Towards Trustworthy Computing on Social Networks

Measurements and New Applications

Abedelaziz Mohaisen
University of Minnesota – Twin Cities
Minneapolis, Minnesota 55455
mohaisen@cs.umn.edu

ABSTRACT

In this work we study social networks by measurements and verification of assumption widely and blindly used for building security and communication services on top of these networks. We measure the mixing time, the expansion, and the betweenness in social graphs. Our initial findings show that some properties do not hold, whereas others hold with less quality than assumed in the literature. From there, we propose to proceed to study these properties in different settings. We propose to study one of these widely properties, the mixing time, under several conditions used widely as practices for modifying social graphs: graph sampling and omission of graph directions. We explore reasons behind the quality of the mixing time, and propose several heuristics to improve it.

Motivated by the lack of work on accounting for differential trust in social network-based applications—Sybil defenses in particular, we develop four designs and use them to account for trust in a benchmark technique of Sybil defense (SybilLimit). We show empirically that our designs improve the security of this defense at some reasonable cost. We propose to extend these algorithms to other applications, Sybil defenses as well as other routing and information dissemination applications on top of social networks and rely on trust and social graphs' structure in their operation.

Finally, we propose to use social networks for building two types of applications: a distributed computing service (called social cloud) that does not make any direct use of a global property of social graphs (such as the mixing time or the expansion), and an anonymous communication service that incorporates dynamics in the social graph to improve anonymity.

Keywords

Social networks, trustworthy computing, algorithmic properties, the mixing time, Sybil defenses, anonymous communication, measurements, design.

Bibliographical Information

The work in section 2 is based on the published works in [106, 93, 103] and the unpublished technical report in [104]: the work in section 2.2 and section 2.4 is partly based on [106], the work in section 2.5 is partly based on [103], the work in section 2.6 is partly based on [103] and [93]. The proposed work in 2.7 is based on two ongoing works with working titles in [98] and [99]. The main work in section 3 is based on the published work in [97], and the proposed work in section 3.3 is to be initiated. The work in 4 is based on the unpublished technical report [101] (currently submitted for review) and a working title in [105]. Proposed work in 4.1.15 is to be initiated and the main work in section 4.2 is to be initiated based on the preliminaries described therein.

1. INTRODUCTION

There have been many attempts in the past five years to build applications for distributed and peer-to-peer systems that exploit social networks properties. For example, social networks are used for building censorship-resistant Internet storage, content sharing and publishing, routing protocols, and Sybil defenses. In each of these applications, social networks are assumed to be well-connected and trusted. For instance, in social networks-based Sybil defenses a high quality of trust reduces the attackers' ability to produce multiple identities, thus, defending against the Sybil attack—caused by nodes with multiple identities in distributed systems. For these Sybil defenses to work, social networks are assumed to be trust possessing and fast-mixing, a formal quality of high-connectivity of these networks. Other applications require other properties, such as betweenness, expansion, well-balance, etc. Despite their importance to these applications, there has been not much efforts spent understanding the quality of these properties in social graphs and how such quality affects the performance of these designs.

To make matters worse, some common practices in the field of trustworthy computing on social networks lack rigor by claiming authentic properties in social graphs after graph manipulation, thus calling for further investigations. For example, to bring insight on their designs of trustworthy computing—Sybil defenses in particular, many researchers alter social graphs by trimming lower degree nodes, convert naturally directed graphs to undirected ones and thus undermine directionality of edges in the underlying social graph, or sample larger graphs to smaller ones that are easier to work with, among many other practices. In all of these cases, researchers pay no attention to the altered algorithmic property in the underlying social graph and how this affects the operation of the trustworthy computing systems.

To this end, and to enable trustworthy computing on social networks, this work is focused on measuring, analyzing, and improving properties used for building applications using social networks. We proceed to this goal by large-scale *measurements* and *analyses* to understand these properties in their natural contexts, and in settings where they are altered as widely used in the literature. Mindful of lessons learned from the measurement, we then proceed to improving properties so as to improve the performance of systems built on top of social networks. Finally, we investigate the use of existing properties in social graphs, and to look at discovering other properties that can be easily achieved in variety of social networks to build trustworthy systems.

To motivate for our work and put it in the correct context, in this section we elaborate on the context and problems in distributed systems that other researchers in the community tried to solve using social networks, how and why social networks are used, and our take on these solutions. Then we proceed to describe our current

Figure 1: Rating and voting services are widely used in online marketplaces and content providers to rate their products by users. (The snapshot here is for a product on Amazon.com).

findings and results, as well as our proposed work.

1.1 Context and Problems

Many of the social network-based computing designs proposed in the literature address limitations in cyberspace system due to the lack of trust in them. These designs rely on the trust in social networks as the main driving factor and make use of some social networks properties to enable efficient and reliable operation of cyberspace systems. In the following we review some of these systems, where social networks are heavily used in the literature to improve certain aspects of systems opration.

1.1.1 Misuse in Content Voting Systems

Many systems and services utilize features to improve the experience of their users. For example, marketplaces like Amazon and eBay, and online content providers like Youtube and Flickr, make use of "rating" (or voting) on goods, contents, and individuals (sellers) as means of determining their qualities—an example is shown in Figure 1. These ratings are ideally provided by experienced users to tell how they perceive such items, and they usually influence other new users decisions. For example, an item that is highly rated and sold by a highly rated seller on Amazon or eBay, even when it is more expensive than the same item from other sellers, is highly likely to be considered by users based on such ratings. Also, highly rated videos on Youtube are more likely to be watched based on these ratings. Likewise, items with low ratings are unlikely to be considered by new users.

In the above scenarios, users put a good faith in these ratings. They presume that such ratings come from other users who used these systems—whether it is by watching a video on Youtube or by purchasing an item on Amazon—and then casted votes to characterize their experience. These rating systems are however prone to misuse. For example, a single user may purposefully rate an item several times, thus the rating of the item may become an inaccurate representation of its quality as perceived by other users. Many of these systems require users to log-in using authentication credentials, such as an email, in order prevent them from casting multiple votes. However, most of these systems do not bind such credentials (or digital identity) with the real identity of users, thus an attacker with the intention to misuse the system will be able to create multiple email addresses and use them to cast as many votes as she or he wants.

While centralized systems, like Amazon and eBay, my prevent the problem by requesting only users who bought an item to rate it, the problem as described above is not limited to these systems. Rating (or voting) is ubiquitously used in many centralized systems—where such misuse prevention techniques may not apply; examples include aforementioned Youtube and Flickr services—and decentralized systems. For example, ratings are used to determine the quality of videos shared on distributed file-sharing systems, where creating multiple identity to cast multiple votes or claim multiple roles in the system has proven to be easy.

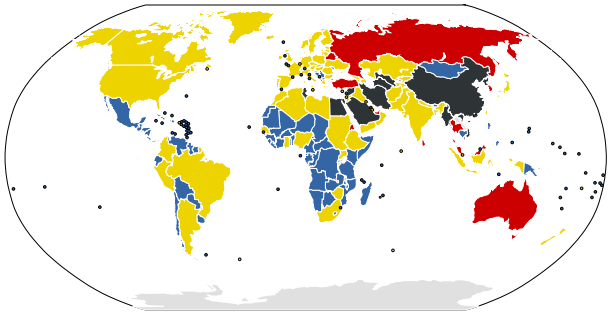


Figure 2: Censorship map of the world, where colors indicate the level of censorship. ■: No censorship, ■: Some censorship, ■: Country under surveillance, ■: Most heavily censored nations (source: Reporters Without Borders `http://en.rsf.org/`)

The problem as described above, when a single user creates multiple identities and try to use them as she or he is multiple users, is called in the literature the "Sybil attack". Defending against the Sybil attack remains challenging in many distributed computing services.

1.1.2 Identity in Distributed Computing Systems

Many distributed systems rely in their functioning on the true participation and collaboration of users. Examples where collaboration include the search functionality in distributed hash tables (DHTs), routing in delay tolerant networks, content dissemination in data delivery infrastructures, among many others. The distributed nature of these systems make it hard to use a globally enforced identity, and even if such identities were to be used, the lack of centralized authority to bind these identities to real identities make it easy for a single user to create multiple identities and perform a Sybil attack, thus abuse the system.

For example, distributed hash tables (DHTs), which are widely used in distributed data management and contents distribution (e.g., files sharing), require each user to store a subset of the identities of contents or users in the system and answer queries concerning these contents or users if needed by other users in the system. A typical misuse scenario would include the creation of multiple identities by a single node and making the search queries fail.

1.1.3 Censorship-resistant Communication

One of the main threats to freedom of the online speech is censorship. Governments in most countries in the world (as shown in Figure 2) censor the contents on the Internet and determine beforehand what contents people can access. To circumvent censorship filters, there has been several systems but the most notable is Tor (the onion router). Tor uses predetermined set of volunteer servers (called relays) to mix traffic of clients so that it is hard for a traffic analyzer to trace client's activities. Two of the main limiting factors of Tor are scalability, which is addressed in several recent works as in [85] and [92], and relays blocking, which is addressed by hidden relays; also called bridges [39].

Another solution to the problem has been using social networks as a mixing medium to communications among people represented in these networks (details are below).

1.1.4 Distributed Computing Services

Many distributed computing systems rely on resource donated by volunteers, people who share similar interests for similar causes [134,

25, 77, 31, 8]. Compute task outsourcers put good faith in the system; they trust other volunteers in the system to perform the compute tasks as intended. Applications of such systems are particularly popular in the research community for scientific computations. One of the main issues that face this compute paradigm is the recruitment of compute workers; in order for volunteers to participate in the system, they should to convinced by the causes for which these systems operate. This is likely to happen in academic institutes settings where institutes would be likely willing to donate their compute idle cycles for other institutes projects, given that other institutes wold do the same.

Another issue arises as the compute paradigm is made open to the public. In many of these applications, the outcome of the individual outsourced computations is usually combined to produce a result representing the different pieces of outcomes. To that end, to marginalize how misbehavior affects the operation of such systems, classical techniques like "replication" of compute tasks on different machines are used. In these systems, the same task is outsourced to different compute workers, and a voting mechanism is used for admitting results based on the outcome of the majority of the compute workers. Although that is likely to mitigate the impact of misbehavior, that could potentially be caused by users not performing computations, or generating multiple identities and taking multiple tasks with the intention of abusing the system, in many cases it is hard to reason about the trustworthiness of such systems and computations delivered by them.

1.2 How Do Social Networks Help?

In most of the systems and applications we described above, the origin of the problem is the lack of trust in the system. All individuals are likely to be treated equally. In a voting system, like the one used in Amazon and eBay, votes coming from supposedly different users are treated equally and presented in the system in an aggregate form to reflect qualities of products. In a distributed computing system, any node that announces availability of resources is likely to be used as a worker for any outsourced computations.

All methods proposed and used for defending against misuse and misbehavior in distributed systems try to fix this aspect in the system by bringing trust into these systems. In centralized solutions to centralized or distributed computing systems, centralized trusted authorities are used to confirm digital identities of users, match them to real identity credentials, or maintain a record of users behavior and history that can be used further to decide whether to accept or deny participation of users in the future.

Another solution to the problem looks at bringing trust from other orthogonal contexts to the distributed system where users in the distributed system are represented in that context. Accordingly, decisions whether to accept participating of users in the distributed system is tied to the trust value of these users in that orthogonal system. One good example of such contexts is social networks. Many social networks enjoy both trust characteristics and algorithmic properties, such as well-connectivity of nodes in the network to each other, that support the operation of algorithms to limit misbehavior in distributed systems. Furthermore, ties among social acquaintances in the social network can be used in a decentralized manner, without a centralized authority, making them a good fit for bringing trust into distributed and decentralized computing environment where trust is a missing factor.

Indeed, this vision has been used intensively in the past five years, where several designs are proposed in the literature to improve the operation of distributed systems using trust and algorithmic properties of social networks. Solutions using social networks included designs to defend against the Sybil attack in applications like rating and voting algorithms that are used in variety of systems, designs to improve routing in delay tolerant networks, designs to improve routing in socially selfish networks, designs to improve trustworthiness of online storage and back-up systems, designs to provide anonymity for online communications, and others to preserve user's privacy in file sharing systems, among many other systems and designs.

Most of these designs have common design principles. A typical design among those proposed in the literature would rely on the trust in the social network, and would support its operation for efficiency or security using theoretical arguments. These theoretical arguments are based on the algorithmic properties claimed in the social networks—properties that wide variety of social networks would enjoy so that the operation of these designs is possible on any of them. For example, many Sybil defenses built on top of social networks to be trust-possessing and "fast-mixing". Other Sybil defenses require social graphs to have "good" expansion, be well-balanced, or to have high betweenness of nodes so that such characteristics can be used to determine the "goodness" of nodes in the social graph and thus decide to admit or deny participations originated by these nodes in the orthogonal distributed system.

Some of these qualitative properties are assumed with certain quantities. For example, social network-based Sybil defenses using the "fast-mixing" property of social graphs assume that a short random walk with a certain length will result into certain property of the sampled nodes from the graph after a walk of that length. This is, walking on the graph for a number of steps that is in the logarithmic order of a graph size would result into sampling nodes driven from a probability distribution that is "almost identical" to the node degree distribution of the graph. Other systems assume other properties, and require them to be in certain quantities so that they work correctly in theory.

Despite the importance of these assumption for both theoretical and practical guarantees of these designs, none of the works in the literature tried to measure these properties directly in social graphs. Typically, all works in the literature inferred these properties indirectly by measuring the performance of these designs (Sybil detection, routing, information dissemination, etc). Given the experimental results of these designs on real-world social networks that meet the theoretical arguments and formulations, authors of these works have assumed that the algorithmic properties used for reasoning about the operation of these designs exist in social networks with the assumed qualities.

1.3 Trustworthy Computing on Social Networks

Our work in this thesis is motivated by the above arguments. We looking at bridging the gap between theory and reality in trustworthy computing at social networks. Motivated by the lack of work on measuring social networks properties used for trustworthy computing, we initiate this direction by measuring several social network properties and relate these measurements to both theoretical and practical guarantees of these designs.

Using outcomes of these measurements, we also ask and answer several natural questions: a) Do social networks have the quality of properties assumed and widely used in the literature for building trustworthy computing systems? b) If not, do these designs operate on top of social networks with the claimed guarantees even when such properties do not hold? c) Can we characterize social network properties and the reason behind their qualities? d) If so, can we improve these properties? e) Do common practices used in the literature of graph sampling and omitting graph directions affect the qualities of these properties and the operation of these systems on top of social networks? f) Can we build new systems that either

improve on prior work using reasonable assumptions or utilize new properties in social networks unused previously?

Although the main property we consider in our work is the mixing time of social graphs, which is widely used as the corner stone for building a wide variety of social network-based Sybil defenses, we also measure other properties: the expansion of graphs and betweenness of nodes that are also used for building Sybil defenses. Betweenness also has been used for improving routing in delay tolerant networks, among others. The mixing time of the graph, which is a measure of connectivity of the graph, is also used indirectly in other systems to support the efficiency of information dissemination, routing, and anonymous communication. We touch upon how these applications are affected by the qualities of these graphs.

To this end, we make two broad multifold contributions in this thesis. First, we test the underlying properties in social networks used for building trustworthy computing systems. Second, we propose several systems and protocols that either improve these properties or use new properties unseen in prior works.

1.4 Summary of Contributions

In this proposal, we make three multifold contributions. In the following, we elaborate on each of these contributions.

1.4.1 Measuring Social Networks Properties

Mindful of the importance of certain social networks properties for the operation of social network-based applications, such as Sybil defenses, information dissemination algorithms, and anonymous communication systems, we proceed to measure these properties in real-world social networks and graphs. Our main motivation of doing this study is as follows. First, to the best of our knowledge, some of these properties—such as the mixing time of social graphs—is not studied before on large scale social graphs. Second, it is not clear what quality of these properties is required for the operation of these applications. Whereas theoretical guarantees of Sybil defenses on social networks, for example, require certain quality of the mixing time, it is not clear if such quality existed in social networks. Last, and as mentioned earlier, several techniques are used to alter social graphs, and it is not clear how such techniques affect the property used for building these applications. Furthermore, it is not clear how this alteration affects the operation of these applications. To this end, in this work we make [1] the following contributions (and highlight the main interesting findings in each contribution):

1. We investigate tools and use them to measure the mixing time of undirected social graphs [106]. For that, we use two methods: the second largest eigenvalue modulus (SLEM) and the mathematical definition of the mixing time. We use the definition to express the richer pattern of the mixing in the social graphs. Unlike what has been claimed in the recent literature of building social network-based Sybil defenses using the "fast mixing" property, we find that many social graphs are slower mixing than anticipated and needed by such defenses. By experimenting with some of the literature defenses, we discover that the quality of the mixing time required for operating these defenses is weaker than claimed in the literature. As such, we made two firm conclusions. First, some of the theoretical guarantees claimed in these defenses based on the claimed property of the mixing time are inaccurate. Second, some of the practical security guarantees of these defenses when operated on some social graphs are weaker than claimed. Additional findings in this work show that the mixing time is greatly improved in social graphs when lower degree nodes are trimmed from the graph, a common practice in the literature.

2. While many social graphs are directed by nature, many applications are often evaluated on undirected versions of them by omitting edge directions. In this direction, we develop tools to measure the mixing time of directed graphs and develop its error bound. We then measure the mixing time of several directed benchmarking graphs and their undirected counterparts. Our initial measurements show that directed graphs are slower mixing than undirected ones [98]. We use two state-of-the-art applications to demonstrate the impact of these findings on designs on top of social networks: SybilLimit and "anonymity in the wild". We found that evaluation of applications on the undirected graphs always overestimates the security provided by these applications.

3. To understand why some graphs are fast-mixing and why some others are not, we related the mixing time to degeneracy, which captures cohesiveness of the graph [103]. We show that fast-mixing graphs have a larger single core, whereas slow mixing graphs have smaller multiple cores. We build on these observations by designing techniques to improve the mixing time of slow mixing graphs using auxiliary links [103]. We also show that another property recently used for building a Sybil defense—namely graph expansion, relates to the mixing time. While fast mixing graphs have good expansion, slow mixing graphs have poor expansion.

4. Graph sampling is widely used to address infeasibility of measurements in large graphs, or for crawling graphs when accessing an entire social graph at once is infeasible. Many sampling algorithms are used, and they aimed at maintaining certain properties of original graphs in the sampled ones. We studied sampling algorithms' bias on the mixing time [99]. We show that some sampling algorithms, including those unbiased to degree distribution, always produce biased estimation of the mixing time. We further conclude that bias in sampling algorithms is rather metric-dependent, and while an algorithm may work nicely to one property, it may produce considerable bias in the mixing time.

1.4.2 Better Assumptions by Incorporating Trust

Social trust is the other main feature used—along with the properties examined above—for building trustworthy computing applications on social networks. While most applications in the literature required certain qualities of social trust so as to reason about their operation and guarantees, the very same applications are experimented on online social graphs which are known for their weak trust characteristics.

For example, social network-based Sybil defenses do not consider the different amounts of trust represented by different graphs nor the different levels of trust between different nodes, though trust is a crucial requirement in these defenses. To address this problem, we introduced two theoretical (lazy- and originator-based) and two data-driven (similarity- and interaction-based) designs to tune the performance of Sybil defenses by accounting for differential trust [97].

Each of these designs biases the random walks used for operating these Sybil defenses. Interestingly, we find that the cost of operating Sybil defenses is greater in graphs with high trust than in graphs

[1] In this proposal we use the present and past tenses though some of these proposed works are ongoing works to be submitted (working titles; initial results are reported).

with low trust values. We discovered that this behavior is due to the community structure in high-trust graphs, requiring higher costs to traverse multiple communities. Furthermore, we showed that our proposed designs to account for trust—while they increase the cost of operating Sybil defenses on graphs with low trust value—greatly decrease the advantage of attacker.

1.4.3 Building New Applications On Social Networks

In parallel with other contributions, we explored the vein of social networks-based systems design. In this direction, we make three independent contributions. We build SocialCloud, a new volunteer-based time-sharing computing paradigm, DynaMix, a new anonymity enabling communication system on social structure by exploiting edge dynamics, and MeetUp, a secure encounter-based social network.

1. SocialCloud. We explore a new computing paradigm, called SocialCloud [100], in which computing nodes are governed by social ties driven from a trust-possessing social graph. We show that incentives to adopt this paradigm are intuitive and natural, and security guarantees provided by it are solid. We propose metrics for measuring the utility of this computing paradigm, and consider several design trade-offs for its operation. Using real-world social traces, we run an event-driven simulator of SocialCloud, and demonstrate the potential of this paradigm for ordinary users. Interestingly, we find graphs known to perform poorly for Sybil defenses [106] are good candidates for our SocialCloud for their "self load-balancing" features.

2. DynaMix. Existing solutions for anonymous communication on social structures undermine the impact of networks dynamics on anonymity guarantees. We propose DynaMix, an anonymous communication system that exploits dynamic structures in social networks [105]. We formally show an intuitive connection between anonymity on dynamic graphs and random walks on weighted graphs in which weights summarize the history of edges and allow for future dynamics to weight adjustment. We showed several measurements of our proposed model on dynamic graphs extracted from real-world social networks and compared it to static structures driven from the same graphs, highlighting potential of our proposed system enriching graph structure and improving quantitative anonymity as both entropy and anonymity sets.

1.5 Roadmap

The rest of this proposal is organized is three main parts (sections). In section 2 we introduce the first major contribution (summarized in section 1.4.1) in more details and articulate the ongoing and proposed work on measuring, verifying, and understand several social network properties used for building applications on top of social networks. In section 3, we introduce the second major contribution on accounting for trust in social network-based applications (summarized in section 1.4.2). In section 4 we review two applications we propose to build on top of social networks (summarized in section 1.4.3). Each section has a summary of the work that serves as concluding remarks.

2. MEASURING SOCIAL NETWORKS

The Sybil attack is a well-known and powerful attack in distributed systems. In the basic form of this attack, a peer representing the attacker generates as many identities as she can and acts as if she is multiple peers in the system. These "virtual" peers are then utilized to influence the behavior of the system [43]. The number of identities that an attacker can generate depends on the attacker's resources such as bandwidth, memory, and computational power. With the sharp hardware growth—in terms of storage and processing capacities—and the popularity of broadband Internet, even attackers who use "commodity" hardware can cause a substantial harm to large systems. Classical solutions to the problem are insufficient in many distributed systems contexts, for that they assume the existence of centralized authorities in such systems [17, 43, 21, 5, 64, 41, 81, 137, 122, 139]. On the other hand, social networks-based solutions to the problem [148, 147, 146, 145, 131, 132, 129, 130, 69, 71, 35] avoid any assumption on centralized authorities, which are replaced by (social) peers participation in the distributed system under certain assumptions

In these decentralized defenses, peers in the network are not merely computational entities—the human users behind them are tied to each other to construct a social network. The social network is then used for creating designs (such as those in bootstrapping the security and detecting Sybils under two assumptions: algorithmic and sociological. The algorithmic assumption is the existence of a "sparse cut between the Sybil and non-Sybil subgraphs" in the social network, which implies a limited number of attacker edges; edges between Sybil and non-Sybil nodes. Furthermore more, these defenses assume that honest region of the social network is "***fast-mixing***"; a quantified quality of the connectivity of the honest graph (the graph that contains the honest nodes only). The sociological assumption is a constraint on the trust in the underlying social graph: the social graph used in these defenses needs to exhibit "***strong trust***" as evidenced, for example, by face-to-face interaction demonstrating social actors' knowledge of each other [148, 147, 146, 145, 131, 132, 129, 130, 69, 71, 35]. Despite their essential role to the practicality of any potential design on top of social network, these designs are blindly accepted without rigorous verification of their validity. Other examples of scenarios of using social networks for building applications without verifying properties being used include Sybil defenses in mobile networks [115], the use of social networks for routing and social search [82, 26, 36, 14, 32, 83], which all assume the features of "***well-balance***", "***good expansion***", "***good betweenness***", or "***clustering of graphs***", among others.

Until now (late of 2009; the time of starting this work) these assumptions are not challenged. Although several social network properties like clustering, betweenness, and connectivity properties, are widely studied and measured [22, 87, 118, 63, 6], none of these works is directed towards these applications built on top of social networks using these properties. To the best of our knowledge, and before our work, no work tried to measure the mixing time of social graphs and relate the quality of the mixing time in these graphs to guarantees of Sybil defenses. The only work in the literature addressing this issue is in [38], which does not address these defenses.

Concurrent to work, the algorithmic assumption has been indirectly questioned in [136], where it is shown that some Sybil defenses are sensitive to the community structure *in a selected set of social graphs*. However, the authors did not measure the mixing time nor tried to relate its quality to these systems guarantees. On the other hand, although there has been several works in the recent literature on social network infiltration [15, 55, 18]—where authors of these works have claimed that their findings would affect the operation of Sybil defenses on top of social networks, none of these works tried to quantify how these attacks would affect the operation of Sybil defenses. More importantly, none of these works tried to provide fixes for these attacks to improve the operation of Sybil defenses.

In the rest of this section, we investigate measuring these properties and relate that to the guarantees of these systems. In particular, we investigate tools and use them to measure the mixing time of different social graphs and relate that to guarantees of the Sybil defenses. We propose to measure how the mixing time is affected by widely used practices, like omission of edge directions, and sampling, and how that affects two classes of applications built using this property, the Sybil defenses and anonymous communication systems. Then, we measure two other properties, namely the expansion of the graph and the betweenness of nodes. While the former is used for building a Sybil defense, the latter is used for building a Sybil defense and a routing protocol. We relate these measurements to the mixing time and applications built on top of social networks. Finally, we explore a heuristic to improve the mixing time by auxiliary links, given our understanding for the reasons beyond its quality in different graphs.

2.1 How Graph Properties Are Used

To show how graph properties are used for building these systems, in this section describe two of them, a Sybil defense, called SybilLimit, and an anonymous communication system, called "anonymity in the wild". Both systems rely on the mixing time for their operation, and certain qualities of the mixing time are assumed for their theoretical and practical guarantees.

2.1.1 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 $w = O(\log n)$ and picking the last edge in the walk in the sample. Under certain assumptions on the graph's mixing characteristics, there is an overwhelming probability that two sampled subsets of honest nodes/edges 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 $O(\log n)$ so that the distance between the stationary distribution of the graph and the walk graph after $O(\log n)$ is $\Theta(1/n)$ [2]—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 $1-\frac{1}{n}$ (Theorem 1 in [146]). 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 a non-empty set 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, probability for random walks originated from honest region to dishonest region are limited. The impact of such "escaping tails" on the operation of the defense is further marginalized using a "balance condition" which ensures that accepting a suspect would not cause a spike of the number of accepted suspects via a certain edge in the graph. Chances of dishonest nodes being accepted by sampling honest edges is limited, and bounded by the number of attack edges.

2.1.2 Anonymous Communication Systems

The idea of mixers over social links is very simple [109]. In these systems [109, 34], users recruit their social acquaintance to relay their traffic and to provide anonymity to them. In the nutshell, each node (user) forwards her own traffic to her friends, and friends forward that traffic to their friends, and so on, for a certain number of hops, e.g. w. The number of hops w is a system-wide parameter, which is determined by the security level desired in the system. The anonymity is defined for two parties; the sender and the receiver of traffic (we follow the same model in [109] for defining the anonymity of both parties).

For a sender, the anonymity defined in terms of the *anonymity set* is n, thus the entropy of the probability distribution of any node being the sender is $\log_2(n)$—same for both directed and undirected graphs. On the other hand, the anonymity set for a node being the receiver is determined by the probability distribution achieved after the fixed number of hops w used in the system. Let the distribution of the final node selected in a *random walk* after w hops be $\pi_i^w = \pi_i \mathbf{P}^w$, where $\pi_i^w = [\pi_i^w(j)]^{1\times n}$ (π_i is an initial distribution). The anonymity of the receiver of the traffic (the last hop in the walk) is measured by the entropy H_w, which is given as

$$H_w = -\sum_{j=1}^{n} \pi_i^w(j) \log_2 \pi_i^w(j) \quad (1)$$

Using the entropy in Eq. (1), we define the *anonymity set* $A_w = 2^{H_w}$. The maximum entropy and anonymity set for a walk on a graph are achieved with the probability distribution of that walk as it approaches the stationary distribution.

We use $\overline{H_w^d}$ and $\overline{A_w^d}$ for the average entropy and anonymity sets in a directed graph, while $\overline{H_w^u}$ and $\overline{A_w^u}$ are used for the average entropy and anonymity sets in an undirected graph. We define the average entropy and anonymity sets for 1000 random walks starting from different sources (see below).

Unlike SybilLimit, where certain parameters are required for its operation and is used for its security proof, "anonymity in the wild" does not assume any qualities of the mixing time. It rather shows that a short random walk on the graph is sufficient to reach close enough to the stationary distribution so that the anonymity set of the walk is a large proportion of the maximum anonymity set (i.e., the power of the entropy in the stationary distribution).

2.2 Tools to Measure the Mixing Time

Theoretical results that provide tools to measure graph properties have been already studied intensively over the past decades in other contexts, including graph and probability theories. Such results can be applied directly, or with small modifications to the problem in hand, and to understand the extent to which assumptions being made about social networks exist in reality. In the following, we review some of these results and using them to understand the algorithmic properties of social graphs in real-world social networks, an assumption that is being used for reasoning about the behavior of social network based systems.

2.2.1 Social Networks: Directed vs. Undirected

Both directed and undirected social networks exist. Popular online directed social networks include Twitter, Youtube, and Google+, whereas undirected social networks include Facebook, Myspace, and LinkedIn, to mention some. While most of the literature work on using social networks to build trustworthy computing systems assumes undirected social graphs, some others (like [19, 71, 146, 136, 91]) have used directed graphs and altered them by either omitting edge directions entirely or by considering a connected subgraph in which an edge is established between two nodes if it is symmetric (i.e., an edge that exists in both directions). Accordingly, in this section, we consider both types of graphs, directed and undirected.

Formally, we refer to an undirected graph as $G = (V, E)$, where V ($|V| = n$) is the set of nodes in the G and E ($|E| = m$) is the set of edges (relationships or interdependencies) between the

[2] This quantity is further assumed to be $1/n$ in many lemmas in SybilLimit's proof; see for example Lemma 7 and Theorem 3 in [146].

nodes. For G, we define the symmetric adjacency matrix $\mathbf{A} = [a_{ij}]^{n\times n}$. where the $a_{ij} = 1$ if an edge exists between v_i and v_j in V. We define the degree of a node $v_i \in V$ as the number of nodes in V adjacent to v_i and denote it by $\deg(v_i)$. For G, we define $\mathbf{P} = [p_ij]^{n\times n}$ as the transition matrix, where $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. $\mathbf{D}$ is defined as a diagonal matrix where the $ii-$th element in $\mathbf{D}$ is $\deg(v_i)$.

For clarity, we refer to a directed graph on n vertices and m edges as $\mathbb{G}$. Similar to the undirected graph case, let $\mathbf{A} = [a_{ij}]^{n\times n}$ be the adjacency matrix of $\mathbb{G}$, where $a_{ij} = 1$ if there is an edge from v_i to v_j in $\mathbb{G}$ (denoted by $v_i \to v_j$), and 0 otherwise. Let $\deg(v_i)^-$ be the out-degree of node v_i.

We define the transition probability matrix $\mathbf{P} = [p_{ij}]$ where $p_{ij} = 1/\deg(v_i)^-$ iff $v_i \to v_j$, and 0 otherwise. In a clean matrix form, $\mathbf{P} = (\mathbf{D}^-)^{-1}\mathbf{A}$, where $\mathbf{D}^-$ is a diagonal matrix in which the ii-th element is defined as $\sum_j a_{ij}$.

The stationary distribution, π, of random walks on $\mathbb{G}$ (G, respectively), is defined as a probability distribution that is invariant to $\mathbf{P}$ (i.e., $\pi = \pi\mathbf{P}$). Formally, π is defined in Theorem 1 for undirected graphs (with certain properties; see below—the same definition also applies to the case of strongly connected directed graphs as well).

THEOREM 1. *Let $\mathbf{P}$ be the probability transition matrix of a Markov chain that is periodic and strongly connected, defined on a graph $\mathbb{G}$. Then,*

$$\lim_{t\to\infty} \mathbf{P}^t = \mathbf{P}^\infty \tag{2}$$

where $\mathbf{P}^\infty$ is an $(n \times n)$ matrix of identical rows, where each row equals to π, the stationary distribution of every walk on $\mathbb{G}$.

For an undirected graph, $\pi = [\deg(v_i)/2m]^{1\times n}$, whereas π in directed graphs in general has no closed-form expression.

2.2.2 The Mixing Time: The Definition

Significant fraction of social network-based designs use the mixing time as one of their assumption for operation and security guarantees. In these designs social graphs are assumed to be "fast-mixing" for all Sybil defenses. For G defined in §2.2.1, recall that $\mathbf{P} = [p_{ij}]^{n\times n}$, where

$$p_{ij} = \begin{cases} \frac{1}{\deg(v_i)} & \text{if } v_i \text{ is adjacent to } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The "event" of moving from a node to another in the graph is captured by the Markov chain which represents a random walk over the graph G. A random walk R of length k over G is a sequence of vertices in G beginning from an initial node v_i and ending at v_t, the terminal node, following the transition probability defined in Eq. (3). The Markov chain is said to be *ergodic* if it is irreducible and aperiodic. In that case, it has a unique stationary distribution π and the distribution after random walk of length k converges to π as $k \to \infty$ (Theorem 1). The *mixing time* of the Markov chain, T is defined as the minimal length of the random walk in order to reach the stationary distribution. More precisely, T (parameterized by ϵ) of a Markov chain is defined as

$$T(\epsilon) = \max_i \min\{t : |\pi - \pi^{(i)}\mathbf{P}^t|_1 < \epsilon\}, \tag{4}$$

where π is the stationary distribution, $\pi^{(i)}$ is the initial distribution concentrated at vertex v_i, $\mathbf{P}^t$ is the transition matrix after t steps, and $|\cdot|_1$ is the total variation distance defined as $\frac{1}{2}\sum_j |\pi(j) - \pi_i^t(j)|$ (where $\pi_i^t = \pi_i\mathbf{P}^t$).

The Second Largest Eigenvalue Modulus (SLEM). In addition to the definition in Eq. (4), which can be used to compute the mixing time of an undirected graph directly from its transition matrix, the mixing time is bounded by the second largest eigenvalue of the transition matrix $\mathbf{P}$. This is, let $\mathbf{P}$ be the transition matrix of G with ergodic random walk, and λ_i for $1 \le i \le n$ be the eigenvalues of $\mathbf{P}$. Then all of λ_i are real numbers. If we label them in decreasing order, $1 = \lambda_1 > \lambda_2 \ge \cdots \ge \lambda_{n-1} \ge \lambda_n > -1$ holds. We define the second largest eigenvalue μ as $\mu = \max(|\lambda_2|, |\lambda_{n-1}|)$. Then, the mixing time $T(\epsilon)$ is bounded by

$$\frac{\mu}{2(1-\mu)}\log(\frac{1}{2\epsilon}) \le T(\epsilon) \le \frac{\log(n) + \log(\frac{1}{\epsilon})}{1-\mu}. \tag{5}$$

"Fast-Mixing" Social Graphs. We say that the Markov chain is rapidly mixing if $T(\epsilon) = \text{poly}(\log n, \log\frac{1}{\epsilon})$. In literature [30, 123], the rapid mixing of the Markov chain is cited as "fast mixing" for the graph [35, 71, 146, 147]. Here, we follow the tradition of referring to this bound as "fast mixing". Also, again following these previous work, we strengthen the definition by considering only the case $\epsilon = \Theta(\frac{1}{n})$, and requiring $T(\epsilon) = O(\log n)$ [35, 71, 146, 147]. For the mixing time definition in Eq. (4), and when considering an undirected unweighted graph, the stationary distribution is $\pi = [\pi_{v_i}]^{1\times n} = [\frac{\deg(v_i)}{2m}]^{1\times n}$ for $i = 1 \ldots n$. As mentioned before, computing the exact stationary distribution for a directed graph in a closed-form expression is not possible, although computing an estimate and its error might be possible.

The Mixing Time and Other Properties. The mixing time is tightly related to the connectivity of the graph, which is explicitly assumed in many contexts of previous works [35, 69, 97, 129, 131, 145, 146, 147, 148, 91]. This is, strongly-connected graphs are fast mixing and have small mixing time while weakly connected graphs are slow mixing and have large mixing time [123]. Some of the works cited (e.g., [131]) above informally refer to requiring a well-connected graph by having a expansion factor [108], another terms that is related to the mixing time [60]. Finally, the second largest eigenvalue used for measuring the mixing time (in Eq. (5)) bounds the graph conductance, a measure for the community structure [136] in these graphs. In short, the conductance is $\Phi \ge 1 - \mu$ [58].

Average "Mixing Time". Although the mixing time as defined in Eq. (4) is defined overall random walks as the shortest walk from the worst mixing source in the graph, there has been recent work in the literature to define an average mixing time and the average conductance [52], rather than the worst case. This measure is further bounded by other graph properties; see definition 7.13 and Lemma 7.14 in [53]. In a fellow-up to our work, and agreeing with our conclusions in [106], the author of [144] suggested that the average mixing time might be in use for the Sybil defenses, rather than the definition in Eq. (4).

2.3 Other Algorithmic Properties

Other basic properties used for building applications on top of social networks include the *betweenness centrality* of nodes in the social graph. The *betweenness centrality* [47] captures the significance of nodes to the flow between other nodes. Two types of betweenness centrality are known in literature: *shortest-path betweenness* [47, 140] and *random walk-based betweenness* [110].

The ***shortest path betweenness*** weighs the significance of nodes as a result of their occurrence at the shortest path between other nodes by normalizing the number of paths between every two nodes in the network that pass through a particular node by the total number of such paths. While the shortest path betweenness requires a global knowledge of the whole topology—an issue that raises a lot

of concerns in social network-based applications, the random walk based betweenness can be implemented at fully decentralized settings. Recall the random walk defined earlier and recall that the random walk connecting nodes v_i and v_j, denoted as $v_i \overset{p_r}{\rightsquigarrow} v_j$, is the set of nodes where each node on the random walk is selected uniformly at random by its prior node.

We define the ***random walk-based betweenness*** for a node v_i as the normalized expected number of times it is hit by the random walk. In short, the same model of the shortest path-based betweenness is used to compute the random walk-based betweenness by replacing the short path in the first one by the random walk in the latter one.The betweenness is used in [115] for defending against the Sybil attack and in [32, 33] to improve routing in DTNs.

The ***similarity*** is another property that is also used for building or improving the quality of applications that leverage social networks [32, 33, 113, 50, 65, 51]. The similarity between two nodes in social networks is used for measuring the strength of social links and potentially predicting future interactions [29, 74], or even weighting the value and significance of future interactions between nodes in social graphs [50, 65, 84]. Intuitive simple meaning of the similarity between two nodes in the context of a friend-to-friend social network is the number of friends common to both of them.A measure of similarity that reflects this intuitive meaning is the "cosine similarity" which computes how vectors are close to each other, each representing the row of a node in $\mathbf{A}$ [74].

Another measure of centrality in graphs is the ***closeness***. The shortest path closeness captures how close is a node to others in the graph based on the expected length of the shortest pathes between that node and other nodes in the graph. Applications of such measure include routing on top of social networks [11, 26].

The (vertex) ***expansion*** of the graph is used in proving theoretical security guarantees of a Sybil defense. Let $S \subset V$, where $0 < |S| \leq 1/2|V|$, be any set of vertices in the graph. The (vertex) expansion factor α is defined as

$$\alpha = \min_{0<|S|\leq \frac{n}{2}} \frac{|N(S)|}{|S|} \tag{6}$$

Where the minimum is over all nonempty sets S of at most $n/2$ vertices—S need not be connected and thus the number of possible configurations of S is exponential in n—and $N(S)$ is the set of vertices not in S but each of which is connected by an edge to another node in S. In [130], where this property is used for building a Sybil defense, the definition of S is further restricted so as S is connected. Such restriction reduces the number configurations of S to become linear in n. In the subsequent subsections, we elaborate on how to measure α in the latter context.

2.4 Measuring the Mixing Time

Now we proceed to measure the mixing time of different social graphs to see if the qualities assumed in the prior work in the literature exists in these graphs or not.

Although the mathematical tools for measuring the mixing time are known in literature, measuring the mixing time—especially of large graphs—is a computationally non-trivial task, requiring $O(tn^3)$ computations for a walk length parameter t and network size n, and that might be the reason why fewer efforts are made to measure this essential property in large social graphs.

2.4.1 Datasets

The social graphs used in our experiments are in Table 5. These graphs are selected to feature two models of knowledge between nodes in the social networks. These networks are categorized as follows. (1) social networks that exhibit knowledge between nodes and are good for the trust assumptions of the Sybil defenses; e.g., physics co-authorships and DBLP. These are slow mixing, as we will see later. (2) Graphs of networks that may not require face-to-face knowledge but require interaction; e.g., Youtube and Livejournal. Closely related to those is the set of graphs that may not require prior knowledge between nodes or where the social links between nodes are less meaningful to the context of the Sybil defenses; e.g., Facebook and wiki-vote, which are shown to be fast mixing.

2.4.2 Methodology

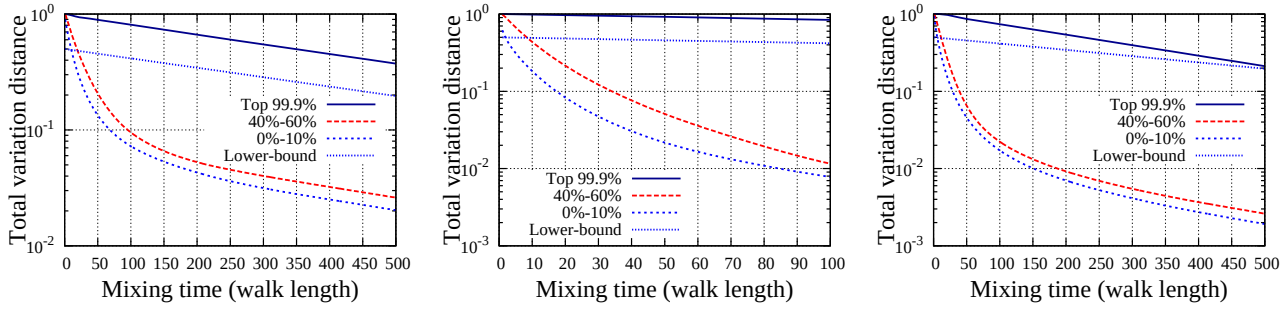
Equipped with the mathematical tools explained in section 2.2, we measure the mixing time of the different social graphs shown in Table 5. In order to apply the tools in section 2.2 for measuring the mixing time, and to be consistent with other works in the literature of using social networks for applications that exploit the mixing time [136, 35, 71, 69, 132, 146, 147, 148], we first convert directed graphs to undirected. Our conversion method connects two nodes if an edge exists between them in either directions, or both. We further compute the largest connected component in each graph and use it as a representative social structure for measuring the mixing time, as the mixing time is undefined for disconnected graphs. For small to medium-sized graphs, we compute SLEM directly from the transition matrix of the graph. On the other hand, for feasibility reasons, we sample the representative subgraphs from each of the four large data sets (Facebook A, B and Livejournal A, B) using the breadth first search (BFS) algorithm beginning from a random node in the graph as an initial point.[3] We perform this sampling process to obtain graphs of 10K, 100K and 1000K nodes out of 3 to 5 million nodes in each original social graph. Bearing the different social graphs sizes in mind, as shown in Table 5, we proceed to describe the results of our experiments.

2.4.3 Results

Figure 3 and Figure 4 plot the lower bound of the mixing time for the different graphs in Table 5. We choose to use the lower-bound, but not the upper bound, because it is more relevant to the context of our study. In particular, as we observe that the lower-bound of the mixing time to satisfy a given ϵ is large, it is obvious that the mixing time for social graphs is slower than anticipated. As shown in Figure 3, we also observe that the mixing time is very slow, in particular for social graphs that require physical acquaintance of the social actors, as can be seen in the general tendency of these graphs. For example, physics co-authorship, Enron, and Epinion, though the social network is small, a mixing time of 200 to 400 is required to achieve $\epsilon = 0.1$. Similarly for larger social graphs, as shown in Figure 4, the mixing time to achieve $\epsilon = 0.1$ is varying and depends on the nature of the data set. For example, while it is about 1500 to 2500 in case of Livejournal, it ranges from 100 to about 400 in case of DBLP, Youtube, and Facebook.

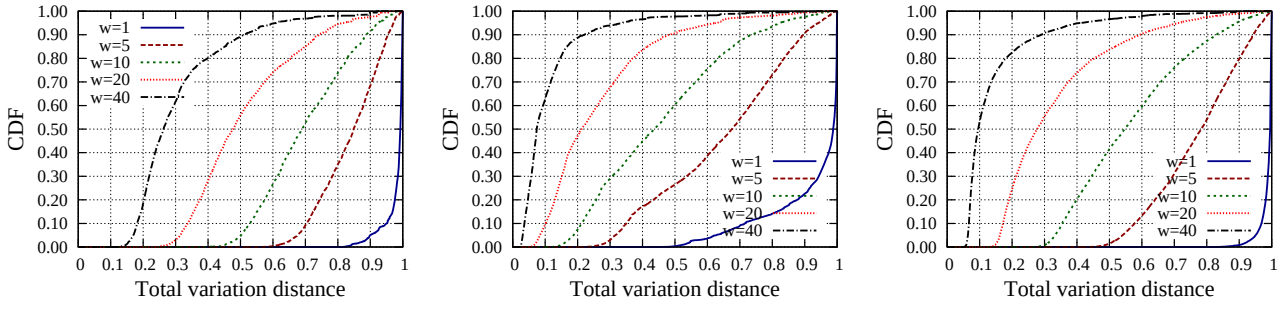
To see how tight are these measurements we perform the following experiment. We first compute the lower bound of the mixing time for the physics co-authorship data sets, which are also reasonably small and feasible to do exhaustive computations. Then we measure the mixing time using the model in (4) from every possible source in the graph; the CDFs of the raw measurements are shown in Figure 6 for different t values. We aggregate these measurements into Figure 5, by sorting ϵ at each t and averaging values in various intervals as percentiles. We observe that while the mixing time of most sources in social graphs is better than that of the mixing time given by SLEM, the measurements using SLEM are

[3]Note that BFS algorithm may bias the sampled graph to have faster mixing. Since our goal is to show that the mixing time is slower than expected, this only strengthens our position.



(a) Physics 1 (b) Physics 2 (c) Physics 3

Figure 5: Lower-bound of the mixing time compared to the mixing time when measured using the sampling method for the entire graphs brute-forcefully — different measurements meet the guarantees.



(a) Physics 1 (b) Physics 2 (c) Physics 3

Figure 6: The commutative distribution function (CDF) of mixing time for the three physics datasets in Table 5. The variation distance is computed for every possible node in the graph, brute-forcefully.

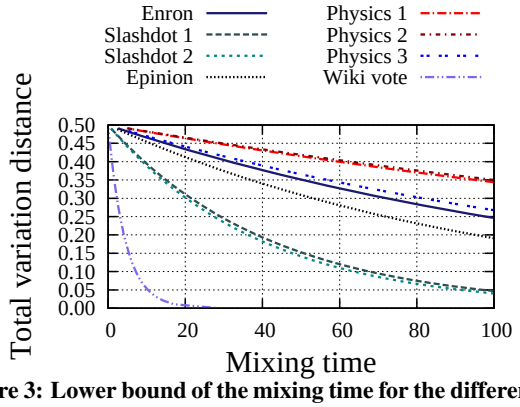


Figure 3: Lower bound of the mixing time for the different data sets used in our experiments — the case of small data sets.

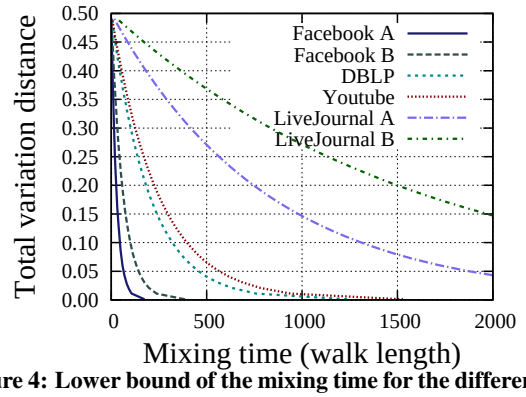
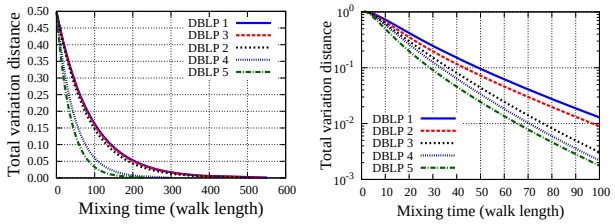


Figure 4: Lower bound of the mixing time for the different data sets used in our experiments — the case of large data sets

correct since the mixing time is by definition maximum of walk lengths for given ϵ as shown in (4). However, even considering this effect, still for most sources the mixing time is slower than used by other papers (10 and 15 in SybilLimit).

In [70], Lesniewski-Laas has interpreted our results—stated in [106] and shown in Figure 5—as that walks initiated by a few sources are slow mixing while the overwhelming majority of sources have fast mixing walks. However, we notice that the slower mixing sources in some of these graphs are still large portion of nodes. For example, based on the results shown in Figure 6(a), even though that a random walk of length 40 yields $\epsilon < 0.3$ for about 60% of the sources in the network, the remaining sources have $\epsilon \geq 0.3$. For about 10% among the total number of nodes in the graph (about 400 nodes), ϵ for the same length of a random walk of 40 steps is about 0.5, making the statement about the quality of "fast mixing" and "slow mixing" rather ambiguous. As we see latter, our observations in [106] are accurate when taken with further findings in the same context. This is, given that the quality required for operating these defenses might not be as strict as assumed, even those slower mixing sources can be considered "fast mixing" for these applications. Notice that these findings are not limited to the dataset in Figure 6(a) but also apply to those in Figure 6(b) and Figure 6(c). Furthermore, notice that these measurements are not in line with assumptions used in Sybil defenses, like SybilLimit, where ϵ is assumed $1/n$ for a walk length of 10 to 15.

To understand the relationship between the network size and the mixing time (of the same social graph) we use the different previously sampled subgraphs, using BFS, from Facebook and Livejournal data sets (10K, 100K, and 1000K). We further measure the mixing time using SLEM and the model in (4) for 1000 initial distributions. We further aggregate the top 10, median 20, and lowest 10 percentile of ϵ corresponding to the given random walk, and plot them along with the mixing time derived using SLEM where

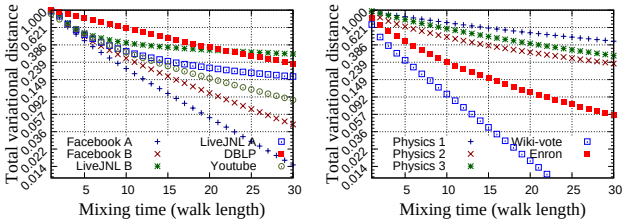


(a) Lower bound (b) Average mixing time

Figure 7: Lower-bound vs. the top the average mixing time for a sample of 1000 nodes in each data set, where DBLP x means the minimum degree in that data set is x.

the results are shown in Figure 8. We observe that for a million nodes graph, while the mixing time in the top 10% in the sample we computed is 100 for an averaged $\epsilon = 10^{-5}$—an excellent value to the "theoretical" guarantees of the Sybil defenses, the SLEM-based mixing time results in only $\epsilon = 10^{-2}$ as shown in Figure 8(i). We attribute this difference to similar scenario as in the physics co-authorship graphs. Similar observations can be seen in each of the different large social graphs. It is worth mentioning that Livejournal (Figure 8(k) and Figure 8(l)) present poor mixing in relation with Facebook data sets, which are shown to be fast mixing.

In Figure 9 we plot the average ϵ obtained when walking from the 1000 initial distributions as we increase t. We find that average ϵ is quite related to the underlying structure of graphs as well, as shown earlier for the lower bound.



(a) Large datasets (b) Small to medium datasets

Figure 9: The average mixing time of a sample of 1000 initial distributions in several social networks using the sampling method for computing the mixing time using the definition.

Finally, to understand the methodology used for experimenting in Sybilguard and SybilLimit, we perform the same trimming technique by iteratively removing lower degree nodes (for 1 up to 5) from the DBLP data set and computed the mixing time of the resulting graphs at each time (results shown in Figure 7). We observe that the pruning of lower degree greatly improves the mixing time of the social graph: for fixed mixing time of 100, by successive trimming the variation distance is reduced from about 0.2 to 0.03 (Figure 7(a)), and from about 0.015 to 0.002 (Figure 7(b)). But this improvement happens only with a huge reduction of the graph size: while DBLP 1 has 614,981 nodes, DBLP 5 has only 145,497 nodes. This means that about 75% of nodes are removed out of the social network, and could potentially be denied joining the service outright in order to boost the mixing time.

2.4.4 *Discussion*

While the main finding in this study is that the mixing time of social graphs is higher than has been used in literature, we also conclude that different nodes approach the stationary distribution at different rates. This is, while the majority of walks initiated from different nodes reach closer to the stationary distribution at "higher" rate than that of the mixing time, which is defined as the maximum rate from any source, we still find—except in a few cases of online social networks—that the mixing time of the majority of nodes is larger than anticipated and used in the previous studies [146, 147, 71]. This has several implications and call for several actions.

First, since most of the theoretical guarantees of social graphs consider the model in (4), and since the majority of nodes in the social graphs measured in this work have better mixing time than the bound in that model, this calls for rigorous study by basing such designs and analyses on the average case of the mixing, which is relatively small, instead of the worst case of the mixing time.

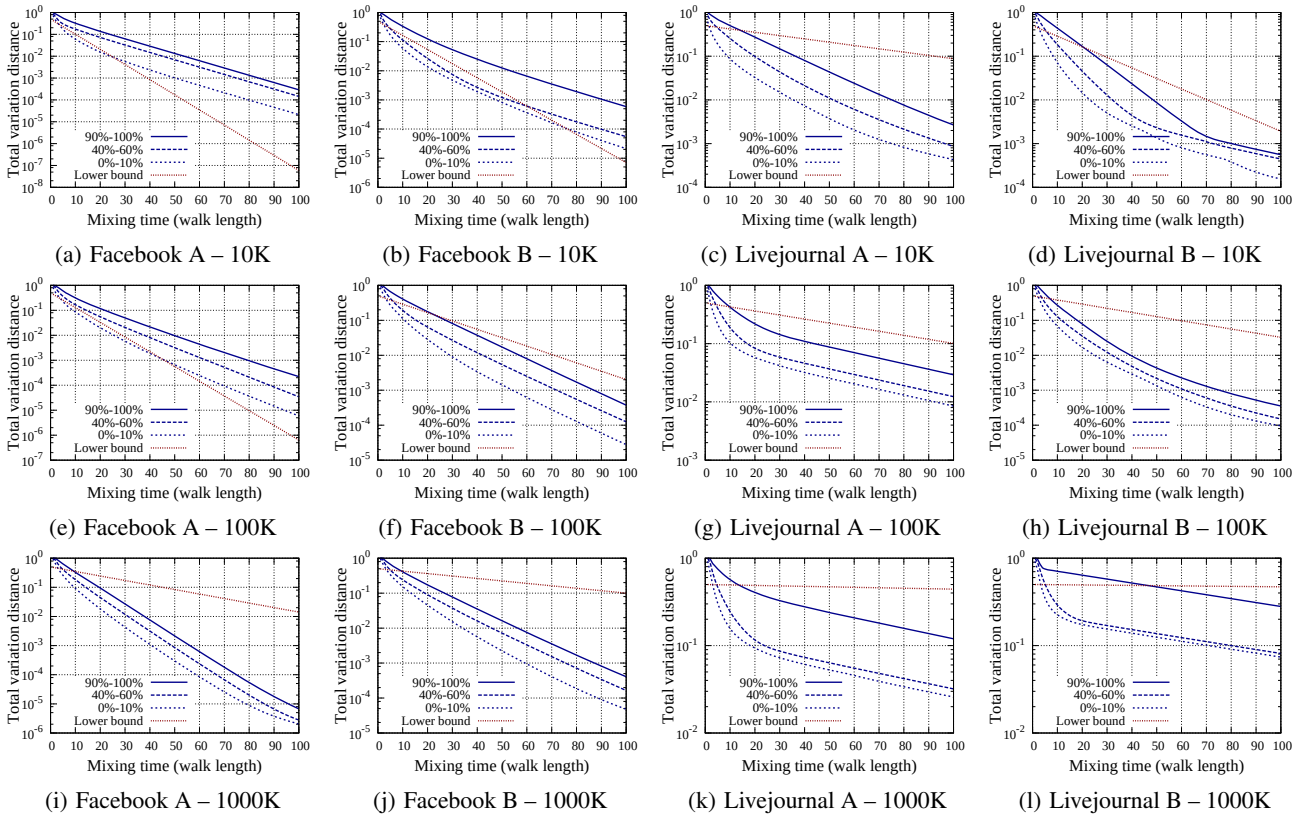
Second, the obvious implication of our findings is that one has to either give up some of the utility (service) guarantees—which are implied by that almost all honest nodes admit other honest nodes—by using relatively shorter walks, or give up part of the performance and security by enabling longer random walks in order to reach these isolated parts of the social graphs. Though this looks straightforward, going either way is not as simple as it seems. On the one hand, if one uses longer random walks in order to reach such isolated parts of the network it would be equally likely to escape to the Sbyil region which has a cut similar in its nature to that of the slower mixing part of the original social graph. On the other hand, using random walks shorter than the mixing time of the majority of nodes would also be at the expense of the utility; not only for the isolated part but also the faster mixing part as well. The end detection guarantees of the design would work as long as g, the number of attack edges is less than $\frac{n}{w}$.

Third, papers introducing SybilGuard, SybilLimit, and Whānau all did experiments on their schemes. Despite the short mixing time that these experiments use, their results seem to support that their schemes work as expected. The explanation of this is two part. First, the trimming of lower-degree nodes would shorten the mixing time. Second, although they claim that the social networks are fast mixing and as a part of the definition—which is also used in parts of the proofs for the theoretical guarantees—they insist $\epsilon = \Theta(1/n)$, this is a very strong burden to achieve and perhaps somewhat larger ϵ might also be good enough for these schemes to work. Also, we suspect that the difference between the average mixing time and the worst-case mixing time may have some effects on the discrepancy between the analysis and the experiment. In practice, the majority of nodes with "fast" mixing would be served and those few other nodes with very slow mixing would be denied service, which then will not be a problem for the probabilistic average-case guarantees. This last observation has been confirmed by authors of SybilLimit in [144] as a potential reason why Sybil defenses would still operate reasonably although graphs are not as fast mixing as assumed.

Finally, one of the assumptions in Sybil defenses based on social networks is that the used trust model requires physical acquaintance, which is the case in social networks such DBLP and Physics co-authorship networks, for which we show slower mixing time than other "online social networks" which are known to possess less strict trust models [44, 15], which by nature tolerate Sybil nodes. This calls for considering the trust model resulting from the underlying social network as a parameter, along with the mixing time, in order to evaluate the effectiveness of the social network-based defenses according to their real value. Our work in [96, 95] is a preliminarily result in this direction.

2.4.5 *Performance Implications—SybilLimit*

In order to quantitively measure the impact of the findings of slower mixing time on the performance of Sybil defenses, we implement SybilLimit and operate it on some of the different social



(a) Facebook A – 10K (b) Facebook B – 10K (c) Livejournal A – 10K (d) Livejournal B – 10K

(e) Facebook A – 100K (f) Facebook B – 100K (g) Livejournal A – 100K (h) Livejournal B – 100K

(i) Facebook A – 1000K (j) Facebook B – 1000K (k) Livejournal A – 1000K (l) Livejournal B – 1000K

Figure 8: Sampling vs. lower-bound measurements of the mixing time for 10K, 100K and 1000K of four large-scale datasets.

networks with the following settings. Since we already know the social graphs size—both m and n, we select the proper r that guarantees high probability of intersection. We set r to $r_0\sqrt{m}$, where m is the number of undirected edges in the graph and r_0 is computed from the birthday paradox to guarantee a given intersection probability. In this experiment, we consider the case without an attacker, since SybilLimit bounds the number of the Sybil identities introduced based on the number of the attacker edges. We increase t until the number of accepted nodes by a trusted node (the verifier) reaches almost all honest nodes in the social network. Then, with this t, we find the (average) total variation distance required in each graph, which is the necessary for the operation of these designs. It is then easy to compute the number of accepted Sybil identities which is $t \times g$, where g is the number of attack edges. SybilLimit works as long as $t < \frac{n}{w}$.

The result of this experiment is in Figure 10. We find that in some of these graphs the length of random walk is much longer than assumed previously in order to accept the majority of honest nodes. For example, even when the random walk length is 30 in Physics 1, we find that only 95% of the honest nodes are accepted, whereas the parameter r_0 we used ($= 4$) would ensure more than 99% acceptance rate if the graph is fast mixing. This happens to be the case for graphs like Facebook, where more 99% of the nodes are accepted by other honest nodes for a walk length of 6.

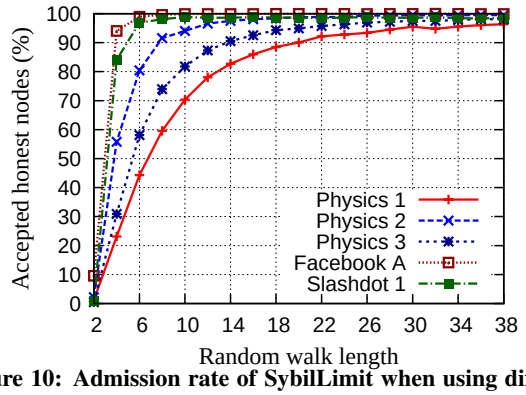


Figure 10: Admission rate of SybilLimit when using different t. Facebook (A) and Slashdot (1) have 10,000 nodes each.

2.5 Understanding the Mixing Time

A natural question following our measurements of the mixing time would be: what make a fast mixing time graph, fast mixing. Indeed graph measures, like conductance and modularity are very related to the mixing time. Here we seek a simple method to compute and indicate the mixing characteristics of these graphs.

2.5.1 The Mixing Time and k-Coreness

Among the measures of graph cohesiveness is k-coreness measure. For an undirected graph $G = (V, E)$ as defined earlier, and for any k, let $G_k = (V_k, E_k)$ be a subgraph in G such that $|V_k| = n_k$, and with the constraint that for all $v_i \in V$, the minimum degree of any node $v_j \in V_k$ is k. G_k is said to be a $k-$core of G if, in addition to the above condition, it is a maximal and connected graph. By relaxing the connectivity condition, we get a set of cores (potentially more than one) each of which satisfies the degree condition. For such (potentially disconnected) k-core, we define the normalized size as n_k/n.

An efficient algorithm for decomposing a simple graph to its

Table 1: Datasets, their properties and their second largest eigenvalues of the transition matrix

Dataset	Nodes	Edges	μ	Dataset	Nodes	Edges	μ
Wiki-vote [66]	7,066	100,736	0.899418	Enron [67]	33,696	180,811	0.996473
Physics 1 [67]	4,158	13,428	0.998133	DBLP [73]	614,981	1,155,148	0.997494
Physics 2 [67]	11,204	117,649	0.998221	Physics 3 [67]	8,638	24,827	0.996879
Facebook A [142]	1,000,000	20,353,734	0.982477	Facebook B [142]	1,000,000	15,807,563	0.992020
Livejournal A [87]	1,000,000	26,151,771	0.999387	Livejournal B [87]	1,000,000	27,562,349	0.999695
Youtube [87]	1,134,890	2,987,624	0.997972				

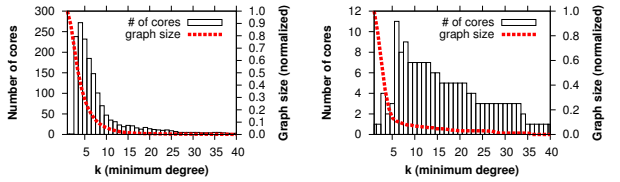
$k-$cores by iteratively pruning nodes with degree less than k has the complexity of $O(m)$ [13]. The definition of k-core [76] is equivalent to k-coloring [45]. For more details, see [104].

2.5.2 *Measurements and Results*

We use some of the datasets in Table 5 to demonstrate the relationship between core structure and the mixing time of social graphs. In short, the mixing characteristics of these graphs are shown in Figure 9, for the average case of the mixing time.

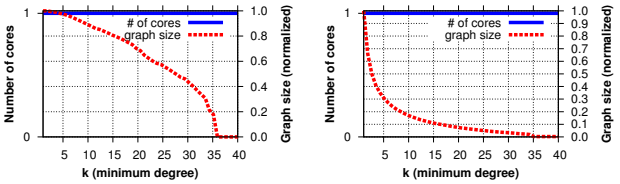
For each of these graphs we use an off-the-shelf implementation of the linear-time algorithm in [13] to compute the $k-$core by relaxing the connectivity assumption as described above. As k increases to its ultimate value at which the graph disappears, we compute the following: (1) the number of cores in each k-core, (2) the normalized size of each k-core. The results of these measurements are shown in Figure 11 and Figure 12 [106]. Notice that graphs in Figure 11 are slow mixing and graphs in Figure 12 are fast mixing, as demonstrated in Figure 9 for average mixing.

By comparing Figures 11 and Figure 12, we observe that slow mixing graphs are less cohesive whereas fast mixing graphs are more cohesive. This is reflected on the number of cores in the k-core of each graph as we increase k till the graph is dissolved entirely. Also, whereas slow mixing graphs—shown in Figure 11—are decomposed into multiple cores as we increase k, fast mixing graphs resist this decomposition and remain cohesive as we increase k.



(a) DBLP (b) Physics 1

Figure 11: Core structure (slow mixing social graphs).



(a) Wiki-vote (b) Slashdot

Figure 12: Core structure (fast mixing social graphs).

Second, despite that slow mixing graphs are decomposed into multiple cores, these cores are relatively small in size and the disappearance of the graph is very quick in many cases as we increase k. Fast mixing graphs have on the other hand remain in a single core, which is relatively larger in size than the counterpart core in slow mixing graphs—see [104] for more findings on core structure and experiments with other social graphs.

2.5.3 *Improving the mixing time*

Here, we propose several heuristics to improve the mixing time of slow-mixing social graphs using their structural properties which are discussed earlier in section 2.5.2.

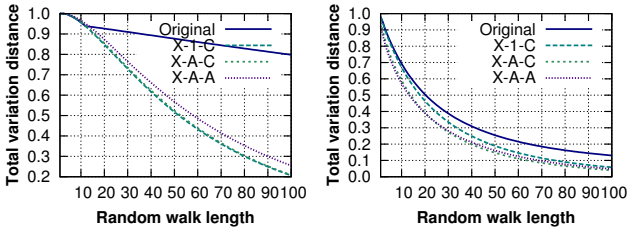
Heuristics to Improve the Mixing Time. Each of the following heuristics aims to prevent the creation of multiple cores as k increases using auxiliary edges. We call the process "core wiring". We introduce these heuristics with Sybil defenses [146] in mind as potential applications. We refer to the largest core as the *major core* and any other core is a *minor core*.

[1] **Heuristic X-1-C:** wires a single node in each minor core X with a node in the major core C using *one* edge. The end vertices of added edges can be arbitrarily chosen. By doing so, we can easily see that the graph will always have a single core at any time while increasing k. The number of added edges is the sum of the number of cores in each k-core, for all k, minus k.

[2] **Heuristic X-A-C:** wires each node in each of the minor cores with a node in the major component, as we increase k. Same as above, this would prevent producing multiple cores at time and the number of auxiliary edges is bounded by the *number of nodes* in the minor components.

[3] **Heuristic X-A-A:** wires 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.

For further discussions on the rational of this method, in relation with prior literature work, see [104]. In short, auxiliary edges added in our heuristics can be made part of the evolution of the social graph through link recommendation. Alternatively, when centralized systems are built on top of social networks, these edges can be virtually created among honest nodes if labels of nodes are given.

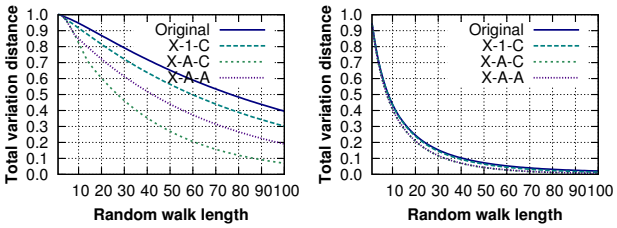


(a) Max (b) Mean

Figure 13: Mixing time measurement of Physics 1 before/after improving its mixing characteristics.

2.6 Measuring Other Properties

Here we outline our work on other properties used for trustworthy computing, the betweenness of nodes, and graph expansion.



(a) Max (b) Mean

Figure 14: Mixing time measurement of Physics 2 before/after improving its mixing characteristics.

2.6.1 *Betweenness and Sybil Attack*

MobID [115], a social-network based Sybil defense has recently attracted attention because it newly provides a robust defense for mobile environments while existing defenses have largely been designed for peer-to-peer networks. Furthermore, MobID introduces *betweenness*, a graph-theoretic property in the social graph, as a metric of the goodness of nodes in order to defend against the Sybil attacks. By using this betweenness, MobID operates on two fundamental assumptions: i) highly enmeshed nodes in the social graphs have a nonzero betweenness, and ii) verifiers and suspects in an honest social graph have common friends. To understand the extent to which assumptions about the betweenness in the social networks is valid, we perform extensive experiments on several datasets. On each of these graphs, mostly shown in Table 5, we use the shortest path betweenness defined earlier. The betweenness of the nodes (as a CDF) for some of these graphs is shown in Figure 15 (Further details are in [93]). From these measurements, we observe that big proportion (35% to 50%) of the nodes in the social graph even without considering malicious nodes, have betweenness close to zero). Such finding is striking, in the sense that the betweenness alone cannot be used to build applications and make meaningful judgment on nodes in social networks to identify honest and malicious nodes.

2.6.2 *Measurements of The Expansion*

We estimate the expansion factor of a graph by constructing an envelope Env_d (the same terminology used in [130]) formed by all nodes that are within a (shortest-path) distance i from a core node. The expansion Exp_i of the envelope consists of all of its immediate neighbors. We define the expansion factor as $\alpha_i = |Exp_i|/|Env_i|$. In our experiments, by letting each of the nodes in the graph to be the core, we calculate the expansion factor α_i, with $0 \leq i \leq d-1$, where d is the diameter of the graph, by building a tree rooted at the core that expands in the breadth-firth search manner. Let L_i be the number of nodes at level i in the tree, we have

$$\alpha_i = \frac{L_{i+1}}{\sum_{j=0}^{i} L_j} \tag{7}$$

To estimate the expansion characteristics of the social graphs, we run our experiment by letting each node in the graph to be the core and build a breadth-first search tree rooted at that core. We then count the number of nodes in each level of the tree to calculate the expansion factor as in Eq. (7).

To motivate for this measurement, consider the measurements in Table 2. In this experiment, we run Gatekeeper [130] on four different datasets with different characteristics. Unsurprisingly, we observe that results of operating Gatekeeper on such graphs are quite anticipated given that that such graphs are experimented on other social network-based Sybil defenses (in [106] and [136]), despite that other defenses require social graphs to be fast-mixing

Table 2: Numerical results of operating Gatekeeper [130] on top of different social graphs with different characteristics. 10 Attackers are selected randomly and 99 distributers are sampled in each case (attack edges are $131, 145, 277,$ and 344, respectively). f is a security parameter, honest acceptance percent is of the whole graph size and Sybil is per attach edge.

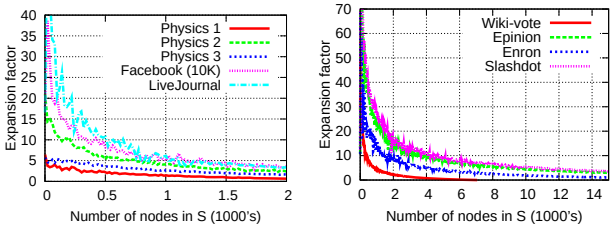
Dataset	Accept.	$f = 0.1$	$f = 0.3$	$f = 0.5$
Physics 1	Honest	89.90%	70.50%	54.40%
	Sybil	8.4	1.7	0.7
Facebook	Honest	98.40%	79.00%	51.30%
	Sybil	10.1	1.8	0.7
LiveJournal	Honest	97.00%	78.60%	53.20%
	Sybil	3.7	0.7	0.3
Slashdot	Honest	97.00%	81.10%	55.20%
	Sybil	3.1	0.8	0.4

whereas GateKeeper requires the graphs to be expanders with good expansion factor. Hence, we establish that the property in action is closely related to the mixing time.

Following the model in (7), we construct a breadth first search tree for each source and compute its expansion as we go down the stream in the search tree. For all sources in the graph, where each node is considered as a source of expansion, the running time of a naive implementation of our algorithm takes $O(nm)$, which is a manageable overhead for small to medium sized social graphs.

We develop this algorithm to compute the expansion for all datasets in Table 5. Each source node has $d-1$ measurements of sets of nodes and their neighbors, where d is the diameter of the graph. To visualize the tendency of the expansion as the set size increases, we aggregate the unique sizes of sets, and find the number of neighbors to each of them. For each set of neighbors to a unique size of nodes, we compute the maximum, minimum, and expected number of neighbors. These statistics of measurements for a selected set of datasets are shown in Fig. 16.

To further investigate the expansion of social graphs when compared to each other, we use the model in (7). For all sets of nodes with the same size, we compute the expected expansion as the average of all sizes of different neighbor sets. The result of the average expansion is shown is shown in Figure 17. The reader should keep in mind that these curves and α for each graph needs to be consider in relation with the graph size. For example, where one would see that slashdot's curve is above the curve of Wiki-vote, suggesting that expansion characteristics of Slashdot are better than wiki-vote, the size of wiki-vote is one-tenth the size of Slashdot.

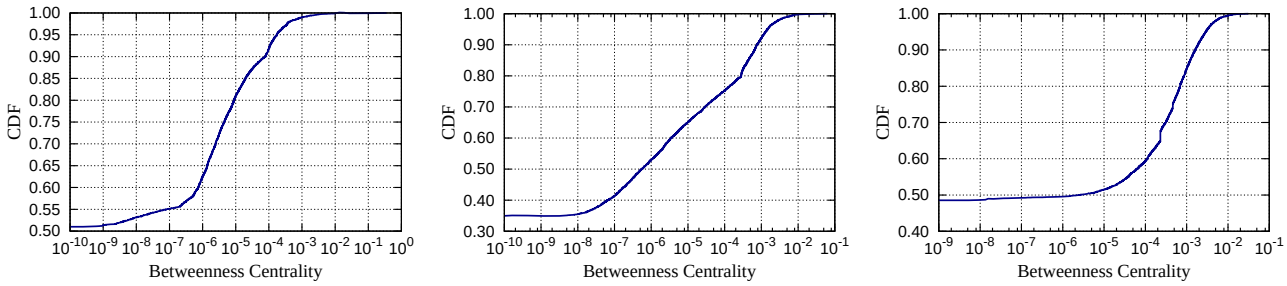


(a) Small datasets (b) Medium datasets

Figure 17: Expected expansion for various set sizes of various social graphs.

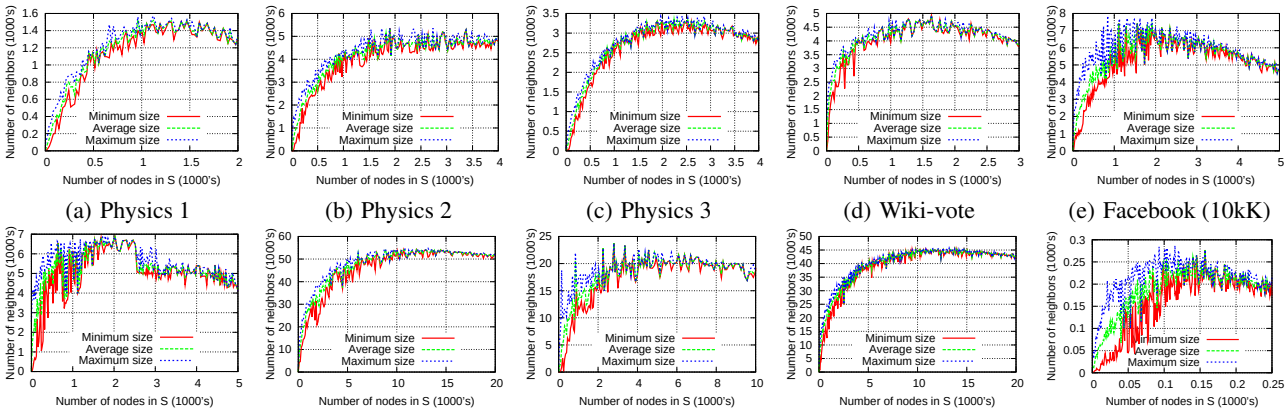
2.7 Proposed Work

To continue our earlier works measuring the mixing time and other properties, we want to consider the impact of graph manipulation and how it affects these properties. We will examine how omitting directions and sampling graphs according to difference



(a) Youtube (b) Wiki-vote (c) Physics 3

Figure 15: Preliminary measurements of the betweenness of nodes in different social graphs.



(a) Physics 1 (b) Physics 2 (c) Physics 3 (d) Wiki-vote (e) Facebook (10kK)

(f) LiveJournal (g) Slashdot (h) Enron (i) Epinion (j) Rice-Grad

Figure 16: A measured of the expansion of sets of nodes of different sizes beginning from every nodes in the given graphs as potential core for the expansion.

sampling algorithms would both influence the mixing time and the operation of Sybil defenses (and anonymous communications systems) when built on top of sampled (or modified by omission of directions) graphs.

2.7.1 *Social Graph Manipulation*

To make matters worse, some of these works—e.g., [146, 91, 19, 147, 148, 71, 136]—have considered "manipulated" social graphs among those mentioned above, which not only disguise the quality of the used trust assumption, but also manipulate the quality of the algorithmic property these systems use, and are supposed to find and use naturally in such graphs. Graph manipulation techniques include: 1) Trimming lower degree nodes which we already studied in [106] and studied in 2.4, 2) Omitting edge directions, by converting directed graphs to undirected ones and 3) sampling larger graphs to smaller ones and apply these designs on sampled graphs to relate to the applicability of these designs on the larger graphs.

2.7.2 *The Mixing Time of Directed Graphs*

Unfortunately, it is not clear how the process of altering social graphs by omitting directions would affects the quality of their mixing time. Although, the intuition is that directed social graphs (for which the mixing time is well-defined) would have different—and potentially slower—mixing time than undirected graphs. Motivated by the lack of prior work on this problem, we will investigate mathematical tools for measuring the mixing time of directed social graphs and its associated error bounds. We will use these tools to measure the mixing time of several benchmarking directed social graphs and to understand the difference in the mixing time quality between directed graphs and their undirected counterparts. We will then measure how this difference impacts two applications built on top of social networks: a Sybil defense mechanism and an anonymous communication system. Both applications are described in section 2.1. Using these findings, we will then provide several recommendations suitable for the design and experimenting these systems on top of social networks.

Our initial investigation on the problem shows that one does not need to compute an exact form of the stationary distribution of random walks on directed graphs. Under certain conditions on directed graphs—to make random walks on the graph both aperiodic and ergodic—-any walk on these graph would converge to the (unknown) stationary distribution. Allow very large walk length from any arbitrary distribution on the graph would allow for such convergence. The distribution of the walk after such large walk length can be used as an estimate of the stationary distribution. The remaining task would be to bound the error between the unknown stationary distribution and the estimated one and make it arbitrarily small.

Our initial results on measuring the mixing time of directed graphs using this method, without considering the error bounds which to be considered in the future, show that indeed directed graphs are often slower mixing than undirected graphs. This has consequences, as we anticipate such difference in the mixing time would affect the operation of systems built on the modified graphs using the inflated quality of the mixing time. Quantifying such overestimation of security (in Sybil defenses) and privacy (in anonymous communications) remain a future work to consider in the upcoming months.

2.7.3 *The Impact of Graph Sampling*

Sampling of large social graphs is used for addressing infeasibility of measurements in large social graphs, or for crawling graphs from online social network services where accessing an entire so-

cial graph at once is often impossible. Sampling algorithms aim at maintaining certain properties of the original graphs in the sampled (or crawled) ones. Several sampling algorithms, such as breadth-first search (BFS), standard random walk (RW), and Metropolis-Hastings (MH) random walk, among others, are widely used in the literature for sampling graphs. Some of these sampling algorithms are known for their bias, mainly towards high degree nodes, while bias for other metrics is not well-studied. In this direction, we propose to study these sampling algorithms and their bias when sampling the mixing properties of graphs.

The contribution of this work would be a comparative study on the bias of sampling algorithms in estimating the mixing time of social graphs. We want to consider several real-world social graphs with different structures, and several sampling algorithms and compare them according to their bias introduced on the measured mixing time. Our initial findings show that some existing sampling algorithms, even those which are unbiased to the degree distribution, always produce biased estimation of the mixing time of social graphs. We found that, unlike degree distribution that is easy to understand, and even possible to formally model for the amount of bias introduced due to sampling, the bias on the mixing time due to sampling has different patterns that are not captured in a single tendency. Characterizing this bias and better understanding it would be open question that we will try to understand in the upcoming months.

2.8 Related Work

There has been a lot of works in the literature that try to leverage properties of social networks for applications, that along with these properties weigh the value of trust between nodes, and an algorithmic property in the social graph. In this subsection, we summarize some of these works and the properties they use.

Sybil Defenses. It is probably unarguable that the most popular direction of using social network has been for defending against the Sybil defenses in a decentralized manner based on the fast mixing assumption of social graphs, where every passing network or security venue has a new result in this direction [136, 35, 69, 70, 106, 97, 95, 115, 96, 132, 129, 131, 145, 146, 147, 148, 72]. The direction has been initiated by Yu et al. [147, 148], where they used the fast mixing property of a graph to build a defense mechanism, called SybilGuard, that limits the number of Sybil identities introduced by *attack edges* (edges the connect malicious and honest nodes). In [145, 146], Yu et al. extended SybilGuard by introducing SybilLimit which further improves on the previous bounds of the number of Sybil identities per attack edges to near optimal. An optimal solution that uses similar ingredients from SybilLimit is introduced in [131, 129] by Tran et al. All of these systems use social graphs and assume that these graphs are fast mixing according to strict mixing definition mentioned in section 2.2. Danezis and Mittal [35] used the fast mixing property to build an inference (detection) mechanism for Sybil nodes in peer-to-peer Systems. Lesniewski-Laas et al. [71, 69] introduced a routing protocol that uses the fast mixing property of the social graph that builds a distributed hash table (DHT) system. Tran et al. introduced SumUp, a Sybil-resilient online content voting [132]. Mislove et al. in [88] introduced Ostra that leverages trust in social networks and thwart unwanted communication, and indirectly mitigating the impact of the Sybil attack. Kaustz et al. introduced ReferralWeb [61], a referral system that combines social networks and collaborative filtering and assumes a well-connected social network graph, a property that is very tied to the mixing time of the graph [58]. Similar filtering systems proposed in [65, 113] as well. Despite their dependence on the assumption for their performance, none of the papers mentioned above measured the mixing time from the social networks to show that they are fast mixing, in order to meet the guarantees being theoretical proven (for more exposition see [106]).

Anonymous communication. A very closely related set of designs that also use the mixing time is the construction of anonymous communication systems on top of social networks. Such studies can be found in [90, 109, 114]. Other related work tries to exploit social links for privacy preserving content sharing [56]

Betweenness to defend Sybil. Most recently, the betweenness has been proposed by Quercia et al. in [115] as a measure for the goodness of nodes: good nodes have a higher betweenness than a threshold, when they are incorporated into their friends' networks, and bad (or Sybil) nodes have lower betweenness than a threshold. However, whether this assumption hold in reality or not is not being tested on real-world social graphs that exhibit value of trust. Our work shows that this assumption does not hold in many online and other social networks.

Routing. Another active field of applications that leverages social networks is routing [36, 32, 80, 14, 26, 75, 82, 119, 86]. While they are in principle similar, and most try to exploit the closeness features of a small-world social graph, these works differ in the contexts they are proposed. For example, Davitz et al [36] introduces iLink, a search and routing utility on social graphs that can fit into "expert search" applications. Mabrouki et al. [80] exploit social networks for routing in sensor networks. Bigwood et al. [14] exploit social networks for routing in delay tolerant network (DTN). Chandrasekaran et al. [26] propose a solution for routing based on social network in P2P VoIP networks. Liben-Nowell et al. [75] study the potential of geographic routing in social networks. Marti et al. [82] study constructing a DHT-like system on top of social networks, and Sandberg [119] studies routing on top of social network in general contexts. Gossiping-based routing and information dissemination, on top of social networks, is studied in [46, 23, 4].

3. INCORPORATING DIFFERENTIAL TRUST

In most of the literature that considered social networks for building Sybil defenses, the simple uniform random walk highlighted earlier in the context of measuring the mixing time is used. As we have observed earlier, the mixing time of social graphs depends greatly on the underlying social graphs and there is a negative association between the strength of the social links and the mixing time. In part of the project, we propose to investigate several designs of modulated random walks that consider a "trust" parameter between nodes in order to tune the performance of the designs built on top social network, which use the random walk theory. In all of the proposed random walks, the purpose is to assign "trust-driven" weights and thus deviate from uniform random walk. Supported by initial preliminary results [97], we do this by either capturing the random walk in the originator or current node, as the case of originator-biased and lazy random walks, or by biasing the random walk probability at each node, as the case of interaction and similarity-biased random walks, or a combination of them. The intuition of the lazy and originator-biased random walk is that nodes trust "their own selves" and other nodes within their community more than others. On the other hand, interaction and similarity-biased trust assignments try to weigh the natural social aspect of trust levels.

It is worth noting that while this direction is motivated by the need for incorporating trust in social network-based Sybil defenses, the designs below are not limited to these Sybil defenses but can also be used without any modifications for any random walk based algorithm on graphs, including random walk based community discovery and random walk-based betweenness [112, 111]. Further-

more, these designs can be naturally and easily extended to other applications that use social networks as bootstrapping graphs, with minimal modifications. For example, the trust or distrust of nodes among each other can be incorporated as a random process driven from these models to identify decisions on collaboration among nodes. This collaboration can be used to understand the implication on the behavior of the different designs built on top of social networks. This latter direction is partly what we want to investigate further as part of the proposed work in project.

3.1 Designs to Account for Trust

Given the motivation for these designs, we now briefly describe them by deriving $\mathbf{P}$ and π required for characterizing them. We omit the details for lack of space (see [96] for the complete proofs, further experiments, and discussions).

3.1.1 Lazy Random Walks

To accommodate for the trust exhibited in the social graph, for simplicity we assume a global single parameter α in the network which is used to characterize this trust level. We use this parameter in the different schemes to enforce and apply the trust along with other parameters used for ensuring the (e.g., driven from the algorithmic property in the graph). The transition matrix

$$\mathbf{P}' = \alpha\mathbf{I} + (1-\alpha)\mathbf{P} \tag{8}$$

which yields a transition according to p_{ij} defined as follows:

$$p_{ij} = \begin{cases} \frac{1-\alpha}{\deg(v_i)} & v_j \in N(v_i) \\ \alpha & v_j = v_i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

We note that for the transition probability defined in ((8)), by adding self loops it does not alter the final stationary distribution from that in the uniform random walk. Further details on the proof of this are in [96].

3.1.2 Originator-biased Random Walk

We incorporate the concept of biased random on the social graph walks to characterize the bias introduced by the trust among different social actors (nodes). At each time step, each node decides to direct the random walk back towards the node that initiates the random walk, i.e., node v_r, with a fixed probability α or follow the original simple random walk by *uniformly* selecting among its neighbors with the total remaining probability $1-\alpha$. The transition probability that captures the movement of the random walk, initiated by a random node v_r, and moving from node v_i to node v_j is defined according to p_{ij} as follows

$$p_{ij} = \begin{cases} \alpha & j = r, v_r \notin N(v_i) \\ \alpha + \frac{1-\alpha}{\deg(v_i)} & j = r, v_r \in N(v_i) \\ \frac{1-\alpha}{\deg(v_i)} & j \neq r, v_j \in N(v_i) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

We note that, unlike the lazy random walks, the transition probability here considers moving the state back to the originator of the random walk, a state that may not be connected to the current state in the social graph. This requires a virtual connection between each node through the walk – every node in the graph – and each originator of a random walk. To mathematically model this transition loop, for each node $v_r (1 \leq r \leq n)$, we define $\mathbf{A}_r$ as an all-zero matrix with the exception of the r^{th} row which is 1's. Using $\mathbf{A}_r$, we further define the originator-biased transition matrix, for the walk originated from v_r, as

$$\mathbf{P}' = \alpha\mathbf{A}_r + (1-\alpha)\mathbf{P}. \tag{11}$$

We can show that $\mathbf{P}'$ is stochastic since each row in it sums to 1. Furthermore, since $\mathbf{P}'$ depends on the initial state v_r, we observe that the "stationary" distribution is not unique among all initial states, and so we refer to it as the "bounding distribution" for the walk initiated from v_r. The bounding distribution in that case is $\pi^{(v_r)} = [\pi_i]^{1\times n}$ where π_i is

$$\pi_i = \begin{cases} (1-\alpha)\frac{\deg(v_i)}{2m} & v_i \in V \setminus \{v_r\} \\ \alpha + \frac{\deg(v_i)}{2m} & v_i = v_r \end{cases} \tag{12}$$

We note also that the bounding distribution in ((12)) is a valid probability distribution since it satisfies the distribution conditions (sums to 1 and invariant to $\mathbf{P}'$). Details on the proof are in [96].

3.1.3 Interaction-biased Random Walk

The interaction between nodes can be used to measure the strength of the social links between nodes in the social network [142]. In this model, high weights are assigned to edges between nodes with high interaction and low weights are assigned to edges between nodes with low interaction. Formally, let $\mathbf{B}$ be the raw interaction measurements between nodes in G and $\mathbf{D}$ be a diagonal matrix representing the row norm of $\mathbf{B}$. The transition matrix $\mathbf{P}$ of the random walk based on interaction is then computed as $\mathbf{P}' = \mathbf{D}^{-1}\mathbf{B}$. The stationary distribution of the random walk on G following to the probability in $\mathbf{P}'$ is $\pi = [\pi_i]^{1\times n}$ where

$$\pi_i = (\sum_{j=1}^{n}\sum_{k=1}^{n} b_{jk})^{-1}(\sum_{z=1}^{n} b_{zi}). \tag{13}$$

We observe that this distribution makes a valid probability distribution since $\sum_{i=1}^{n}\pi_i = 1$ and is a stationary distribution since $\pi\mathbf{P} = \pi$.

Wilson et al. [142] introduced a slightly different model to capture interaction between nodes in the social graph. The interaction graph $G' = (V, E')$ is defined for a social graph $G = (V, E)$ where $E' \subseteq E$ and $e_{ij} \in E'$ if $I(v_i, v_j) \geq \delta$, where I is an interaction measure to assign weights on edges between v_i and v_j for all i, j, and δ is a threshold parameter. The interaction measure used in [142] is the number of interactions over a period of time. This later model further simplifies the random walk where the $\mathbf{P}'$ is defined over G', as well as the stationary distribution. In our measurements, we use this model and some of the datasets used in Wilson et al.'s work, though we do not exclude to further investigate the characteristics and potential of non-threshold based interaction model (the one described above) in the near future.

3.1.4 Similarity-biased random walk

The similarity between social nodes in social networks is used for measuring the strength of social links and predicting future interactions [29, 74]. For two nodes v_i and v_j with sets of neighbors $N(v_i)$ and $N(v_j)$, respectively, the similarity is $\frac{N(v_i)\cap N(v_j)}{N(v_i)\cup N(v_j)}$. For $\mathbf{a}_i$ and $\mathbf{a}_j$, two rows in $\mathbf{A}$ corresponding to the entries of v_i and v_j, we use the cosine similarity measure given as $S(v_i, v_j) = \frac{\mathbf{v}_i\cdot\mathbf{v}_j}{|\mathbf{v}_i|_2|\mathbf{v}_j|_2}$, where $|\cdot|_2$ is the L2-Norm. To avoid disconnected graphs resulting from edge cases, we augment the similarity by adding 1 to the denominator to account for the edge between the nodes. Also, we compute the similarity for adjacent nodes only, so that $\mathbf{S} = [s_{ij}]$ where $s_{ij} = S(v_i, v_j)$ if $v_j \in N(v_i)$ or 0 otherwise. The transition matrix $\mathbf{P}$ of a random walk defined using the similarity is given as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{S}$ where $\mathbf{D}$ is a diagonal matrix with diagonal elements being the row norm of $\mathbf{S}$. Accordingly, the stationary distribution of random walks on G according to $\mathbf{P}$ is $\pi = [\pi_i]^{1\times n}$ where $\pi_i = (\sum_{z=1}^{n} s_{zi})(\sum_{j=1}^{n}\sum_{k=1}^{n} s_{jk})^{-1}$.

Table 3: Social graphs with their size, diameter, and radius. Physics 1, 2, 3 are relativity, high energy and high energy theory co-authorship respectively [67].

Social network	Nodes	Edges	Diameter	Radius
Physics 1 [67]	4,158	13,428	17	9
Sdot [68]	10,000	14,6469	6	3
Physics 2 [67]	11,204	117,649	13	7
Physics 3 [67]	8,638	24,827	18	10
Wiki-vote [66]	7,066	100,736	7	4
Enron [67]	10,000	108,373	4	2
Epinion [117]	10,000	210,173	4	2
DBLP [73]	10,000	20,684	8	4
Facebook [142]	10,000	81,460	4	2
Livejournal [87]	10,000	135,633	6	3
Youtube [87]	10,000	58,362	4	2
Rice-cs-grad [89]	501	3255	9	5
Rice-cs-ugrad [89]	1221	43153	6	3

3.1.5 Implication of the designs on the mixing time

Along with the simple random walk-based design, we implement three of the proposed designs: lazy, originator, and similarity biased random walks. We use the simple random walk-based implementation over the interaction graph of Wilson et al.'s [142] to learn the performance of the interaction-based model. We examine the impact of each design on the mixing time on some graphs from Table 3. The results are shown in Fig. 18 and Fig. 19. We observe that, while they bound the mixing time of the different social graphs, the originator-biased random walk is too sensitive even to a small α. For example, as in Fig. 19(a) for Facebook social graph in Table 3, $\epsilon \approx 1/4$ is realizable at $w = 6$ with the simple random walk, $w > 10$ for both lazy and originator-biased random walk. However, this happens with $\alpha = 0.5$ in the lazy against $\alpha \approx 0.1$ in the originator-biased walk. This observation is made clearer on Fig. 19 which compares the mixing time of four different social graphs with different characteristics when using the simple and modified random walks.

We also observe in Fig. 18 and Fig. 19 that the linear increments in the parameters do not necessarily have linear effect on the measured mixing time. Furthermore, this behavior is made clearer in the experiments performed on SybilLimit and shown in Fig. 20 and Fig. 21. This however is not surprising, at least with the originator-biased random walk since the probability of intersection when sampling from the stationary distribution is $\leq e^{-8(1-\alpha)^4}$ from which one can see the exponential effect of α on the admission rate. While this explains the general tendency in the admission rates of SybilLimit, it does not answer some inconsistency shown in Fig. 21(b) for the transition between $\alpha = 0.12, 0.16$, and 0.20. One additional explanation for that is the community structure in this graph, which is shown in [136] to be clear in Physics 1 and problematic for Sybil defenses (results for the same graph are in Fig. 20(b) and Fig. 21(b)). On the other hand, some graphs are less sensitive to the same value of these parameters, e.g., Facebook with the results shown in figures 18(a), 19(a), 20(d), and 21(d). One possible explanation for this behavior is that this graph has less community structure. Reasoning about this behavior and its quantification is to be our future work.

Sybil defense performance over simple random walks To understand the necessary mixing time quality required for the operation of SybilLimit, we measure the performance of SybilLimit using simple random walks, where the evaluation metric is the percent of honest nodes accepted by other honest nodes. For each walk with length $w(0 \leq w \leq 30)$, we compute the number of accepted nodes as a percent out of $n(n-1)$—total verifier/suspect pairs. Since SybilLimit accepts nodes on edges only, it works for $w \geq 2$. The results are shown in Fig. 22 and the variable mixing time shown earlier is further highlighted by observing the percent of accepted nodes when varying w. We observe that, unlike claims in SybilLimit where one would expect 95% admission rate at $w = 4$, some graphs require $w = 30$; where graphs which admit high percent of nodes for small w are those with poor trust.

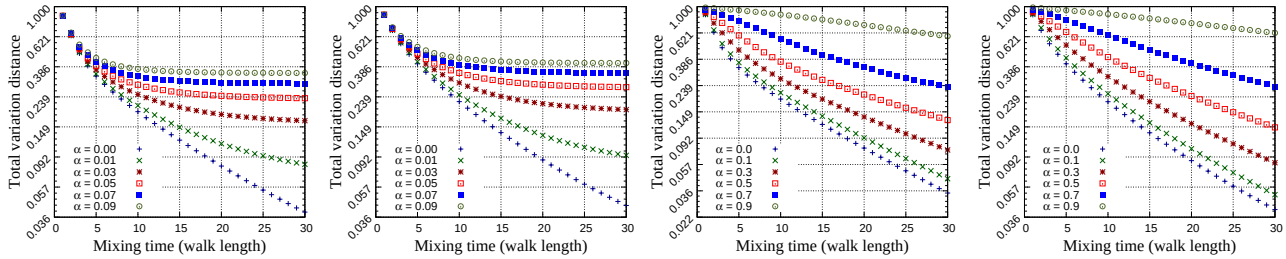
3.1.6 Defense Performance with Modified Walks

Now we study the impact of the modified random walks on the performance of SybilLimit. We select four datasets with different characteristics from Table 3: DBLP, Facebook, Facebook (Rice grad), and Physics 1 (relativity theory). We implement modified SybilLimit versions that consider changes introduced by the modified random walks and test the admission rate of honest nodes under different values of α and w.

Performance over lazy random walk we measure the performance of SybilLimit operating with the lazy random walks – results are shown in Fig. 20. We vary w from 0 to 30 with steps of 2. We further vary α associated with the lazy random walk from 0 to 0.80 with steps of 0.16—$\alpha = 0$ means simple random walk. While the performance of SybilLimit is generally degraded when increasing α, we observe that the amount of degradation varies and depends on the initial quality of the graph. For example, by comparing DBLP (Fig. 20(c)) to Facebook (Fig. 20(d)) we observe that for $w = 10$, DBLP and Facebook admit about 97% and 100% of the honest nodes respectively for $\alpha = 0$. For the same w and $\alpha = 0.64$, the accepted nodes in Facebook are still close to 100% while the accepted nodes in DBLP are only 50% suggesting variable sensitivity of different graphs to same α. Once we raise α to 0.80, the number of accepted nodes in Facebook decreases to 80% while giving only 25% in DBLP. One explanation of this behavior is what we have discussed in section 3.1.5. Also, since the ultimiate goal of this model is to characterize trust, which already differs in these graphs, we know that α should not necessarily be equal in both cases. For instance, if one is concerned about achieving same admission rate for the same w in both cases, one may choose $\alpha = 0.48$ in DBLP and $\alpha = 0.80$ in Facebook where $w = 10$ in both cases which yields 80% admission rate in both cases.

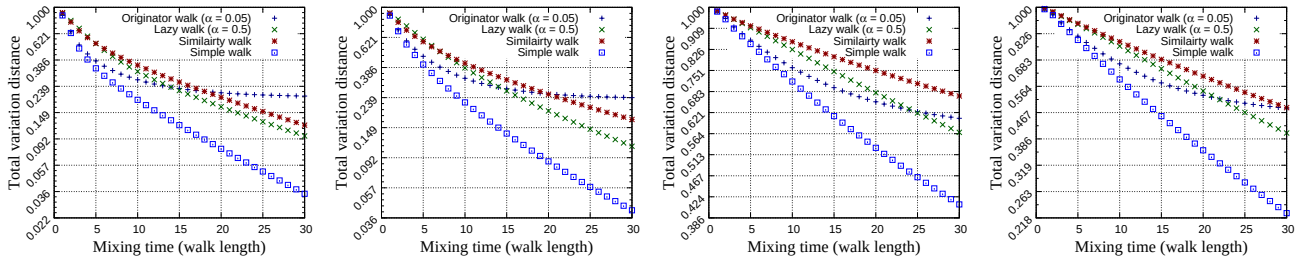
Performance over originator-biased random walk The same settings in section 3.1.6 are used in this experiment but here we vary α from 0 to 0.2 with 0.02 steps since the originator-biased walk is more sensitive to smaller α than the lazy-random walk. Similar to the lazy walk, the originator-biased walk, as shown in Fig. 21, influences the performance of SybilLimit on different graphs differently, and depending on the underlying graph. However, two differences are specific to the originator-biased walk over the lazy random walk.

First, the insensitivity shown earlier is even clearer in the originator-biased model. Second, while the end result of SybilLimit operating with lazy random walk is identical to the simple random walk if one allows long enough walk to compensate for the laziness, the behavior of the originator-biased walk is different. The indirect implication of the originator-assigned probability to herself is "discontinuity" in the graph (with respect to each node), where each node gives up some of the network by not trusting nodes in it. To cover the whole graph with that same α, w needs to be exponentially large. To challenge the insensitivity of the fast mixing social graphs, we extend α beyond the values used in Fig. 21 with Facebook from Table 3 and use $\alpha(0 \leq \alpha \leq 0.5)$ with 0.1 steps and



(a) Facebook A (b) Livejournal A (c) Facebook A (d) Livejournal A

Figure 18: The impact of the originator and lazy walks on the mixing time—(a) and (b) are for originator-biased while (c) and (d) are for lazy random walks.



(a) Facebook A (b) Livejournal A (c) Physics 1 (d) DBLP

Figure 19: The mixing time of four different social graphs when using simple vs. lazy, originator, and similarity-biased random walks, for each graph. While they are similar in size, a mixing time (parameterized by the same ϵ) is variable.

compute the admission rate. The result shows (not included here) that the originator-biased walk limits the number of accepted nodes, even in fast mixing graphs, but for larger α.

Performance over similarity and interaction-biased walk The similarity and interaction-biased random walks as used in this work are unparameterized. We compute the similarity for Facebook in Table 3, as explained in **??**. The similarity is then used to assign weights to edges between nodes, and bias the transition matrix. We run SybilLimit with similarity-biased random walks on Facebook in Table 3, where the result is shown in Fig. 23. In short, the similarity – while expected to capture some truth about the underlying graph – has less influence on the behavior of SybilLimit. It is however worth noting that the impact of the similarity-biased random walk is clearer on other social graphs, such as DBLP and Physics, which have clearer community structures.

For the interaction-biased design, we borrow the interaction graph of Wilson et al. [142] on Facebook. The interaction model introduces a richer model than the mere connections between nodes: it shows how strong are the links between nodes in the graph. With the same settings as earlier, we run SybilLimit – as a simple random walks – over the interaction graph. The results are shown in Fig. 23.

3.2 All designs: comparative study

Finally, we consider all designs at the same time. Because we only have interaction measurements for the Facebook dataset, we limit ourselves to that dataset. The result is shown in Fig. 23. While the performance of the similarity-biased random walk produces *almost* same results as the simple random-walk, the interaction-biased walk affects the number of the accepted nodes. Furthermore, the lazy random walk captures the behavior of model when deviated from the simple random-walk. As shown for this dataset, the interaction model behavior is characterized by the behavior of the lazy random walk for two given parameters ($\alpha = 0.48$ and $\alpha = 0.64$) suggesting that the interaction model can be further modeled as a lazy random walk where the problem is to find the proper parameters to match its behavior. Note that the value of α

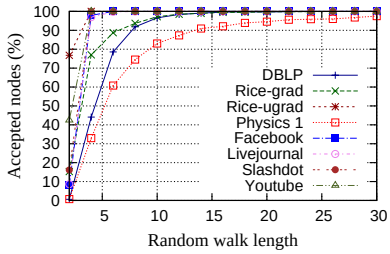
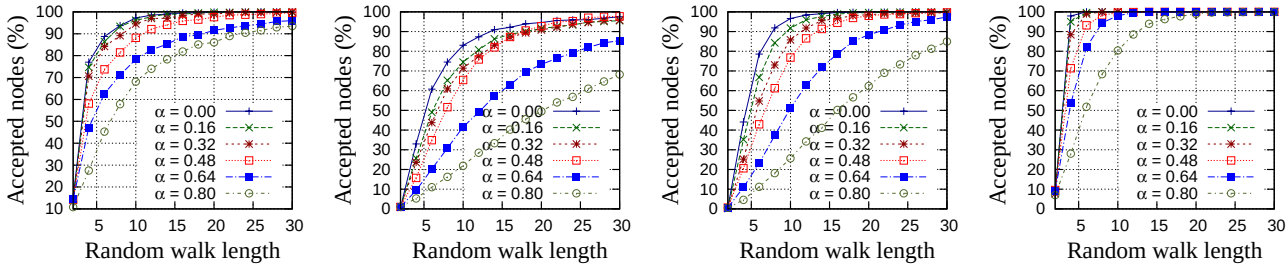


Figure 22: Accepted honest nodes in SybilLimit versus random walk length – with simple random walk. Different graphs have different quality of the algorithmic property though being with same size.

works for this dataset in particular. However, other datasets may be characterized by other values. We also find that the number of escaping tails per node is also decreased using our design, as shown in Fig. 24. In this last experiment, we compute the average escaping tails per 100 honest node samples, and by running the experiment 5 times, independently with a the given attackers edges for which nodes are selected uniformly at random from the honest region. In the experiment of Fig. 24, and for the interaction model, we assume that the attacker may infiltrate the social graph but cannot produce meaningful interactions, and thus the number of escaping tails to the attacker is always zero. It would be interesting in the future to generalize this model to an attacker with limited budget of interactions, and see how this changes the number of escaping tails with varying budgets. Finally to understand the impact of the different random walks on the accepted Sybil nodes per attack edge, we experiment with the same dataset (Facebook) and for varying g. The results are shown in Figure 25. Similar to above, our designs outperform the uniform design.

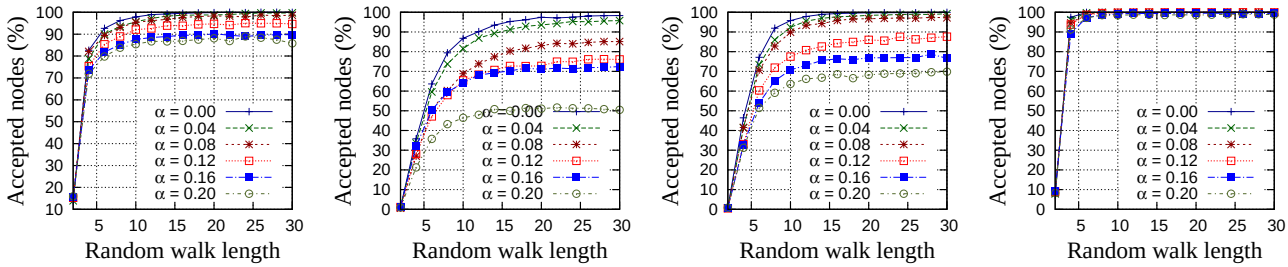
3.2.1 *Implications of Findings*

To sum up, we find in this study that one can control the behavior



(a) Rice CS Grad (b) Physics 1 (c) DBLP (d) Facebook

Figure 20: The performance of SybilLimit measured for accepted honest nodes when using different lengths of lazy random walk for different social graphs.



(a) Rice CS Grad (b) Physics 1 (c) DBLP (d) Facebook

Figure 21: The performance of SybilLimit depends on the underlying social graph, where different graphs require different walk lengths to ensure the same number of accepted nodes. The originator-biased random walk can further influence the number of nodes accepted in each graph.

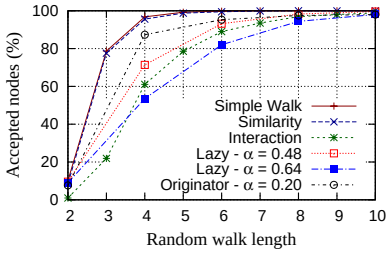


Figure 23: Accepted honest nodes in SybilLimit versus random walk length, when using the different designs to model the of trust in the social graph. The social graph of Facebook in Table 3.

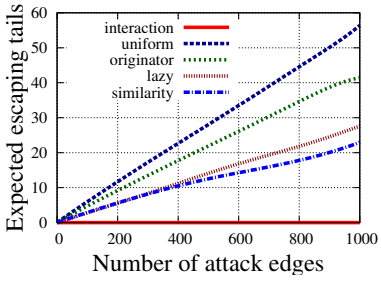


Figure 24: Expected escaping walks per node (among 100 nodes, $r = 850$) in Facebook dataset (in Table 3) where $w = 6$ and $\alpha = 0.4$ for both of the originator and lazy random walks.

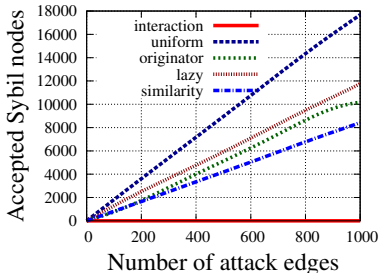


Figure 25: Accepted Sybil nodes over tainted tails when varying $g$in Facebook dataset (in Table 3) where $w = 6$ and $\alpha = 0.4$ for both of the originator and lazy random walks.

of the social network-based Sybil defenses by incorporating parameters for trust. For this purpose, we introduced and experimented the behavior of four designs. In graphs that are empirically-proven to be fast mixing and well-performing for the utility of the Sybil defense – though having poor value of trust – we have shown that one can select the necessary parameters to account for trust and make the performance of the defense on that graph equivalent to stronger and richer version of the same graph – e.g., the case of the interaction-based model versus the mere connections on the Facebook dataset. With these designs being intuitive in characterizing trust, the results being in agreement one another, and with this work being the first of its own type in this direction, we believe that this study is a first step in the direction of bringing well-received theoretical results into practice. The implications of our findings can be summarized as follows.

First, the mixing time and utility of the Sybil defense depend on the underlying graph. Through measurements, we supported our hypothesis that the quality of the social graph depends on the characteristic of the social links between the nodes. On one hand,

social links that are easier to make result in well-enmeshed graphs but are bad in principle for the Sybil defense since they already tolerate bad edges. However, these are shown to provide good honest nodes acceptance rate even with shorter random walks. On the other hand, social links that are harder to make result in graphs with more community structure, which are bad for the detection (as shown in [136]) and require longer walks to operate for the honest nodes.

Second, it is now possible for the Sybil defense operator, when given multiple options of social graphs, to further derive the utility of the Sybil defense using several criteria. Our study empowers the operators by an additional dimension that influences the behavior of the Sybil defense: trust.

Third, our findings answer a recently called for question in [136] of studying the behavior of Sbyil defenses when operated on the interaction-based model rather than the mere social connections, which are sometimes less meaningful. In short, our study shows that the interaction model can influence the behavior of the Sybil defense, by requiring longer random walk for the defense to work for honest nodes. However, this finding also suggests that a more community-structure is in the interaction model than in the mere social graph. This implies that, while the original social graph does not possess clear community structure, the use of the interaction model would add sensitivity for the detection part of the defense and result in weaker detection. However, the underlying graphs in both cases are different and the interpretation of the results should also consider the trust value in the interaction model, which is a better fit to the trust required in the Sybil defense.

Finally, online social graphs are known to possess weaker value of trust [44]. However, their potential for being used for Sybil defenses is very high since alternatives are limited, too expensive, and may not fit into the Sybil defense settings. For example, co-authorship social graphs which are known for their trust value may not necessarily include most users of a particular online system that tries to deploy the Sybil defense. On the other hand, given the popularity of online social networks, Sybil defenses may benefit from them, across systems and networks. To this end, the main finding of the work is to open the door wide for investigating trust, its modeling, and quantification for these systems.

3.3 Proposed work

In this part of the project, we propose to investigate three directions as follows

- The potential of the designs above of capturing further properties of the social graphs in the context of applications built on top of these graphs. While these primitives affect the quality of the social networks properties (such as the mixing time), it is unclear how they affect other properties that can be also used for building applications on top of social networks. For example, on a graph that uses interactions rather than social connections, one would expect more clear community structure that corresponds to different betweenness, closeness, and clustering coefficient characteristics than that of the original social graph that considers social links only. This hypothesis is particularly supported by our preliminary findings that interaction graphs have a lower mixing than that of the original social graphs using social links. We would like to investigate this direction emphasizing on the properties previously measured and how these designs affect them. Further, we propose to investigate how the change in the quality of the properties affect the applications on top of the social graphs.

- In both of the parameterized design that we have suggested in our prior work (the originator-biased and the lazy), we have considered some assumptions to simplify the operation of the designs. For example, we considered that each design uses a globally fixed parameter (α) that is used by each node in the graph. While this simplifies the analysis and make it possible to derive a clean model for the transition probability and the bounding (or stationary) distribution of the random walk of the graph, it is natural to consider different random walks with mixed values for the same parameter – assigned locally by each node depending on its perception of the overall graph and trust in it. Providing clean formulation for this node-wise (as opposed to graph-wise) parameters stay an open question that we would like to investigate further. This particularly will be more meaningful in designs that do not use random walks for their operation, and would be more challenging in designs that operate on other measures, such as centralities. Indeed, investigating how a node-wise parameter, as opposed to network-wise parameter is an area worth of investigation in its own right.

- We want to extend these results and findings by applying these designs to other Sybil defenses, other routing algorithms (initial results that tested how a simple variation of these designs impacts shortest path and random walk routing, as well as gossiping, is to appear in [94]).

4. NEW APPLICATIONS

In this part of the thesis, I intend to study new applications on top of social networks that rely on new assumptions that are easy to achieve. Two applications are proposed: 1) SocialCloud – a distributed computing service on top of social networks, and 2) Dynamix – an anonymous communication system that uses dynamic social networks as a bootstrapping structure for source/destination anonynmity in low-latency systems

4.1 Distributed Computing on Social Networks

In this direction, we oversee a new type of computing paradigm, called SOCIALCLOUD, that enjoys parts of the merits provided by the conventional cloud. Imagine the scenario of a computing paradigm where users who collectively construct a pool of resources perform computational tasks on behalf of their social acquaintance. Our paradigm and model are similar in many aspects to the conventional grid-computing paradigm. It exhibits such similarities in that users can outsource their computational tasks to peers, complementarily to using friends for storage, which is extensively studied in literature. Our paradigm is, however, very unique in many aspects as well. Most importantly, our paradigm exploits the trust exhibited in social networks as a guarantee for the good behavior of other "workers in the system". Accordingly, the most important ingredient to our paradigm is the social bootstrapping graph, a graph that is used for recruiting workers for a social network.

This popularity of social networks has opened the door wide for investigating the potential of these networks for many applications. Problems that are unsolvable in the cyberspace are easily solvable using social networks, for that they possess both algorithmic properties—such as connectivity—and trust, which are used to reason about the behavior of honest users in the social network, and limit the misbehavior introduced by other malicious users supported by efficiency features. Most important to the context of our paradigm is the aggregate computational power of nodes in the social network. Indeed, beyond the nodes and social links, the social networks consist of users with computing machines that are idle for

most of the time [12]. Furthermore, owners of these computing machines might be willing to share their computing resources for their friends, and for a different economical model than in the conventional cloud computing paradigm—fully altruistic one. This behavior makes our work share commonalities with an existing stream of work on creating computing services through volunteers [141, 25]. Our results hence highlight technical aspects of this direction and pose challenges for designs options when using social networks for recruiting such workers and enabling trust.

4.1.1 Contributions

To this end, our contribution in this direction is mainly:

- First, we investigate the potential of the social cloud computing paradigm by introducing a design that bootstraps from social graphs to construct distributing computing services. We advocate the merits of this paradigm over existing ones such as the grid computing paradigm.
- Second, we verify the potential of our paradigm using simulation set-up and real-world social graphs with varying social characteristics that reflect different, and possibly contradicting, trust models. Both graphs and the simulator are made public [102] to the community to make use of them, and improve by additional features.

4.1.2 The Case for SocialCloud

In this work, we look at the potential of using unstructured social graphs for building distributed computing systems. These systems are proposed with several anticipated benefits in mind. First, such systems would exploit locality of data based on the applications they are intended for, under the assumption that the data would be stored at multiple locations and shared among users represented in the social network—see §4.1.5 and [141] for concrete examples of such applications. This is in fact not a far-fetched assumption. For example, consider a co-authorship social graph, like the one used in our experiments, where the SocialCloud is proposed for deployment. In that scenario, data on which computations are to be performed is likely to be at multiple locations; on machines of research collaborators, co-authors, or previous co-authors. Even for some online social networks, the assumption and achieved benefits are not far-fetched as well, considering that friends would have similar interests, and likely to have contents replicated across different machines, which could be potentially of interest to use in our computing paradigm. Examples of such settings include photos taken at parties, videos—for image processing applications, among others.

The second advantage of this paradigm is its trustworthiness. In the recent literature, there has been a lot of interest in the distributed computing community for exploiting social networks to perform trustworthy computations. Examples of these literature works include exploiting social networks for cryptographic signing services [143], Sybil defenses [147, 35, 146], and routing in many settings including the delay tolerant networks [14, 32]. In all of these cases, along with the algorithmic property in these social networks, the built designs exploit the trust in social networks. The trust in these networks rationalizes the assumption of collaboration in these built system, and the tendency of nodes in the network to act according to the intended protocol with the theorized guarantees. Same as in all of these applications, SocialCloud tries to exploit the trust aspect of the social network, and thus it is easy to reason about the behavior of nodes in this paradigm (c.f. §4.1.4).

Related to trust exhibited in the social fabric utilized in our paradigm, the third advantage is that it is also easy to reason about the recruitment of workers. In this context, workers are nodes that are willing to perform computing tasks for other nodes (tasks outsourcers). This feature, when associated with the aforementioned trust, is quite advantageous when compared to the challenge of performing trustworthy computing on dedicated workers in the conventional grid-computing paradigm, where it is hard to recruit such workers.

Finally, our design oversees an altruistic model of SocialCloud, where nodes participate in the system and do not expect in return. Further details on this model are in §4.1.4.

Grid Computing. While the SocialCloud uses a similar paradigm to that of the grid computing paradigm—in the sense that both try to outsource computations and use high aggregate computational resources, the SocialCloud is slightly different. In particular, in the SocialCloud, there is a pre-defined relationship between the task outsourcer and the computing worker, which does not exist in the grid-computing paradigm. We limit the computations to $1-$hop neighbors, which further improve trustworthiness of computations in our model.

4.1.3 Social Networks and Systems Bootstrapping

Social networks are so popular. Nine of the twenty most popular sites on the web are for social networking [1]. The top ten online social networking websites have more than 650 million of unique visitors per month in total. The most popular social network, Facebook [2] alone serves 250 million unique visitors per month, with more than 96 unique visitors per second. Such popularity of social networks has motivated so many designs, protocols, and applications on top of social networks. Examples include routing [14, 32, 36, 82], social gossip [4, 46, 23], and Sybil defenses [147] (c.f. §4.1.14). While they are different in the details of their operation, all of these designs and protocols weigh algorithmic properties (connectivity), trust, and collaboration in the underlying social networks, which are used for bootstrapping such systems.

4.1.4 Economics of SocialCloud

In our design we assume an altruistic model, which simplifies the behavior of users and arguments on the attacker model. In this altruistic model, users in the social network *donate* their *computing resources*—while not using them—to other users in the social network to use them for specific computational tasks. In return, the same users who donated their resources for others would anticipate others as well to perform their computations on behalf of them when needed.

One can further improve this model. Social networks are rich of trust characteristics that capture additional features, and can be used to rationalize this model in several ways. For example, trust in social networks, a well studied vein of research in this context [97], can be used to adjust this model so as users would bind their participation in computations to trust values that they assign to other users. In this work, in order to make use of and confirm this model, we limit outsourced computations at 1-hop.

While we do not consider that in this work (and would be a potential proposed work over the current findings), another model using interests and groups is worth mentioning for its popularity and potential as a future work. The incentives model can be further relaxed by enabling "interest" based model of computation where workers do computation to other nodes in the graph that only share some interest with them. This interest can be publicly identified by the membership of a node in a group. Investigating this model is left as a future work.

4.1.5 Use Model and Applications

For our paradigm, we envision compute intensive applications, for which other systems have been developed in the past using dif-

ferent design principles, but lacking trust features; where trust is needed in such applications and provided by our paradigm. These systems include ones with resources provided by volunteers, as well as grid-like systems, like in Condor [79], MOON [77], Nebula [25, 141], and SETI@Home [8].

Specific examples of applications built on top of these systems, that would as well fit to our use model, include blog analysis [141], web crawling and social-network applications (collaborative filtering, image processing, etc) [20], scientific computing [138], among others.

Notice that each of these applications requires certain levels of trust for which social ties are best suited as a trust bootstrapping and enabling tool. Especially, reasoning about the behavior of systems and expected outcomes (in a computing system in particular) would be well-served by this trust model. We notice that this social trust has been previously used as an enabler for privacy in file-sharing systems [56], anonymity in communications systems [109], and collaboration in sybil defenses [69, 146, 97], among others. In this work, we use the same insight to propose a computing paradigm that relies on such trust and volunteered resources, in the form of shared computing time. With that in mind, in the following section we elaborate on the attacker used in our system and trust models provided by our design, thus highlight its advantage and distancing our work from prior works in the literature.

4.1.6 Attacker Model

In this work, as it is the case in many other systems built on top of social networks [146, 147, 132], we assume that the attacker is restricted in many aspects. For example, the attacker has a limited capability of creating arbitrarily many edges between himself and other nodes in the social graph.

While this restriction may contradict some recent results in the literature [15]—where it is shown that some legitimate users befriend random users in the social network who are potentially attackers, it can be relaxed to achieved the intended trust and attack model by considering an overlay of subset of friends of each users. This overlay expresses the trust value of the social graph well and eliminates the influence introduced by the attacker who infiltrated the social graph [97]. For example, since each user decides on to which node among his adjacent nodes to outsource computations to, each user is aware of other users he knows well and those who are just social encounters that could be potential attackers. Accordingly, the user himself decides whether to include a given node in his overlay or not, thus minimizing or eliminating harm and achieving the required trust and attack model.

The description of the above attacker model might be at odds with the rest of the work, especially that we use some online social networks that do not reflect characteristics of trust required in our paradigm. However, such networks, when used, are used for two reasons. First, to derive insight on the potential of such social networks, and others that share similar topological characteristics, for performing computational tasks according to the method devised in this work. Second, we use them to illustrate that some of these social networks might be less effective than the trust-possessing social graphs, which we strongly advocate for our computing paradigm.

Comparison with Trust in Grid Computing Systems. While there has been a lot of research on characterizing and improving trust in the conventional grid computing paradigm [9, 10, 124, 59]—which is the closest paradigm to compare to ours, trust guarantees in such paradigm are less strict than what is expressed by social trust. For that, it is easy to see that some nodes in the grid computing paradigm may act maliciously by, for example, giving wrong computations, or refusing to collaborate; which is even easier to detect and tolerate, as opposed to acting maliciously [24].

4.1.7 The Design of SocialCloud

The main design of SOCIALCLOUD is very simple, where complexities are hidden in design choices and options. In SOCIALCLOUD, the computing overlay is bootstrapped by the underlying social structure. Accordingly, nodes in the social graph act as workers to their adjacent nodes (i.e., nodes which are one hop away from the outsourcer of computations). An illustration of this design is depicted in Figure 26. In this design, nodes in the social graph, and those in the SOCIALCLOUD overlay, use their neighbors to outsource computational tasks to them. For that purpose, they utilize local information to decide on the way they schedule the amount of computations they want each and every one of their neighbors to take care of. Accordingly, each node has a scheduler which she uses for deciding the proportion of tasks that a node wants to outsource to any given worker among her neighbors. Once a task is outsourced to the given worker, and assuming that both data and code for processing the task are transferred to the worker, the worker is left to decide how to schedule the task locally to compute it. Upon completion of a task, the worker sends back the computations result to the outsourcer.

Design Options: Scheduling Entity In the SOCIALCLOUD, two schedulers are used. The first scheduler is used for determining the proportion of task outsourced to each worker and the second scheduler is used at each worker to determine how tasks outsourced by outsourcers are computed and in which order. While the latter scheduler can be easily implemented locally without impacting the system complexity, the decision used for whether to centralize or decentralize the former scheduler impacts the complexity and operation of the entire system. In the following, we elaborate on both design decisions, their characteristics, and compare them.

- **Decentralized scheduler.** In our paradigm, we limit selection of workers to 1-hop from the outsourcer. This makes it possible, and perhaps plausible, to incorporate scheduling of outsourcing tasks at the side of the outsourcer in a decentralized manner—thus each node takes care of scheduling its tasks. On the one hand, this could reduce the complexity of the design by eliminating the scheduling server in a centralized alternative. However, on the other hand, this could increase the complexity of the used protocols and the cost associated with them for exchanging *states*—such as availability of resources, online and offline time, among others. All of such states are exchanged between workers and outsourcers in our paradigm. These states are essential for building basic primitives in any distributed computing system to improve efficiency (see below for further details). An illustration of this design option is shown in Figure 26. In this scenario, each outsourcer, as well as worker, has its own separate scheduling component.

- **Centralized Scheduler.** Despite the fact that nodes may only require their neighbors to perform the computational tasks on behalf of them and that may require only local information—which could be available to these nodes in advance, the use of a centralized scheduler might be necessitated to reduce communication overhead at the protocol level. For example, in order to decide upon the best set of nodes to which to outsource computations, a node needs to know which of its neighbors are available, among other statistics. For that purpose, and given that the underlying communication network topology may not necessarily have the same proximity of the social network topology, the protocol among nodes needs to

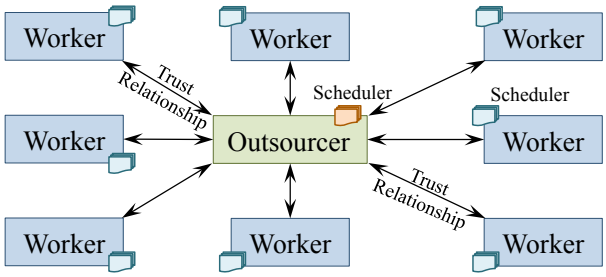


Figure 26: A depiction of the main SOCIALCLOUD paradigm as viewed by an outsourcer of computations. The different nodes in the social network act as workers for their friends, who act as potential jobs/tasks outsourcers. The links between social nodes are ideally governed by a strong trust relationship, which is the main source of trust for the constructed computing overlay. Both job outsourcers and workers have their own, and potentially different, schedulers.

incur back and forth communication cost. One possible solution to the problem is to use a centralized server that maintains states of the different nodes. Instead of communicating directly with neighbor nodes, an outsourcer would request the best set of candidates among its neighbors to the centralized scheduling server. In response, the server will produce a set of candidates, based on the locally stored states. Such candidates would typically be those that would have the most available resources to handle the outsourced computation task.

An illustration of this design option is shown in Figure 27. In this design, each node in SOCIALCLOUD would periodically send states to a centralized server. When needed, an outsourcer node contacts the centralized server to return to it the best set of candidates for outsourcing computations, which the server would return based on the states of these candidates. Notice that only states are returned to the outsourcer, upon which the outsourcer would send tasks to these nodes on its own—Thus, the server involvement is limited to the control protocol.

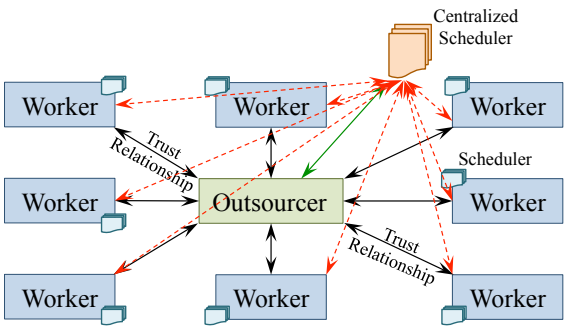


Figure 27: The decentralized model of task scheduling in SOCIALCLOUD.

The communication overhead of this design option to transfer states between a set of d nodes is $2d$, where d messages are required to deliver all nodes' states and d messages are required to deliver states of all other nodes to each node in the set. On the other hand, $d(d-1)$ messages are required in the decentralized option (which requires pairwise communication of states update). When outsourcing of computations is possible among all nodes in the graph, this translates into $O(n)$ for the centralized versus $O(n^2)$ communication overhead for the decentralized option. To sum up, Table 4 shows a comparison between both options.

Table 4: A comparison between the centralized and decentralized scheduler options. Compared features are resistance to failure, communication overhead, required additional hardware, and required additional trust.

Option	Failure	Communication	Hardware	Trust
Centralized	✖	$O(n)$	✖	✖
Decentralized	✔	$O(n^2)$	✔	✔

4.1.8 *Tasks Scheduling Policy*

While the use of distributed or centralized scheduling entity resolves the issue of scheduling at the outsourcer side, two decisions remain unsolved: how much computation to outsource to each node (worker), and how much time a node among these workers should spend on a given task for a certain outsourcer. We handle these two issues separately.

As mentioned earlier, any off-the-shelf scheduling algorithm can be utilized to decide the right scheduling policy at the side of the outsourcer, which can be further improved by incorporating trust characterization models for weighted job scheduling [97]. On the other hand, for workers scheduling, we consider several scheduling options as follows (notice that all of these policies are applied with respect to "computing time". This further requires estimating the time required for each task as a first step for using these policies).

- **Round Robin (RR) Scheduling Policy.** This is the simplest policy to implement, in which a worker spends an equal share of time on each outsourced task in a round robin fashion among all tasks he has.
- **Shortest First (SF) Scheduling Policy.** The worker performs shortest task first.
- **Longest First (LF) Scheduling Policy.** The worker performs longest task first.

Notice that we omit a lot of details about the underlying computing infrastructure, and abstract such infrastructure to "time sharing machines", which further simplifies much of the analysis in this work. In the results, we experiment with the three scheduling policies.

4.1.9 *Handling Outliers*

The main performance criterion used for evaluating SOCIALCLOUD is the time required to finish computing tasks for all nodes with tasks in the system. Accordingly, an outlier (also called a computing straggler) is a node with computational tasks that take a long time to finish, thus increasing the overall time to finish and decreasing the performance of the overall system. Detecting outliers in our system is simple: since the total time is given in advance, outliers are nodes with computing tasks that have longer time to finish when other nodes participating in the same outsourced computation are idle. Our method for handling outliers is simple too: when an outlier is detected, we outsource the remaining part of computations on all idle nodes neighboring the original outsourcer. For that, we

use the same scheduling policy used by the outsourcer when she first outsourced this task. In the simulation part, we consider both scenarios of handled and unhandled outliers, and observe how they affect the performance of the system.

4.1.10 *Deciding Workers Based on Resources*

In real-world deployment of a system like SOCIALCLOUD, we expect heterogeneity of resources, such as bandwidth, storage, and computing power, in workers. This heterogeneity would result in different results and utilization statistics of a system like SOCIALCLOUD, depending on which nodes are used for what tasks. While our work does not address this issue, and leaves it as a future work. We further believe that simple decisions can be made in this regard so as to meet the design goals and achieve the good performance. For example, we expect that nodes would select workers among their social neighbors that have resources and link capacities exceeding a threshold, thus meeting an expected performance outcome.

4.1.11 *Simulator of* SOCIALCLOUD

To demonstrate the potential of SOCIALCLOUD as a computing paradigm, we implement a batch-based simulator [102] that considers a variety of scheduling algorithms, an outlier handling mechanism, job generation handling, and failure simulation. A flow diagram of the simulator is in Figure 28.

The flow of the simulator, which represents the flow of the system, is depicted in Figure 28. First, the node factory uses the bootstrapping social graph to create nodes and their workers. Each node then decides on whether she has a task or not, and if she has a task she schedule the task according to her scheduling algorithm. If needed, each node then transfers code on which computations are to be performed to the worker along with the splits of the data for these codes to run on. Each worker then performs the computation according to the scheduling algorithm of the worker and returns the results of the computations to the outsourcer.

Timing. In SOCIALCLOUD, we use *virtual time* to simulate computations and resources sharing. We scale down the simulated time by 3 orders of magnitude of that in reality. This is, for every second worth of computations in real-world, we use one millisecond in the simulation environment. Thus, units of times in the rest of this work are in virtual seconds.

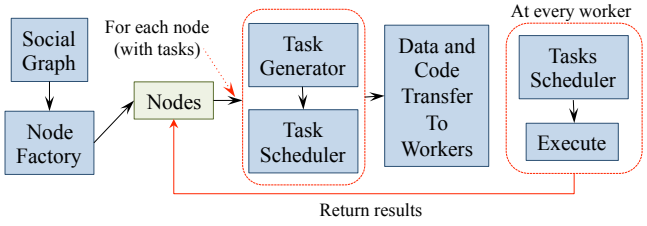


Figure 28: The flow diagram of SOCIALCLOUD: social graph is used for bootstrapping the computing service and recruit workers, nodes are responsible for scheduling their tasks by determining the amount of work each of its neighbors would process, and each worker (node) uses its local scheduler to determine how much time is allowed for each sub-task by its neighbors.

4.1.12 *Settings*

In this section, in order to derive insight on the potential of SOCIALCLOUD, we experiment with the simulator described above. Before getting into the details of the experiments, we describe the data and evaluation metric used in this section.

Evaluation Metric To demonstrate the potential of operating SOCIALCLOUD, we use the "normalized finishing time" of a task outsourced by a user to other nodes in the SOCIALCLOUD as the performance metric. We consider the same metric over the different graphs used in the simulation. To demonstrate the performance for the population of all nodes that have tasks to be computed in the system, we use the empirical CDF (commutative distribution function) as an aggregate measure. For a random variable X, the CDF is defined as $F_X(x) = P_r(X \leq x)$. In our experiments, the CDF measures the fraction (or percent) of nodes that finish their tasks before a point in time x, as part of the overall number of tasks. We define x as the factors of time of normal operation per dedicated machines, if they were to be used instead of outsourcing computations. This is, suppose that the overall time of a task is T_{tot} and the time it takes to compute the subtask by the slowest worker is T_{last}, then x for that node is defined as T_{last}/T_{tot}.

Tasks Generation Also for demonstrating the operation of our simulator, and the trade-off that such operation provides, we consider two different approaches for the tasks generated by each user. The size of each generated task is measured by virtual units of time, and for our demonstration we use two different scenarios:

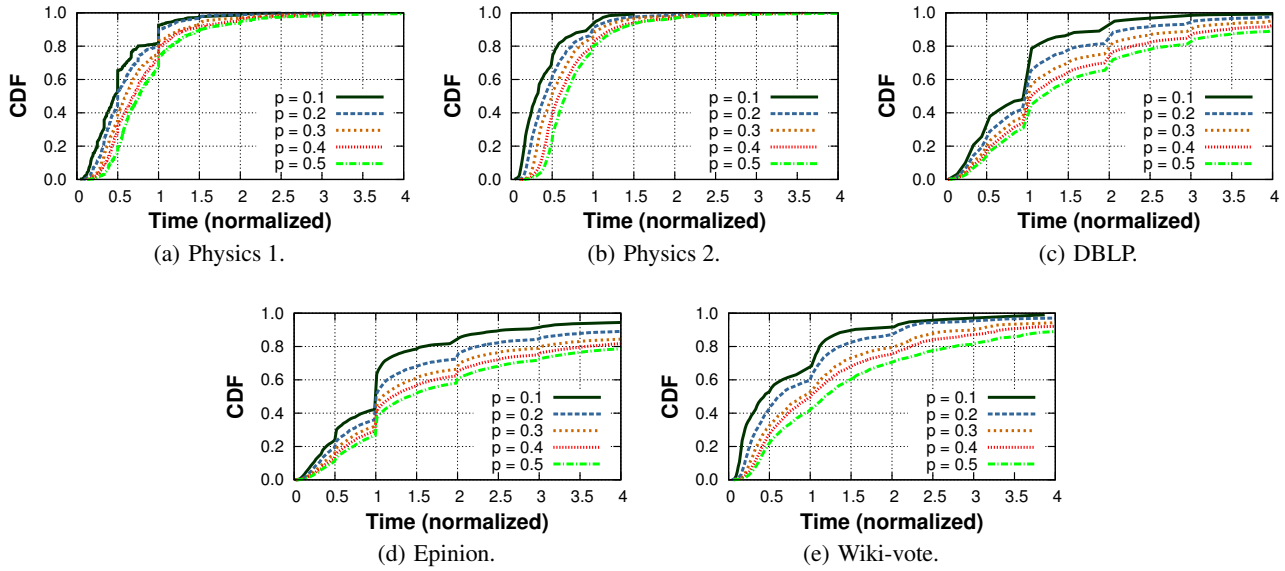
- **Constant task weight.** each outsourcer generates tasks with an equal size. These tasks are divided into equal shares and distributed among different workers in the computing system. The size of each task is $\bar{T}$.
- **Variable task weight.** each outsourcer has a different task size. We model the size of tasks as a uniformly distributed random variable in the range of $[\bar{T} - \ell, \bar{T} + \ell]$ for some $\bar{T} > \ell$. Each worker receives an equal share of the task from the outsourcer.

Deciding Tasks Outsourcers Not all nodes in the system are likely to have tasks to outsource for computation at the same time. Accordingly, we denote the fraction of nodes that have tasks to compute by p, where $0 < p < 1$. In our experiments we use p from 0.1 to 0.5 with increments of 0.1. We further consider that each node in the network has a task to compute with probability p, and has no task with probability $1 - p$—thus, whether a node has a task to distribute among its neighbors and compute or not follows a binomial distribution with a parameter p. Once a node is determined to be among nodes with tasks at the current round of run of the simulator, we fix the task length. For tasks length, we use both scenarios mentioned in §4.1.12; with fixed or constant and variable tasks weights.

Social Graphs To derive insight on the potential of SOCIALCLOUD, we run our simulator on several social graphs with different size and density, as shown in Table **??** (part of those used in Table 5). The graphs used in these experiments represent three co-authorship social structures (DBLP, Physics 1, and Physics 2), one voting network (of Wiki-vote for wikipedia administrators election), and one friendship network (of the consumer review website, Epinion). All of these graphs are made undirected, if they are not already, which rationalizes their use in our system. Notice the varying density of these graphs, which also reflects on varying topological characteristics. Also, notice the nature of these social graphs, where they are built in different social contexts and possess varying qualities of trust [97].

4.1.13 *Main Results*

In this section we demonstrate our paradigm and discuss the main results of this work. Due to the lack of space, we delegate



(a) Physics 1. (b) Physics 2. (c) DBLP.

(d) Epinion. (e) Wiki-vote.

Figure 29: The normalized time it takes to perform outsourced computations in SOCIALCLOUD. Different graphs with different social characteristics have different performance results, where those with well-defined social structures have self-load-balancing features, in general. These measurements are taken with round-robin scheduling algorithm that uses the outlier handling policy in §4.1.9 for a fixed task size (of 1000 simulation time units).

Table 5: Social graphs used in our experiments.

Dataset	# nodes	# edges	Description
DBLP	614981	1155148	CS Co-authorship
Epinion	75877	405739	Friendship network
Physics 2	11204	117649	Co-authorship
Wiki-vote	7066	100736	Voting network
Physics 1	4158	13428	Co-authorship

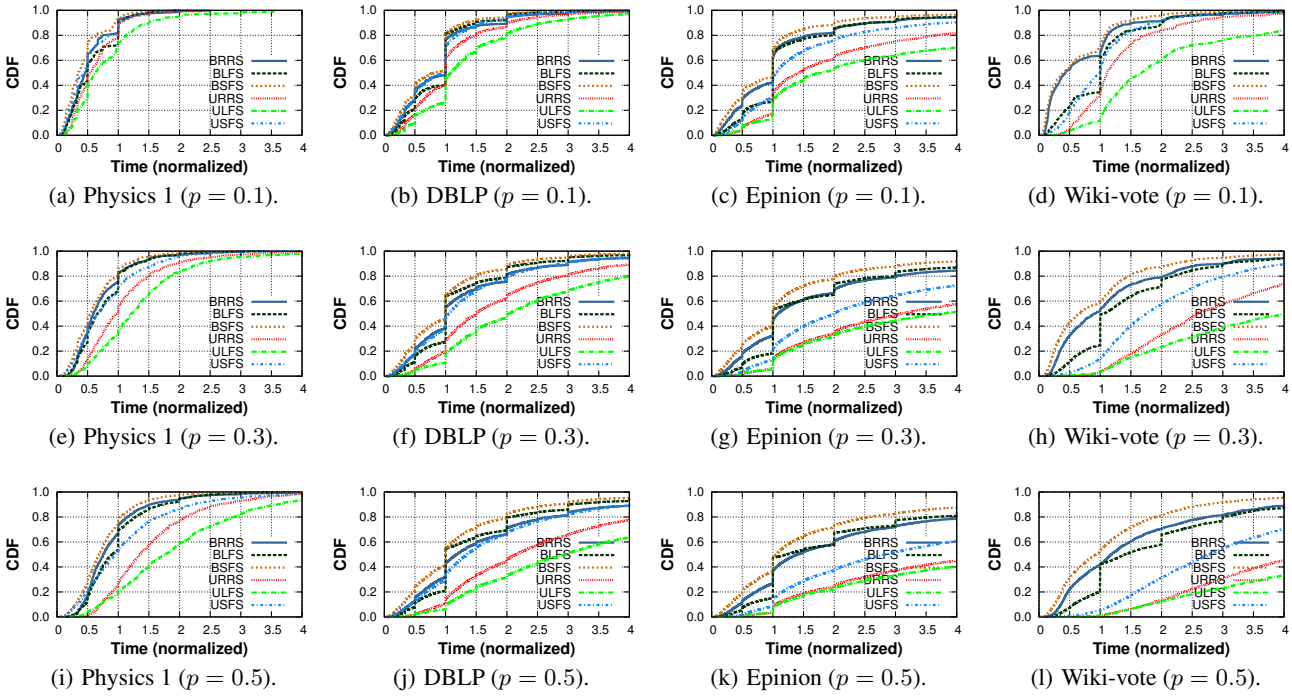
additional results to the technical report in [101]. For all measurements, our metric of performance and comparison is the normalized time to finish metric, explained in section 4.1.12.

Performance When Varying the Number of Outsourcers In the first experiment, we run our SOCIALCLOUD simulator on the different social graphs discussed earlier to measure the evaluation metric when the number of the outsourcers of tasks increases. We consider $p = 0.1$ to 0.5 with increments of 0.1 at each time. The results of this experiment are in Figure 29. On the results of this experiment we make several observations.

First, we observe the potential of SOCIALCLOUD, even when the number of outsourcers of computations in the social network is as high as 50% of the total number of nodes, which translates into a small normalized time to finish even in the worst performing social graphs (about 60% of all nodes with tasks would finish in 2 normalized time units). However, this advantage varies for different graphs: we observe that sparse graphs, like co-authorship graphs, generally outperform other graphs used in the experiments (by observing the tendency in the performance in figures 30(b) through 29(c) versus figures 29(d) and 29(e)). In the aforementioned graphs, for example, we see that when 10% of nodes in each case is used, and by fixing x, the normalized time, to 1, the difference of performance is about 30%. This difference of performance is observed between the Physics co-authorship graphs—where 95% of nodes finish their computations—and the Epinion graph—where only about 65% of nodes finish their computations.

Second, we observe that the impact of p, the fraction of nodes with tasks in the system, would depend on the graph rather than p alone. For example, in Figure 30(b), we observe that moving from $p = 0.1$ to $p = 0.5$ (when $x = 1$) leads to a decrease in the fraction of nodes that finish their computations from 95% to about 75%. On the other hand, for the same settings, this would lead to a decrease from about 80% to 40%, a decrease from about 65% to 30%, and a decrease from 70% to 30% in DBLP, Epinion, and Wiki-vote, respectively. This suggests that the decreases in the performance are due to an inherit property of each graph. The inherit property of each graph and how it affects the performance of SOCIALCLOUD is further illustrated in Figure 30. Interestingly, we find that even if DBLP is almost two orders of magnitude the size of Wiki-vote, for example, it outperforms Wiki-vote when not using outlier handling, and gives almost the same performance when using outliers handling.

Performance with different scheduling policies Now, we turn our attention to understanding the impact of the different scheduling policies discussed in §4.1.8 on the performance of SOCIALCLOUD. We consider the different datasets, and use $p = 0.1$ to 0.5 with 0.2 increments (the results are shown in Figure 31). The observed consistent pattern in almost all figures in this experiment tells that shortest first policy always outperforms the round robin scheduling policy, whereas the round robin scheduling policy outperforms the longest first. This pattern is consistent regardless of p and the outlier handling policy. The difference in the performance when using different policies can be as low as 2% (when $p = 0.1$ in physics co-authorship; shown in figure 32(b)) and as high as 70% (when using $p = 0.5$ and outlier handling as in wiki-vote (figure 31(l))). The patterns are made clearer in Figure 31 by observing combinations of parameters and policies.

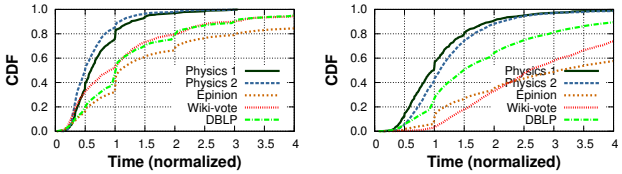


(a) Physics 1 ($p = 0.1$). (b) DBLP ($p = 0.1$). (c) Epinion ($p = 0.1$). (d) Wiki-vote ($p = 0.1$).

(e) Physics 1 ($p = 0.3$). (f) DBLP ($p = 0.3$). (g) Epinion ($p = 0.3$). (h) Wiki-vote ($p = 0.3$).

(i) Physics 1 ($p = 0.5$). (j) DBLP ($p = 0.5$). (k) Epinion ($p = 0.5$). (l) Wiki-vote ($p = 0.5$).

Figure 31: The normalized time it takes to perform outsourced computations in SOCIALCLOUD for different scheduling policies. Naming convention: U stands for unhandled outlier and B stands for handled outliers (Balanced). RRS, SFS, and LFS stand for round-robin, shortest first, and longest first scheduling.



(a) Handled outliers (b) Unhandled outliers

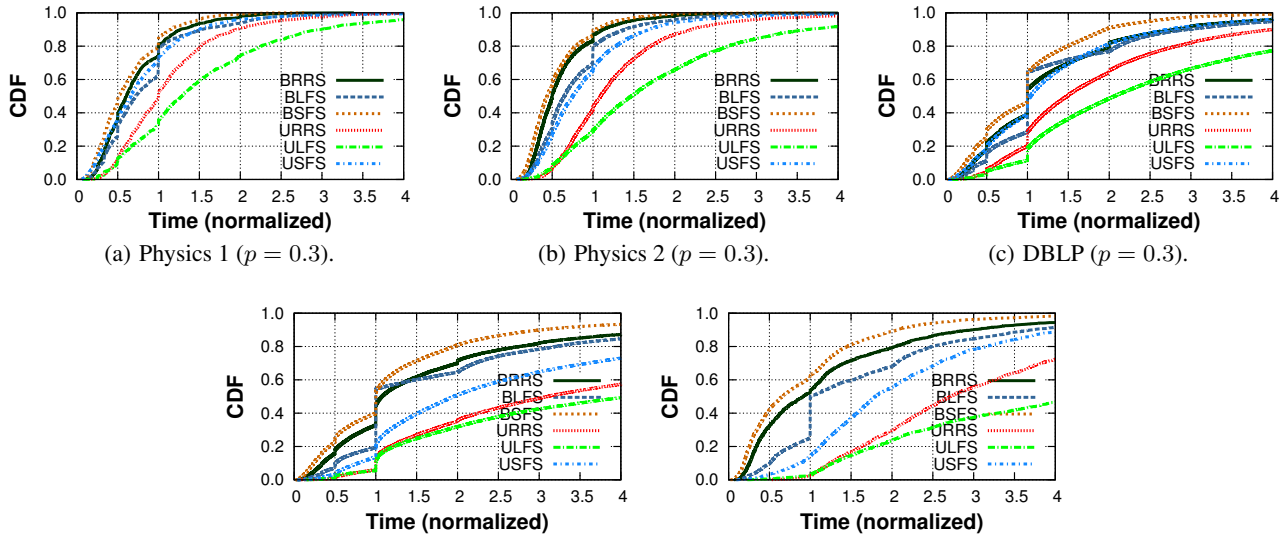
Figure 30: The performance of SOCIALCLOUD on the different social graphs used for our experiments, demonstrating the inherent differences in the different social graphs. Both figures use $p = 0.3$ and the round robin scheduling algorithm. Left figure is when handling outliers, whereas the right figure without handling the outliers.

Performance with Outliers Handling Outliers, as defined in §4.1.9, drag the performance of the entire system down. However, as pointed out earlier, handling outliers is quite simple in SOCIALCLOUD if accurate timing is used in the system. Here we consider the impact of the outlier handling policy explained in §4.1.9. The impact of using the outlier handling policy can be also seen on figure 31, which is used for demonstrating the impact of using different scheduling policies as well. In this figure, we see that the simple handling policy we proposed improves the performance of the system greatly in all cases. The improvement differs depending on other parameters, such as p, and the scheduling policy. As with the scheduling policy, the improvement can be as low as 2% and as high as more than 60%. When p is large, the potential for improvement is high—see, for example, $p = 5$ in Physics 2 with the round robin scheduling policy where almost 65% improvement is due to outlier handling when $x = 1$.

Performance with Variable Task Size In all of the above experiments, we considered computational tasks of fixed size; 1000 of virtual time units in each of them. Whether the same pattern would be observed in tasks with variable size is unclear. Here we experimentally address this concern by using variable duty size that is uniformly distributed in the interval of $[500, 1500]$ time units. The results are shown in Figure 32. Comparing these results to the middle row of Figure 31 (for the fixed size tasks), we make two observations. (i) While the average task size in both scenarios is same, we observe that the performance with variable task size is worse. This performance is anticipated as our measure of performance is the time to finish that would be definitely increased as some tasks with longer time to finish are added. (ii) The same patterns advantaging a given scheduling policy on another are maintained as in earlier with fixed task length.

Relationship Between Structure and Performance It is worth noting that the performance of SOCIALCLOUD is quite related to the underlying structure of the social graph. For example, sparse graphs such as co-authorship graphs—which are pointed out in [97] to be slow mixing graphs—are the graphs with performance advantage in SOCIALCLOUD. These graphs, in particular, are shown to possess a nice trust value that can be further utilized for SOCIALCLOUD. Furthermore, this trust value is unlikely to be found in online social networks which are prone to infiltration, making the case for trust-possessing graphs even stronger, as they achieve performance guarantees as well. This, indeed, is an interesting finding by itself, since it shows opposite outcomes to what is known in the literature on the usefulness of these graphs.

Additional Features and Limitations of Experiments Our sim-



(a) Physics 1 ($p = 0.3$). (b) Physics 2 ($p = 0.3$). (c) DBLP ($p = 0.3$).

(d) Epinion ($p = 0.3$). (e) Wiki-vote ($p = 0.3$).

Figure 32: The normalized time it takes to perform outsourced computations in SOCIALCLOUD, for variable task size.

ulator of SOCIALCLOUD omits a few details concerning the way a distributed system behaves in reality. In particular, our measurements do not report on or experiment with failure. However, our simulator is equipped with functionality for handling failure in the same way used for handling outliers (c.f. §4.1.9). Furthermore, our simulator considers a simplistic scenario of study by abstracting the hardware infrastructure, and does not consider additional resources consumed, such as memory and I/O resources. In the future, we will consider equipping our simulator with such functionalities and see how this affects the behavior and benefits of SOCIALCLOUD.

One last concern related to our demonstration of our paradigm is that we do not consider the heterogeneity of resources, such as bandwidth and resources, in nodes acting as workers in the system. Furthermore, we did not consider how this affects the usability of our system and what decision choices this particular aspect of distributed computing systems would have on the utility of our paradigm. While this would be mainly a future work to consider, we expect that nodes would select workers among their social neighbors that have resources and link capacities exceeding a threshold, thus meeting an expected performance outcome.

4.1.14 Related Work

There have been many works on the use of social networks for building communication and security systems, studying the performance of such designs on top of social networks, and analyzing the assumptions used in these designs as well. Below we highlight a few examples of these efforts and works.

Systems built on top of social networks include file sharing systems [56], anonymous communication systems [133, 109] Sybil defenses [35, 69, 145, 147], referral and filtering systems [61, 116], and live streaming [78]. Most of these applications weigh the trust in social graph, and an algorithmic property that makes the operation of these systems on top of social network effective. Another set of applications that exploit social networks' trust is routing [14, 32, 36, 82]—in several settings, where it has been shown that connectivity in social graphs can be of benefit in disconnected networks. Finally, assumptions of social network-based systems are explored recently, where Sybil defenses and their assumptions are studied in [107], and trust is challenged in [97].

Perhaps the closest vein of related work in the literature to our work is on the use of social networks for building computing services. Until the time of writing this work, most of the prior research work has been solely focused on providing storage services, but not a platform of computations. All of these systems share similar motivations as in our systems; they bring trust from an orthogonal context to empower a distributed computing system. In many cases, such storage services use slightly different economical model than the one used in SOCIALCLOUD, where payment per Megabyte per month rates are used as opposed to our eco-system. Examples of such efforts are reported by Tran et al. [128]), Chard et al. [28], and Sato [120]. Xu et al. [143] have further explored a first step in the direction of building cloud computing platforms on top of social networks by considering the access control model in this domain with preferred access control guarantees. The results of this work can be used as a building block in our work to improve the quality of access control and authorization.

Concurrent to our work, and following their work in [28], Chard et al. [27] suggested the use of social networks to build a resource sharing system. Whereas their main realization was still a social storage system as in [28], they also suggested that the same vision can be used to build a distributed computing service as we advocate in this work. Recent realizations of this vision have been reported in [126] and [62]. In [126], Thaufeeg et al. devised an architecture where "individuals or institutions contribute the capacity of their computing resources by means of virtual machines leased through the social network". In [62] Koshy et al. further explored the motivations of users to enable social cloud systems for scientific computing.

With similar flavor of distributed computing services design, there has been prior works in literature on using volunteers' resources for computations exploiting locality of data [25, 141], examination of programing paradigms, like MapReduce [37] on such paradigm [77, 20]. Finally, our work shares several commonalities with the grid and volunteer computing systems [79, 77, 25, 141, 8], of which

many aspects are explored in the literature. Trust of grid computing and volunteer-based systems is explored in [9, 10, 124, 59, 42]. Applications built on top of these systems, that would fit to our use model, are reported in [141, 20, 138], among others.

4.1.15 Proposed Work

In the near future we will look at two directions (one is long term direction that goes beyond this work) as follows.

- In the first direction, we aim to complete the missing ingredient of the simulator and enrich it by further scenarios of deployment of our design, under failure, with different scheduling algorithms at both sides of the outsourcer and workers (in addition to those discussed in this work), and to consider other overhead characteristics that might not be in line with topological characteristics in the social graph. These characteristics may include the uptime, downtime, communication overhead, and I/O overhead consumption, among others. One interesting feature that we will consider is trust-based scheduling, benefiting from the prior work in [97].

- In the second direction, and probably beyond this thesis work as a long term goal, we will turn our attention from the simulation settings to real-world deployment settings, thus addressing options discussed in §4.1.13, and to implement a proof-of-concept application, among those discussed in §4.1.5, by utilizing design options discussed in this work. We anticipate a lot of hidden complexities in the design to arise, and significant findings to come out of the deployment that we will report on in the future work.

- An additional proposed work, depending on whether the time permits or not, would be to investigate other usage models for this paradigm, beyond scientific computing where recruitment of workers and trust are issues for performing computations.

4.2 Anonymity on Dynamic Social Networks

All of the prior works which use social networks for building applications to solve real-world problems by exploiting these networks' structure and trust make certain assumptions. Some of these assumptions are rational and others are irrational in light of our understanding of the evolving settings of social networks. For example, the aforementioned social network-based Sybil defenses assume relatively simple trust model of binary relationships among nodes: a node either knows other nodes in the social graph or does not know them at all [146, 147, 35]. While simple and easy to implement, this model turns problematic when it does not represent reality by not characterizing the richer nature of social graphs that involves "differential trust" [97]. The idea has been recently used, and several designs have been considered to characterize trust for the problem in hand, and shown that differential trust can improve the security of Sybil defenses, despite degradation of algorithmic properties imposed by the differential trust [97].

Another simplifying assumption is the nature of associations among nodes; graphs are assumed to be static [146, 147, 35, 109, 32]. Insight is brought on the potential of these designs by experimenting with static social graphs, and by ignoring the dynamic nature of social graphs. Ignoring this nature might be due to unavailability of tools to capture the dynamic nature of social graphs, or unavailability of measures to quantify the performance of these designs on such dynamic social graphs. However, the limited nature of the static social graphs prohibits us from getting insight of these designs in reality when applied to real-world deployment settings for which social graphs are well-known for their dynamic behavior [63, 49, 125]. Such behavior greatly alters graphs structure, which is an essential determining part of the performance of these designs.

In this work we propose to proceed further to understand dynamic social graphs for another family of applications, anonymous communication systems. On the one hand, we would like to extend and utilize earlier findings in [109] and [34] of using social graphs as good mixers for anonymity. On the other hand, we want to improve on these results by formalizing the use of dynamic social structures for anonymity, and establishing a relationship between dynamic and weighted graphs. We want to show how our new paradigm enables anonymous communication and stands against possible attacks by empowering a richer social structure. We validate our model using empirical studies on two dynamic social structures driven from real-world social networks. To this end, our contributions are as follows:

- We formally define the problem of anonymously communicating on *dynamic* social structures that are natural in many contexts. We provide tools to quantify anonymity when such structures are used. In particular, we show an interesting theoretical connection between dynamic graphs and unweighted graphs representing history of associations among nodes and influencing the way these nodes are selected by their neighbors for anonymity.

- Using our model of dynamic structures, we provide detailed and extensive experiments to show the usefulness of our proposed model in reality considering two real-world network traces that exhibit dynamic structures.

4.2.1 Initial Model and Formulation

In this section we review the preliminaries of known literature on the problem, which are required for understanding the rest of this work. This known literature assumes a static graph. Unless otherwise is mentioned, this formalism follows from [109], which is to the best of our knowledge the only work that directly touches upon the problem.

System Settings and Application Scenario The idea of mixers over social links is very simple: users recruit their social acquaintance to provide anonymity to their traffic. In the nutshell, each node (user) forwards her own traffic to her friends, and friends forward that traffic to their friends, and so on, for a certain number of hops, say ℓ. The number of hops ℓ used for forwarding the traffic is a system-wide parameter, which is determined by the security level desired in the system. For simplicity, and without losing generality, let n be the number of users in the system. Accordingly, the anonymity is defined for two parties; the sender and the receiver of traffic. For the sender, the *anonymity set* is n, and the entropy of the probability distribution for a certain node being the sender is $H_s = \log_2(n)$. On the other hand, the anonymity set provided for the receiver is determined by the probability distribution achieved after the fixed number of hops used in the system. Let the distribution of the final node selected in a *random walk* after ℓ hops be π^ℓ, where $\pi^\ell = [\pi_i^\ell]^{1\times n}$, then the anonymity of the receiver of the traffic (the last hop in the walk) is H_r, which is given as:

$$H_r = -\sum_{i=1}^{n} \pi_i^\ell \log_2 \pi_i^\ell \tag{14}$$

Using (14), we define the *anonymity set* A^ℓ as

$$A^\ell = 2^{H_r} \tag{15}$$

Given a random walk on a graph with certain properties—see section 4.2.2 for details—that random walk has a unique bounding or

stationary distribution (defined in (16)) which captures the maximum achieved entropy.

The idea of using social networks to provide anonymity and empower privacy is appealing for several reasons. First, social networks have been known for certain algorithmic properties that make the maximal entropy of a random walk achievable within a few hops from any node in the social graph [109]. On the other hand, unlike anonymity on fully structured graphs, such as that provided by Tor [40]—in which recruiting relays and maintaining trusted ones is a challenge [57]—social network-based anonymity systems could be a alternative with rich trust characteristics. This trust is the main ingredient used for reasoning about the potential of these networks for safeguarding users privacy [56]. Indeed, social network-based anonymity systems, such as MCON [133] are known for their desirable characteristics, which are challenging in traditional mixing-based anonymous communication systems [127, 121, 54]. In the following, we further formalize the system settings as a graph-theoretic problem.

4.2.2 Formalization (For Static Graphs)

Assume a graph as defined in section 2.2.1. Define the Markov chain on the graph G following the transition matrix $\mathbf{P}$ which is defined according to $\mathbf{P} = [p_{ij}]^{n\times n}$ p_{ij} is defined in (3). A unique stationary distribution is defined for the Markov chain over the transition probabilities defined above if the Markov chain is ergodic—requiring it to be both *irreducible* and *aperiodic* [106]. Theorem 2 states such distribution.

THEOREM 2. *(Stationary distribution) For an undirected and unweighted graph G, the stationary distribution of the Markov chain defined over G according to transitions in* (3) *is the probability vector, given as* $\pi = [\pi_i]^{1\times n}$*, where*

$$\pi_i = \deg(v_i)/2m \tag{16}$$

Using the model in (14) and the distribution in (16), we define the maximal (in size) anonymity set following the same model as in (15) as $A^\infty = 2^{H_r^\infty}$, where:

$$H_r^\infty = -\sum_{i=1}^{n}(\frac{\deg(v_i)}{2m})\log_2(\frac{\deg(v_i)}{2m}) \tag{17}$$

Lower-bound on the Achieved Receiver Anonymity In [109], Nagaraja considered a gross definition of the achieved anonymity for every node in the social graph as a potential destination. This definition considered the average distribution achieved after ℓ hops from any potential source in the social graph. While this captures the average performance in the system, it simply does not show the worst case scenario observed at the lower-bound of the achieved anonymity for receivers. Here, we revise Nagaraja's definition in [109] and outline a straightforward fix for the measure of the anonymity provided in a system that uses walks on the social graph.

Without losing generality, let ℓ be a system-wide parameter, which represents the number of hops from the source to the destination (or receiver) in the graph, and each node between them is chosen uniformly at random from its predecessor. For each source v_j (for $1 \le j \le n$), we define the probability distribution after ℓ hops as $\pi^\ell(v_j) = [\pi_i^\ell(v_j)]^{1\times n}$ for (for $1 \le i \le n$). The anonymity achieved in the system is bounded below by the entropy achieved in the probability distribution obtained by walking from the worst source in the graph:

$$H_r \ge \inf_{v_j}\{-\sum_{i=1}^{n}\pi_i^\ell(v_j)\log_2\pi_i^\ell(v_j)\} \tag{18}$$

By extending (15) to the case in (18), we get the following

$$A^\ell = 2^{H_r} \ge 2^{\inf_{v_j}\{-\sum_{i=1}^{n}\pi_i^\ell(v_j)\log_2\pi_i^\ell(v_j)\}} \tag{19}$$

The intuition of this lower bound is very simple, and follows from the definition. Technically, this lower bound follows the classical theoretical trend in security: proving lower bounds of security (or anonymity as it is the case in hand) would enable us to guarantee, in the worst time, that our system would perform better than this bound for every user. On the other hand, considering the average case for achieved entropy might be very deceiving since many receivers are likely not to achieve this average bound.

Here, we extend the findings in the literature on using static graphs as mixers for anonymous communication to the case of the dynamic graphs. Such dynamic graphs arise naturally in many contexts due to social churn imposed by node and edge dynamics (joining and leaving social networks). It is worth noting that this is the first work of its own type to consider extending such results for building anonymous communication systems on top of dynamic social graphs.

4.2.3 Formalization (Dynamic Graphs)

The dynamic graph is a simple generalization of the static graph used in literature. In particular, $G = \{G^{(i)}\}$ for $1 \le i \le t$ is a dynamic graph over t time periods. Let $G^{(i)} = (V^{(i)}, E^{(i)})$ for $1 \le i \le t$, where $|V^{(i)}| = n^{(i)}$ and $|E^{(i)}| = m^{(i)}$, be an unweighted and undirected graph (later we extend that to the weighted graph case). Let $V^{(i)} = \{v_1^{(i)}, v_2^{(i)}, \ldots, v_{n^{(i)}}^{(i)}\}$ and $E^{(i)}$ be the set of pairs of vertices $v_j^{(i)}$-$v_k^{(i)}$ if both nodes $v_j^{(i)}$ and $v_k^{(i)}$ in $V^{(i)}$ are connected to each other. For $G^{(i)}$, we define $\mathbf{A}^{(i)}$ where $\mathbf{A}^{(i)} = [a_{jk}^{(i)}]^{n^{(i)}\times n^{(i)}}$ (the superscription is used as part of the notation, and does not mean power), where:

$$a_{jk}^{(i)} = \begin{cases} 1 & v_j^{(i)} \sim v_k^{(i)} \in G^{(i)} \\ 0 & \text{otherwise} \end{cases}. \tag{20}$$

For the same graph $G^{(i)}$, we define the transition probability matrix $\mathbf{P}^{(i)}$ such that $\mathbf{P}^{(i)} = [p_{jk}^{(i)}]^{n^{(i)}\times n^{(i)}}$, where:

$$p_{jk}^{(i)} = \begin{cases} 1/\deg(a_{jk}^{(i)}) & v_j^{(i)} \sim v_k^{(i)} \in G^{(i)} \\ 0 & \text{otherwise} \end{cases}. \tag{21}$$

Extending and generalizing (20) and (21) to the weighted case is easy if weights are given on edges in the graph. We define

$$a_{jk}^{(i)} = \begin{cases} w(v(i)_j, v(i)_k) & v_j^{(i)} \sim v_k^{(i)} \in G^{(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{22}$$

where $w : E^{(i)} \to \mathbb{R}$ is a weight function that assigns real-valued weights to edges in $G^{(i)}$. Using (22), we define the degree of a node to be in terms of weights associated with edges for which that node is an end-vertex, as

$$\deg^w(v_j^{(i)}) = \sum_k w(v_j^{(i)}, v_k^{(i)}), \quad v_j^{(i)} \sim v_k^{(i)} \in G^{(i)}. \tag{23}$$

Notice that (23) can also be written as $\deg^w(v_j^{(i)}) = \sum_k a_{jk}^{(i)}$—where $a_{jk}^{(i)}$ is defined in (22). Using (23), we can compute $\mathbf{P}^{(i)} = [p_{jk}^{(i)}]$ for weighted graphs, where

$$p_{jk}^{(i)} = \begin{cases} w(v_j^{(i)}, v_k^{(i)})/\deg^w(v_j^{(i)}) & v_k^{(i)} \sim v_k^{(i)} \in G^{(i)} \\ 0 & \text{otherwise} \end{cases}. \tag{24}$$

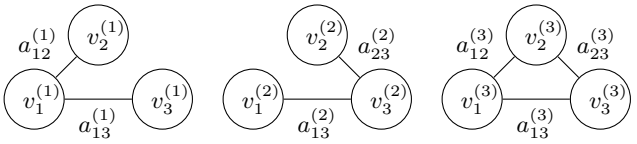


Figure 33: Simple example of dynamic graph. $a_{12}^{(i)} = w(v_1^{(i)}, v_2^{(i)})$ **is as per the definition. One or more of the weights** $a_{12}^{(i)}$ **could be zero.**

In a matrix form, $\mathbf{P}^{(i)}$ can be defined as $\mathbf{P}^{(i)} = (\mathbf{D}^{(i)})^{-1}\mathbf{A}^{(i)}$ where $\mathbf{D}^{(i)}$ is a diagonal matrix computed from $\mathbf{A}^{(i)}$, where the diagonal element $d_{jj}^{(i)}$ in $\mathbf{D}^{(i)}$ is the sum of ones in the j-th row in $\mathbf{A}^{(i)}$ (that is, the degree of node v_{ix} in $G^{(i)}$). At any time slot i, we define the bounding distribution of the Markov chain on the graph G_i as in literature defined $[\deg(v_j^{(i)})/2m^{(i)}]$. It is, however, unclear how to proceed with the different snapshots of the same graphs at different times.

For example, as shown in Figure 33, both nodes $v_1^{(1)}$ and $v_2^{(1)}$ are connected, but not with their future images—$v_1^{(2)}$ or $v_1^{(2)}$ and $v_2^{(2)}$ or $v_2^{(3)}$, respectively. This also applies to states in the future not connected to the past images. In the following, we investigate several techniques for modeling the dynamic social graph as a graph where transitions from future states to past states is possible. Techniques utilize here are generic, and can be used to any graph with multiple labels.

4.2.4 Dynamic graph as a multigraph

As per the formalism in section 4.2.3, a dynamic graph over t time slots can be viewed as a multigraph: all nodes that correspond to a certain past state are collapsed under the same label in the present state. Accordingly, if two nodes that have been labeled with two different labels in the transformation process from multiple graphs to multigraph had an edge between them, the edge is created in the multigraph. Since the multigraph includes states of nodes in the past, current, and future time (for the current snapshot of the graph), multiple edges are potentially created between two labels. Such edges could be weighted or unweighted, depending on the original multiple snapshots representing the dynamic graph.

Formally, for the dynamic graph $G = \{G^{(i)}\}$ described in section 4.2.3, we define a multigraph $\mathbb{G}$ as $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where

$$\mathbb{V} = \bigcup_{i=1\ldots t} \{V^{(i)}\}, \text{ and } \mathbb{E} = \biguplus_{i=1\ldots t} \{E^{(i)}\}. \tag{25}$$

Notice that $\cup$ is a *set union*, which does not allow repetition of vertices, whereas $\uplus$ is a *multiset union*, which allows edge repetition. When $\mathbb{E}$ is computed, the index that corresponds to the time of the edges in E_i can be removed for simplicity. A simple toy example of transforming the multiple snapshots of the dynamic graph in Figure 33 into a multigraph is in Figure 34.

Our formalization above of the graph as a multigraph (rather than union multigraph as per the way defined in [48]) follows the intuition of what a dynamic graph could yield of associations at any time. At a time i, where $1 \leq i \leq t$, constructing the proper graph for operating a potential system, like mixing-based anonymous communication system, and maintaining the same information driven from the original multiple snapshots of the graph would be possible. Furthermore, assigning different weights to different associations, based on the history of the association, would be also possible (see below for details).

It is noted, however, that our model of the graph still uses the high dimension. The literature of graph-theory, however, provides us some powerful tools to reduce the dimensionality of the multi-

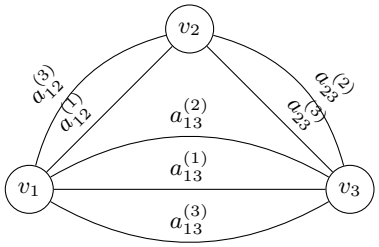


Figure 34: Simple example of converting a dynamic graph (in figure 33) into multigraph by collapsing all images of a node to the node itself (the resulting nodes of the graph are the result of set union) and creating multiple-edges (corresponding to multiset union of the individual graphs).

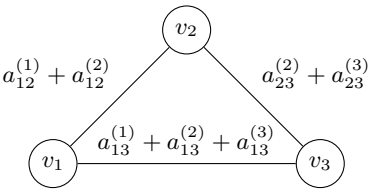


Figure 35: A simple example of multigraph (shown in figure 34) conversion into weighted graph by summing weights of edges between every pair of nodes.

graph above. Indeed, the adjacency matrix representation of multigraph accepted in the literature [16] considers entries in this matrix as the number of edges (or sum of weights) between nodes. Accordingly, reducing the multigraph into a "weighted" graph representation is straightforward.

In the following section, we elaborate on this notation and show a transformation of the multigraph into a weighted graph. Furthermore, we prove some results on the random walk on weighted graphs essential to the rest of this work.

4.2.5 Dynamic graphs as weighted-graphs

Now, it is straightforward to convert the dynamic graph model represented as a multigraph, as in (25), into a weighted graph. For that, we generalize formalizations in section 4.2.3. In particular, the model in (22) can be rewritten (for weighted undirected graph) as $\mathbf{A} = [a_{jk}]^{n\times n}$—here, $n = |\mathbb{V}|$—where

$$a_{jk} = \sum_{i=1\ldots t} w(v_j^{(i)}, v_k^{(i)}), \quad v_j^{(i)} \sim v_k^{(i)} \in G^{(i)} \forall i. \tag{26}$$

Similarly, we extend the model in (23) into

$$\deg^w(v_j) = \sum_{i=1\ldots t} \deg^w(v_j^{(i)}) = \sum_{\forall k} a_k^{(j)} \tag{27}$$

$$= \sum_{\forall k} \sum_{i=1\ldots t} w(v_j^{(i)}, v_k^{(i)}), v_j^{(i)} \sim v_k^{(i)} \in G^{(i)}. \tag{28}$$

We can further extend the transition probability formulation to cover the weighted graph by plugging both (26) and (27) into a similar model to that of (24), to get $\mathbf{P} = [p_{jk}]^{n\times n}$, where

$$p_{jk} = a_{jk} / \deg^w(v_j) \tag{29}$$

For a random walk defined on $\mathbb{G}$ according to the transition probability defined in (29), the following theorem states the stationary distribution. This theorem (and the proof herein) are essential for latter results on characterizing and operating on dynamic graphs.

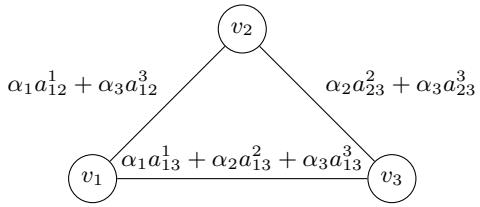


Figure 36: A toy example of the generalized weighted graph model to express dynamic graphs—the same graph in Figure 35 is used.

Also, the proof of Theorem 2 follows similarly as in the proof below.

THEOREM 3. *Let* $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ *be a* connected*,* undirected*, and* weighted *graph defined as in* (25)*. For a random walk following transition probabilities as in* (29)*, the stationary distribution is defined as* $\pi = [\pi_i]^{1\times n}$ *(for* $n = |\mathbb{V}|$*), where:*

$$\pi_i = \deg^w(v_i) / \sum_{k=1\ldots n} \deg^w(v_k) \tag{30}$$

4.2.6 Generalized Weighted Graphs

In many natural social contexts, recent associations are more appreciated than old associations, or vice-versa. By taking the interaction-based model as an example (e.g., [142, 135]), it is noted that the majority of interactions happen in the early time after establishing the social association, where the volume of interaction decays as the social association ages. Accordingly, a general framework for quantifying the potential of any system on top of social networks should consider implicit social network characteristics, such as link age, in addition to the explicit differences among links captured by the topological structure. Here, we generalize the model in section 4.2.5 to accommodate for implicit values of associations over time. Without losing generality, let α_i (for $1 \leq i \leq t$) be a set of parameters that take numerical values. An extension of the social graph model in (26) is as follows:

$$a_{jk} = \sum_{i=1\ldots t} \alpha_i w(v_j^{(i)}, v_k^{(i)}),\ v_j^{(i)} \sim v_k^{(i)} \in G^{(i)} \forall i. \tag{31}$$

The rest of the model in section 4.2.5, particularly in (27) through (31), holds for this generalization after adjusting a_{jk} as in (31). A toy example demonstrating the adjustment of weights in Figure 35 is shown in Figure 36.

4.2.7 Proposed Work

Following our preliminary formalization in above, we look forward to advance this work in many directions as follows.

- We will study how the weighted graph that accounts for dynamics would affect anonymity attained for users by using the new quantification of anonymity and by biasing random walks on the graph according to these weights. We expect these would produce different anonymity guarantees than that obtained when using a static unweighted graph. We will do that on two real-world dynamic graphs (DBLP and Facebook), where dynamics are interactions over time. Our initial measurements show that the anonymity, as both anonymity set and entropy, is improved using our model.
- We will investigate a distributed method for computing the entropy of a walk even before using the social network for real communication so that users can make sure they obtain a high a high anonymity for their communications.

5. REFERENCES

[1] Ebizmba. `www.ebizmba.com/`, 2009.

[2] Facebook. `www.facebook.com`, 2010.

[3] *IEEE 7th International Conference on E-Science, e-Science 2011, Stockholm, Sweden, December 5-8, 2011*. IEEE Computer Society, 2011.

[4] S. M. A. Abbas, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips. A gossip-based distributed social networking system. In *WETICE '09: Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 93–98, Washington, DC, USA, 2009. IEEE Computer Society.

[5] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In *OSDI*, 2002.

[6] Y.-Y. Ahn, S. Han, H. Kwak, S. B. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proc. of WWW*, pages 835–844. ACM, 2007.

[7] E. Al-Shaer, S. Jha, and A. D. Keromytis, editors. *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, 2009.

[8] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[9] F. Azzedin and M. Maheswaran. Evolving and managing trust in grid computing systems. In *IEEE Canadian Conference on Electrical and Computer Engineering*, volume 3, pages 1424–1429. IEEE, 2002.

[10] F. Azzedin and M. Maheswaran. Towards trust-aware resource management in grid computing systems. In *Proc. of CCGRID*, pages 452–452. IEEE, 2002.

[11] L. Banks, P. Bhattacharyya, M. Spear, and S. Wu. Davis social links: Leveraging social networks for future internet communication. In *Applications and the Internet, 2009. SAINT'09. Ninth Annual International Symposium on*, pages 165–168. IEEE, 2009.

[12] L. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[13] V. Batagelj and M. Zaversnik. An $O(m)$ algorithm for cores decomposition of networks. Arxiv preprint cs/0310049, 2003.

[14] G. Bigwood and T. Henderson. Social dtn routing. In *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–2, New York, NY, USA, 2008. ACM.

[15] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 551–560, New York, NY, USA, 2009. ACM.

[16] B. Bollobás. *Modern graph theory*, volume 184. Springer-Verlag, 1998.

[17] N. Borisov. Computational puzzles as sybil defenses. In A. Montresor, A. Wierzbicki, and N. Shahmehri, editors, *Peer-to-Peer Computing*, pages 171–176. IEEE Computer Society, 2006.

[18] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The socialbot network: when bots socialize for fame and

money. In R. H. Zakon, J. P. McDermott, and M. E. Locasto, editors, *ACSAC*, pages 93–102. ACM, 2011.

[19] Z. Cai and C. Jermaine. The latent community model for detecting sybil attacks in social networks. In *Proc. of NDSS*, 2012.

[20] M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman. Exploring mapreduce efficiency with highly-distributed data. In *Proc. of ACM MapReduce*, 2011.

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

[22] J. Caverlee and S. Webb. A large-scale study of MySpace: Observations and implications for online social networks. *Proc. of ICWSM*, 8, 2008.

[23] A. Chaintreau, P. Fraigniaud, and E. Lebhar. Opportunistic spatial gossip over mobile social networks. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 73–78, New York, NY, USA, 2008. ACM.

[24] A. Chakrabarti. *Grid computing security*. Springer Verlag, 2007.

[25] A. Chandra and J. Weissman. Nebulas: Using distributed voluntary resources to build clouds. In *Proc. of HotCloud*, 2010.

[26] V. Chandrasekaran, R. Dantu, N. Gupta, X. Yang, and D. Wijesekera. Efficiency of social connection-based routing in P2P VoIP networks. In *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*, pages 1–6. IEEE, 2010.

[27] K. Chard, K. Bubendorfer, S. Caton, and O. Rana. Social cloud computing: A vision for socially motivated resource sharing. *Services Computing, IEEE Transactions on*, (99):1–1, 2011.

[28] K. Chard, S. Caton, O. Rana, and K. Bubendorfer. Social cloud: Cloud computing in social networks. In *IEEE CLOUD*, pages 99–106. IEEE, 2010.

[29] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 160–168. ACM, 2008.

[30] M. Cryan, M. E. Dyer, L. A. Goldberg, M. Jerrum, and R. A. Martin. Rapidly mixing markov chains for sampling contingency tables with a constant number of rows. *SIAM J. Comput.*, 36(1):247–278, 2006.

[31] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa. From volunteer to cloud computing: cloud@home. In N. M. Amato, H. Franke, and P. H. J. Kelly, editors, *Conf. Computing Frontiers*, pages 103–104. ACM, 2010.

[32] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, New York, NY, USA, 2007. ACM.

[33] E. M. Daly and M. Haahr. Social network analysis for information flow in disconnected delay-tolerant manets. *IEEE Trans. Mob. Comput.*, 8(5):606–621, 2009.

[34] G. Danezis, C. Díaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In *Proc. of PETS*, pages 202–219. Springer, 2010.

[35] G. Danezis and P. Mittal. SybilInfer: Detecting sybil nodes using social networks. In *Proc. of NDSS*, 2009.

[36] J. Davitz, J. Yu, S. Basu, D. Gutelius, and A. Harris. ilink: search and routing in social networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 931–940, New York, NY, USA, 2007. ACM.

[37] J. Dean and S. Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[38] M. Dellamico, , and Y. Roudier. A measurement of mixing time in social networks. In *STM 2009: 5th International Workshop on Security and Trust Management*, 2009.

[39] R. Dingledine and N. Mathewson. Tor bridges specification. Technical report, The Tor Project, `https://svn.torproject.org/svn/tor/trunk/doc/spec/bridgesspec.txt`, 2008.

[40] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *USENIX Security*, 2004.

[41] T. T. A. Dinh and M. Ryan. A sybil-resilient reputation metric for p2p applications. In *SAINT*, pages 193–196, 2008.

[42] P. Domingues, B. Sousa, and L. Moura Silva. Sabotage-tolerance and trust management in desktop grid computing. *Future Generation Computer Systems*, 23(7):904–912, 2007.

[43] J. Douceur and J. S. Donath. The sybil attack. In *IPTPS*, pages 251–260, 2002.

[44] C. Dwyer, S. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. In *Proceedings of AMCIS*, 2007.

[45] P. Erdős and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Hungarica*, 17(1):61–99, 1966.

[46] Y. Fernandess and D. Malkhi. On spreading recommendations via social gossip. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 91–97, New York, NY, USA, 2008. ACM.

[47] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, March 1977.

[48] M. Gjoka, C. T. Butts, M. Kurant, and A. Markopoulou. Multigraph sampling of online social networks. *JSAC*, 2011.

[49] J. Golbeck. The dynamics of web-based social networks: Membership, relationships, and change. *First Monday*, 12(11):1–17, 2007.

[50] J. Golbeck and J. Hendler. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, volume 42, pages 43–44. IEEE, 2006.

[51] J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland at College Park, College Park, MD, USA, 2005. AAI3178583.

[52] M. Gurevich and I. Keidar. Correctness of gossip-based membership under message loss. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 151–160. ACM, 2009.

[53] M. Gurevich and I. Keidar. Correctness of gossip-based membership under message loss. *SIAM J. Comput*, 39:3830–3859, 2011.

[54] N. Hopper, E. Vasserman, and E. Chan-Tin. How much

anonymity does network latency leak? *ACM TISSEC*, 2010.

[55] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu. Reverse social engineering attacks in online social networks. In T. Holz and H. Bos, editors, *DIMVA*, volume 6739 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2011.

[56] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving p2p data sharing with oneswarm. In *Proc. of ACM SIGCOMM*, pages 111–122, 2010.

[57] R. Jansen, N. Hopper, and Y. Kim. Recruiting new tor relays with braids. In *ACM CCS*, 2010.

[58] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of the permanent resolved (preliminary version). In *STOC*, pages 235–244. ACM, 1988.

[59] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. of WWW*, pages 640–651. ACM, 2003.

[60] R. Kannan, L. Lovász, and R. Montenegro. Blocking conductance and mixing in random walks. *Comb. Probab. Comput.*, 15:541–570, July 2006.

[61] H. A. Kautz, B. Selman, and M. A. Shah. Referral web: Combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, 1997.

[62] J. Koshy, K. Bubendorfer, and K. Chard. A social cloud for public eresearch. In *eScience* [3], pages 363–370.

[63] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. *Link Mining: Models, Algorithms, and Applications*, pages 337–357, 2010.

[64] J. Ledlie and M. I. Seltzer. Distributed, secure load balancing with skew, heterogeneity and churn. In *INFOCOM*, pages 1419–1430, 2005.

[65] K. Lerman. Social networks and social information filtering on digg. *Arxiv preprint cs/0612046*, 2006.

[66] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg. Predicting positive and negative links in online social networks. In *Proc. of WWW*. ACM, 2010.

[67] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, New York, NY, USA, 2005. ACM.

[68] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *CoRR*, abs/0810.1355, 2008.

[69] C. Lesniewski-Laas. A Sybil-proof one-hop DHT. In *Proceedings of the 1st workshop on Social network systems*, pages 19–24. ACM, 2008.

[70] C. Lesniewski-Laas. *Design and Applications of a Secure and Decentralized Distributed Hash Table*. PhD thesis, Massachusetts Institute of Technology, 2010.

[71] C. Lesniewski-Lass and M. F. Kaashoek. Whānau: A sybil-proof distributed hash table. In *7th USENIX Symposium on Network Design and Implementation*, pages 3–17, 2010.

[72] F. Lesueur, L. Mé, and V. V. T. Tong. A sybil-resistant admission control coupling sybilguard with distributed certification. In *WETICE*, pages 105–110, 2008.

[73] M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval*, pages 481–486. Springer, 2009.

[74] D. Liben-Nowell and J. M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559. ACM, 2003.

[75] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623, 2005.

[76] D. Lick and A. White. k-Degenerate graphs. *Canadian J. of Mathematics*, 22:1082–1096, 1970.

[77] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang. Moon: Mapreduce on opportunistic environments. In *Proc. of HPDC*, pages 95–106, New York, NY, USA, 2010. ACM.

[78] W. Lin, H. Zhao, and K. Liu. Incentive cooperation strategies for peer-to-peer live multimedia streaming social networks. *IEEE Transactions on Multimedia*, 11(3):396–412, 2009.

[79] M. Litzkow, M. Livny, and M. Mutka. Condor-a hunter of idle workstations. In *Proc. of ICDCS*, pages 104–111. IEEE, 1988.

[80] I. Mabrouki, X. Lagrange, and G. Froc. Random walk based routing protocol for wireless sensor networks. In *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

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

[82] S. Marti, P. Ganesan, and H. Garcia-Molina. Dht routing using social links. In G. M. Voelker and S. Shenker, editors, *IPTPS*, volume 3279 of *Lecture Notes in Computer Science*, pages 100–111. Springer, 2004.

[83] S. Marti, P. Ganesan, and H. Garcia-Molina. Sprout: P2p routing with social networks. In *Proc. of EDBT Workshops*, pages 425–435. ACM, 2004.

[84] D. McDonald. Recommending collaboration with social networks: a comparative evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 593–600. ACM, 2003.

[85] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with torsk. In Al-Shaer et al. [7], pages 590–599.

[86] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting Social Networks for Internet Search. In *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06)*, Irvine, CA, November 2006.

[87] A. Mislove, M. Marcon, P. K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Internet Measurement Comference*, pages 29–42, 2007.

[88] A. Mislove, A. Post, P. Druschel, and P. K. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In J. Crowcroft and M. Dahlin, editors, *NSDI*, pages 15–30. USENIX Association, 2008.

[89] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in Online Social Networks. In *WSDM*, New York, NY, February

2010.

[90] P. Mittal, N. Borisov, C. Troncoso, A. Rial, and I. Leuven. Scalable Anonymous Communication with Provable Security. In *USENIX HotSec*, 2010.

[91] P. Mittal, M. Caesar, and N. Borisov. X-Vine: Secure and pseudonymous routing using social networks. In *Proc. of NDSS*, 2012.

[92] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg. Pir-tor: Scalable anonymous communication using private information retrieval. In *USENIX Security*, 2011.

[93] A. Mohaisen, T. AbuHmed, H. Kang, Y. Kim, and D. Nyang. Mistaking friends for foes: An analysis of a social network-based Sybil defense in mobile networks. In *Proceeding of the 5th ACM ICUIMC*. ACM, 2011.

[94] A. Mohaisen, T. AbuHmed, T. Zhu, and M. Mohaisen. Collaboration in social network-based information dissemination. In *The 28th IEEE International Conference on Communications (IEEE ICC)*, 2012.

[95] A. Mohaisen, N. Hopper, and Y. Kim. Designs to account for trust in social network-based sybil defenses. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 714–716. ACM, 2010.

[96] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. Technical report, University of Minnesota, 2010.

[97] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. In *INFOCOM'11: Proceedings of the 30th conference on Information communications*, pages 336–340, Piscataway, NJ, USA, 2011. IEEE Press.

[98] A. Mohaisen, N. Hopper, and Y. Kim. On the mixing of directed social graphs. Working title, 2012.

[99] A. Mohaisen, P. Luo, Z.-L. Zhang, and Y. Kim. How much is too much: Measuring bias in social networks mixing time due to sampling. Working title, 2012.

[100] A. Mohaisen, H. Tran, , A. Chandra, and Y. Kim. SocialCloud: building distributed computing services on top of social networks. Submitted to IEEE ICDCS, 2012.

[101] A. Mohaisen, H. tran, A. Chandra, and Y. Kim. Socialcloud: Using social networks to build distributed computing services. Technical report, University of Minnesota, 2011.

[102] A. Mohaisen, H. Tran, A. Chandra, and Y. Kim. SocialCloud. `http://socialcloud.cypriv.com`, July 2011S.

[103] A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. Understanding social networks properties for trustworthy computing. In *Proc. of IEEE ICDCS Workshops*, 2011.

[104] A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. Understanding the mixing patterns of social networks. Technical report, UMN, 2011.

[105] A. Mohaisen, H. Tran, and Y. Kim. Dynamix: dynamic social structures for anonymity. Working title, 2012.

[106] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 383–389, New York, NY, USA, 2010. ACM.

[107] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. Technical report, University of Minnesota, 2010.

[108] A. Nachmias and A. Shapira. Testing the expansion of a graph. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(118), 2007.

[109] S. Nagaraja. Anonymity in the wild: Mixes on unstructured networks. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2007.

[110] M. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[111] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577, 2006.

[112] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):26113, 2004.

[113] M. Ohira, N. Ohsugi, T. Ohoka, and K. Matsumoto. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In *Proceedings of the 2005 international workshop on Mining software repositories*, pages 1–5. ACM, 2005.

[114] K. Puttaswamy, A. Sala, and B. Zhao. Improving anonymity using social links. In *Secure Network Protocols, 2008. NPSec 2008. 4th Workshop on*, pages 15–20. IEEE, 2008.

[115] D. Quercia and S. Hailes. Sybil attacks against mobile users: friends and foes to the rescue. In *INFOCOM'10: Proceedings of the 29th conference on Information communications*, pages 336–340, Piscataway, NJ, USA, 2010. IEEE Press.

[116] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of ACM CSCW*, pages 175–186, 1994.

[117] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *ISWC*, LNCS, pages 351–368. Springer, 2003.

[118] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. In *Proc. of WWW*, pages 861–870. ACM, 2010.

[119] O. Sandberg. Distributed routing in small-world networks. In *Proceedings of the eighth Workshop on Algorithm Engineering and Experiments and the third Workshop on Analytic Algorithmics and Combinatorics*, page 144. Society for Industrial Mathematics, 2006.

[120] M. Sato. Creating next generation cloud computing based network services and the contributions of social cloud operation support system (oss) to society. In *Proc. of IEEE WETICE*, pages 52–56, 2009.

[121] M. Schuchard, A. Dean, V. Heorhiadi, N. Hopper, and Y. Kim. Balancing the shadows. In *WPES*. ACM, 2010.

[122] B. Sieka. Using radio device fingerprinting for the detection of impersonation and sybil attacks in wireless networks. In *ESAS*, pages 179–192, 2006.

[123] A. Sinclair. Improved bounds for mixing rates of marcov chains and multicommodity flow. *Comb., Probability & Computing*, 1:351–370, 1992.

[124] S. Song, K. Hwang, and Y. Kwok. Trusted grid computing with security binding and trust integration. *Grid computing*, 3(1), 2005.

[125] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proceeding*

of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 677–685. ACM, 2008.

[126] A. M. Thaufeeg, K. Bubendorfer, and K. Chard. Collaborative eresearch in a social cloud. In *eScience* [3], pages 224–231.

[127] A. Tran, N. Hopper, and Y. Kim. Hashing it out in public: common failure modes of dht-based anonymity schemes. In *WPES*, 2009.

[128] D. Tran, F. Chiang, and J. Li. Friendstore: cooperative online backup using trusted nodes. In *Proc. of SNS*, pages 37–42, 2008.

[129] N. Tran, J. Li, L. Subramanian, and S. Chow. Improving social-network-based sybil-resilient node admission control. In *INFOCOM'11: Proceedings of the 30th conference on Information communications*, pages 336–340, Piscataway, NJ, USA, 2011. IEEE Press.

[130] N. Tran, J. Li, L. Subramanian, and S. S. Chow. Optimal sybil-resilient node admission control. In *Proc. of INFOCOM*. IEEE, 2011.

[131] N. Tran, J. Li, L. Subramanian, and S. S. M. Chow. Brief announcement: improving social-network-based sybil-resilient node admission control. In A. W. Richa and R. Guerraoui, editors, *PODC*, pages 241–242. ACM, 2010.

[132] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association.

[133] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim. Membership-concealing overlay networks. In *ACM CCS*, 2009.

[134] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[135] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks*, August 2009.

[136] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *Proc. of SIGCOMM*. ACM, 2010.

[137] A. Vora, M. Nesterenko, S. Tixeuil, and S. Delaët. Universe detectors for sybil defense in ad hoc wireless networks. In *SSS*, LNCS, pages 63–78. Springer, 2008.

[138] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl. Scientific cloud computing: Early definition and experience. In *Proc. of IEEE HPCC*, pages 825–830. Ieee, 2008.

[139] W. Wang and A. Lu. Visualization assisted detection of sybil attacks in wireless networks. In *VizSEC*, pages 51–60, 2006.

[140] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge Univ Pr, 1994.

[141] J. B. Weissman, P. Sundarrajan, A. Gupta, M. Ryden, R. Nair, and A. Chandra. Early experience with the distributed nebula cloud. In *Proc. of ACM DIDC*, pages 17–26, 2011.

[142] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, pages 205–218, New York, NY, USA, 2009. ACM.

[143] S. Xu, X. Li, and P. Parker. Exploiting social networks for threshold signing: attack-resilience vs. availability. In M. Abe and V. D. Gligor, editors, *ASIACCS*, pages 325–336. ACM, 2008.

[144] H. Yu. Sybil defenses via social networks: a tutorial and survey. *ACM SIGACT News Distributed Computing Column*, 2011.

[145] H. Yu, P. B. Gibbons, and M. Kaminsky. Toward an optimal social network defense against sybil attacks. In *PODC*, pages 376–377. ACM, 2007.

[146] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.

[147] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: defending against sybil attacks via social networks. In *Proc. of SIGCOMM*, pages 267–278. ACM, 2006.

[148] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman. SybilGuard: defending against sybil attacks via social networks. *IEEE/ACM Trans. Netw.*, 16(3):576–589, 2008.