Notice that the total number of nodes is still the same as in Table 1, and the number of edges in X-A-A$^+$ is preserved as in the original graph. In the following we elaborate on how the different heuristics affect the mixing time and the performance of Sybil defenses on top of them.

Table 2. Datasets used in this study.

Dataset	Number of edges (total)			
	Orig.	X-1-C	X-A-C	X-A-A
Physics 1	13,422	13,544	16,482	25,064
Physics 2	117,619	117,687	119,082	121,169

Heuristics impact on the mixing time By comparing the different plots in Fig. 6, it is obvious to see that the heuristics improve the mixing time, and in some cases greatly, for both the average and the minimum time. Particular, we first observe that our simplest heuristic (X-1-C), which produces minimal effect on the graph density—only 122 edges are added to Physics 1—significantly improves the mixing time according to its definition as the *maximal* walk length for a given total variational distance.

Second, the extent to which additional edges improve the mixing time differs and depends on the initial mixing characteristics of the graph. For example, X-1-C adds 68 edges to Physics 2 graph, which exhibits almost no effect on the mixing time, as shown in Fig. 6(d). The original graph already mixes better than Physics 1 dataset on average, and the addition of these edges, although improves the slowest mixing sources, does not improve a lot on average. Finally, by considering the number of added edges in X-A-A in both social graphs and the measured mixing time after adding these edges, we observe that the addition of a lot of edges—despite improving the density of the graph—does not improve the mixing time significantly (sometimes yields worse mixing as in Fig. 6(c)). This last remark tells us that auxiliary edges need to be placed wisely in graph in order to improve the mixing time.

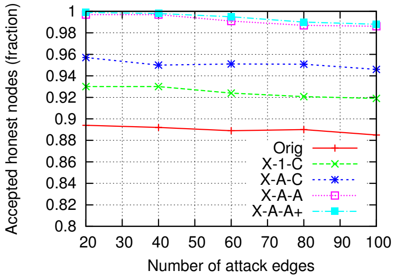


Fig. 7. Accepted honest nodes for varying attack edges.

This is further made clear by observing how rewiring the graph in X-A-A$^+$ improves the mixing time, despite maintaining the same number of edges as in the original graph. We attribute that effect on the performance to the inherent changes added on the graph to enmesh nodes in it, and reduce the number of loops within community (core) that would diverge the random walks.

5.4 Heuristics impact on Sybil defenses

We implement and run SybilLimit [28] over the augmented social graphs, according to the heuristics described earlier, in order to improve their mixing characteristics. In the following, we use describe SybilLimit, then provide our results and findings.

▶ **SybilLimit.** In SybilLimit, each node samples r edges in the graph as "witnesses", where $r = r_0\sqrt{m}$, by running r independent instances of random walks each of length $w = O(\log n)$, which is the mixing time of the social graph. Accordingly, there is an overwhelming probability that the sampled subsets of honest nodes in the social graph will have a non-empty intersection, which would be used for suspect verification. Formally, if the social graph is fast mixing—i.e., has a mixing time of $O(\log n)$—then probability of the last node/edge visited in a walk of length $O(\log n)$ drawn from the edge/node stationary distribution is at least $1 - \frac{1}{n}$. Accordingly, by setting r_0 properly, one can use the birthday paradox to make sure that the intersection between two sampled subsets of edges (by two honest nodes) is non-empty with an overwhelming probability. Furthermore, given that the social graph is fast mixing, and the number of attack edges—edges that connect Sybil with honest nodes—is limited, the probability for random walks originated from honest region ending up to the dishonest region is limited. Chances of dishonest nodes being accepted by sampling honest edges is limited, and bounded by the number of attack edges.

▶ **Results.** To evaluate the performance of SybilLimit when operated on the original and modified graph, we use both the number of accepted honest suspects by honest verifiers when using a fixed walk length and varying number of attack edges, and the number of accepted Sybil nodes introduced in total for the same settings as earlier. We used a random walk length of 16 for the first two heuristics, and notice that a walk length of 38 on the original graph is sufficient to accept 97% of the honest suspects by honest verifiers [22]. Because X-A-A$^+$ improves the mixing time significantly more than other heuristics, we measure the proper walk length that makes more than 99% honest nodes accepted by honest verifiers, and find that to be a walk length of 7 which we use for that experiment only. Results are shown in Fig. 7 and Fig. 8, where Fig 7 shows the number of accepted honest nodes by honest verifiers while varying the number of attack edges, and Fig 8 shows the number of accepted

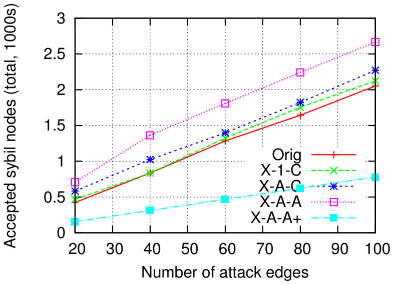


Fig. 8. The performance of SybilLimit: accepted sybils under varying number of attack edges.

Sybils while varying the number of attack edges for the different heuristics on the original graph.

In this measurement we observe that (among the first three heuristics) X-A-A accepted the most honest users followed by X-A-C and X-1-C, which is anticipated given their consistent order with respect to their modified density as shown in Table 2 and the mixing characteristics as in Fig. 6. However, and as anticipated given the theoretical interplay of the mixing characteristics and security guarantees of Sybil defenses, X-A-A also accepted significantly more Sybil nodes than others, given its improved mixing time. Interestingly, until the number of attack edges is 40, X-1-C does not increase the number of accepted Sybil nodes, while increasing the number of accepted honest nodes by honest verifiers by around 3.5%. Comparing the Sybil defense when using the three different heuristics, and that of X-A-A^+, we find that the latter heuristic outperforms them all by accepting most honest nodes and the least of Sybils.

6 Conclusion

In this work we explored understanding and improving the mixing characteristics of social graphs. We pointed out that the mixing characteristics of social graphs are related to the core structure, and used that to improve the mixing time. Using a running example, we demonstrated that the improved mixing time affects Sybil defenses, such as SybilLimit, although findings can be applied to other defenses. In the future, we will also look at measures to identify wider range of the quality of the mixing characteristics, as opposed to both extremes of the maximum and average explained in this work.

Acknowledgement. Part of this work appears in our unpublished prior study in [20]. The first author would like to thank Y. Kim, N. Hopper, and H. Tran for their help with the prior work.

References

1. K. Avrachenkov, B. F. Ribeiro, and D. F. Towsley. Improving random walk estimation accuracy with uniform restarts. In *WAW*, 2010.
2. V. Batagelj and M. Zaversnik. An $O(m)$ algorithm for cores decomposition of networks. Arxiv preprint cs/0310049, 2003.
3. Z. Cai and C. Jermaine. The latent community model for detecting sybil attacks in social networks. In *NDSS*, 2012.
4. G. Danezis and P. Mittal. SybilInfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.
5. M. Dellamico, , and Y. Roudier. A measurement of mixing time in social networks. In *IWSTM*, 2009.
6. J. R. Douceur. The sybil attack. In *IPTPS*, pages 251–260, 2002.
7. P. Erdős and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica*, 17(1), 1966.
8. M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of the permanent resolved. In *STOC*, pages 235–244. ACM, 1988.

9. C.-H. Lee, X. Xu, and D. Y. Eun. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In P. G. Harrison, M. F. Arlitt, and G. Casale, editors, *SIGMETRICS*, pages 319–330. ACM, 2012.
10. C. Lesniewski-Laas. A Sybil-proof one-hop DHT. In *Proceedings of the 1st workshop on SNS*, pages 19–24. ACM, 2008.
11. C. Lesniewski-Laas. *Design and Applications of a Secure and Decentralized Distributed Hash Table.* PhD thesis, Massachusetts Institute of Technology, 2010.
12. C. Lesniewski-Lass and M. F. Kaashoek. Whānau: A sybil-proof distributed hash table. In *NSDI*, pages 3–17, 2010.
13. F. Li, P. Mittal, M. Caesar, and N. Borisov. Sybilcontrol: practical sybil defense with computational puzzles. In *Proceedings of the seventh ACM workshop on Scalable trusted computing*, pages 67–78. ACM, 2012.
14. D. Lick and A. White. k-Degenerate graphs. *Canadian J. of Mathematics*, 22:1082–1096, 1970.
15. P. Mittal, M. Caesar, and N. Borisov. X-Vine: Secure and pseudonymous routing using social networks. In *NDSS*, 2012.
16. A. Mohaisen, N. Hopper, and Y. Kim. Designs to account for trust in social network-based sybil defenses. In *ACM CCS*, 2010.
17. A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust in social network-based sybil defenses. In *INFOCOM*, 2011.
18. A. Mohaisen, P. Luo, Y. Li, Y. Kim, and Z. Zhang. Measuring bias in the mixing time of social graphs due to graph sampling. In *MILCOM*, 2012.
19. A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. Understanding social network properties for trustworthy computing. In *SIMPLEX*, 2011.
20. A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. Understanding the mixing patterns of social networks. Technical report, UMN, 2011.
21. A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. On the mixing time of directed social graphs and implications. In *ACM ASIACCS*, 2012.
22. A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *IMC*, pages 383–389. ACM, 2010.
23. B. F. Ribeiro and D. F. Towsley. Estimating and sampling graphs with multidimensional random walks. In *IMC*, 2010.
24. L. Shi, S. Yu, W. Lou, and Y. T. Hou. Sybilshield: An agent-aided social network-based sybil defense among multiple communities. In *INFOCOM*, 2013.
25. A. Sinclair. Improved bounds for mixing rates of mc and multicommodity flow. *Comb., Prob. & Comp.*, 1:351–370, 1992.
26. C. Yang, R. C. Harkreader, J. Zhang, S. Shin, and G. Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *WWW*, 2012.
27. H. Yu. Sybil defenses via social networks: a tutorial and survey. *SIGACT News*, 42(3):80–101, 2011.
28. H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social net defense against sybil attacks. In *S&P*, 2008.
29. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: defending against sybil attacks via social networks. In *SIGCOMM*, 2006.
30. H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *IEEE Symposium on Security and Privacy*, 2009.
31. Z. Zhou, N. Zhang, Z. Gong, and G. Das. Faster random walks by rewiring online social networks on-the-fly. In *ICDE*, 2013.