SmartWalk: Enhancing Social Network Security via Adaptive Random Walks

Yushan Liu
Princeton University
yushan@princeton.edu

Shouling Ji
Zhejiang University / Georgia Tech
sji@gatech.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

ABSTRACT

Random walks form a critical foundation in many social network based security systems and applications. Currently, the design of such social security mechanisms is limited to the classical paradigm of using fixed-length random walks for all nodes on a social graph. However, the fixed-length walk paradigm induces a poor trade-off between security and other desirable properties.

In this paper, we propose SmartWalk, a security enhancing system which incorporates *adaptive random walks* in social network security applications. We utilize a set of supervised machine learning techniques to predict the necessary random walk length based on the structural characteristics of a social graph. Using experiments on multiple real world topologies, we show that the desired walk length starting from a specific node can be well predicted given the local features of the node, and limited knowledge for a small set of training nodes. We describe node-adaptive and path-adaptive random walk usage models, where the walk length adaptively changes based on the starting node and the intermediate nodes on the path, respectively. We experimentally demonstrate the applicability of adaptive random walks on a number of social network based security and privacy systems, including Sybil defenses, anonymous communication and link privacy preserving systems, and show up to two orders of magnitude improvement in performance.

1. INTRODUCTION

Random walks in security applications. Nowadays, many applications leverage the trust relationships in social networks to improve their system security and privacy, such as Sybil defenses [51, 52, 35, 24, 13, 44, 43], anonymous communication [37, 34, 12, 15], secure routing [31, 24, 32], censorship resilience [41, 1, 2] and secure reputation systems [44, 17]. An important approach used in the design of these systems is to perform random walks on social networks. Random walk is a random sequence of nodes where successive nodes are neighbors. Many algorithms based on random walks are gaining increasing popularity since they are simple to implement and can be used in both centralized and distributed systems to probe structural properties of the whole network [30, 40, 48, 18, 39, 28]. For instance, Pons *et al.* in [40] proposed a random walk based algorithm to capture the community structure in large networks. Since random walks can help sample and obtain some structural information of a social network, they play a crucial role in many social network based security systems. For instance, Danezis *et al.* and others [12, 37, 34] proposed decentralized protocols for anonymous communications that leverage users' social links and use random walks to build circuits for onion routing. Mittal *et al.* in [33] perturbed a social graph by replacing real edges with edges between starting nodes and terminal nodes of random walks in order to provide link privacy. Yu *et al.* proposed SybilGuard [52] and SybilLimit [51], two Sybil defense protocols that perform on random walk based routes and register public keys with the tails to differentiate Sybil users from benign users.

Designers' Dilemma: Security vs. Performance. One important and interesting parameter of random walks is its length. The choice of random walk length is closely related to structural properties of networks and has a significant impact on the trade-off between system security and system performance/utility. In existing security mechanisms, only the classical paradigm of *fixed length random walks* for all nodes is considered, i.e., random walks starting from each node in the network use the same fixed length. Since random walk length greatly influences both the security/privacy and the application utility, the lack of flexibility in the fixed length random walk paradigm can leave the system design in a dilemma. For example, in SybilLimit [51], unnecessarily long random walks give adversaries more power to corrupt the region formed by honest users. However, a small random walk length leads to a high false positive rate, i.e., a high percentage of misclassified benign users. For graph privacy [33], strong link privacy relies on deep perturbation to the original graph, indicating a large random walk length. However, as the fixed random walk length increases, the perturbed graph gradually approaches to a random graph, incurring a significant loss of utility. In social network based anonymous systems [34], the expected anonymity is a function of the random walk length and longer random walks enhance the anonymity at the cost of extra latency. These challenges are difficult to address in the case of fixed length random walks.

Contributions. To address the above challenges, we develop SmartWalk, which introduces the concept of *adaptive*

CCS'16, October 24 - 28, 2016, Vienna, Austria

ISBN 978-1-4503-4139-4/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2976749.2978319

random walks across nodes in the network; our approach uses heterogeneous walk lengths across nodes in the network to enhance the trade-off between system security and performance.

1. We introduce the concept of *local mixing time*, which measures the minimum random walk length for a certain starting node to be within a given distance [1] to stationarity (see Section 3 for formal definitions). We show that in real-world social graphs, the local mixing time across nodes exhibits a heterogeneous and long-tail distribution.

2. We present a local mixing time prediction algorithm, which employs supervised machine-learning methods to *effectively* predict the local mixing time for a node according to its local topological features and *limited* global knowledge of the graph (directly computing the local mixing time requires the knowledge of the entire social graph and can be expensive). Our prediction performance is evaluated using Facebook friendship and interaction graphs and a Twitter graph. We show that with a small subset of training samples (around 1% of all nodes) and local characteristics (about 3-hop neighbourhood), we can get satisfiable prediction of a node's local mixing time.

3. Compared to conventional security mechanisms, which leverage the same walk length for all nodes, we propose two novel algorithms which produce adaptive random walks based on the underlying heterogeneity of the local mixing time in social networks. The *node-adaptive* algorithm determines the length of random walks by the starting node. The *path-adaptive* algorithm automatically adjusts the remaining random walk length according to the intermediate nodes along the walk path.

4. We test the applicability of these two algorithms in a set of random walk based security and privacy applications including *Sybil defense*, *anonymous communication* and *link privacy* preserving systems. Using real-world social network topologies, we show that both the node-adaptive and path-adaptive algorithms significantly outperform the existing fixed length algorithms for any given expected random walk length. *The improvement can be up to two orders of magnitude.* By properly adjusting walk lengths to nodes and paths, our algorithms are able to offer fine-grained control over the trade-off between security/privacy and other metrics for these systems.

To the best of our knowledge, all the existing random walk-based security and privacy mechanisms [29, 3, 14, 11, 36, 6, 4] use a uniform walk length, while our work is the first to adapt the random walk length depending on the structural characteristics of nodes. Looking ahead, our approach has broad potential to impact security-performance trade-offs in applications even outside the context of social networks; this includes graph-theoretic detection mechanisms for P2P botnets, spamming botnets, malicious online communities, and malware [38, 42, 22, 53].

2. SYSTEM OVERVIEW

Motivating Applications. In this paper, our objective is to enhance the security of social network based systems by leveraging adaptive random walks. Existing security systems [12, 37, 34, 33, 52, 51] all adopt a fixed-length random walk scheme, where the random walk length is set as the same sufficiently large value (typically the mixing time of the entire graph [25, 36]) for all walks to meet some system requirements. However, the downside of the fixed-length based schemes is that a poor security-utility trade-off is induced. Our key insight is that for large-scale social graphs, the required random walk length to achieve a certain distance to stationarity has a *heterogeneous and long-tail distribution over different starting nodes*. By predicting an adaptive walk length for different nodes, we are able to achieve a better trade-off between the security and other properties of social network based systems.

We mainly consider three social network based security and privacy systems, including Sybil defense, anonymous communication and link privacy preserving systems.

a) Sybil defenses. A Sybil attack is an attack wherein a malicious user subverts the system by forging multiple distinct identities. With a large number of fake identities inserted by malicious users, the security of the system can be severely sabotaged. To defend against Sybil attacks, many defense mechanisms have been proposed by leveraging the trust relationships in social networks [51, 52, 35, 24, 13, 44, 43, 5, 26]. For instance, SybilLimit [51] is a Sybil defense protocol that performs random walk based routes on social graphs and examines some conditions to detect Sybil users. In SybilLimit, random walk is set as the mixing time [51] to ensure that most benign users can be correctly verified, i.e., low false positives.

b) Anonymous communication. Anonymous communication systems such as Tor preserve user privacy by obfuscating the correspondence between the user and the destination communicating entity. Many anonymous system designs [37, 34, 12, 15] that have been proposed in recent years are built upon leveraging users' trusted relationships, and a typical one of them is the Pisces protocol [34]. Similar to the Tor protocol, Pisces [34] provides low-latency anonymous communication by proxy servers and onion routing. Random walks are performed on a social graph as the relay selection method to create onion routing paths. An important metric to quantify the level of provided anonymity is the Shannon entropy, which can be significantly influenced by the length of random walks.

c) Link privacy. Social trust has been playing a crucial role in various applications in many fields. To avoid revealing the sensitive information about users' social relationships, link privacy preserving systems provide a delicately perturbed social graph to these applications by adding extra noise to the local structure of a social network. Mittal *et al.* in [33] protected link privacy by replacing a real link between two users with a fake link generated by a random walk. The noise introduced to the graph increases as the random walk length gets larger, ensuring stronger privacy.

Pitfalls of fixed-length random walks. To achieve security/privacy guarantees, the length of random walks in the above systems is required to be sufficiently large. However, long random walks enhance the security at the cost of sacrificing other desirable properties of these applications. For SybilLimit, the maximum number of Sybil users that can be possibly misclassified as honest users increases proportionally to the random walk length, indicating that longer random walks result in a growing false negative rate. For Pisces, extra latency is incurred for performing long random walks, which degrades the performance of anonomous communication. For a link privacy preserving system, the length

[1]The distance to stationarity is a tunable parameter to satisfy different application requirements.

of random walks reflects the degree of introduced randomness to perturb the original graph. As the walk length increases, the perturbed graph gets closer to a random graph, resulting in the failure of utility guarantees.

For a fixed-length random walk scheme, the length is typically set as the graph mixing time, which is the minimum length for walks from every possible starting node to approach the stationarity. However, Mohaisen *et al.* showed that the mixing time of social graphs is much larger than anticipated [36], which implies that setting the walk length globally as the same mixing time actually induces weaker utility guarantees or less efficiency in these systems. Hence, we develop SmartWalk to avoid unnecessarily long random walks by properly adjusting the walk length for each node. We demonstrate the applicability of SmartWalk in the above three security systems by showing that up to two orders of magnitude performance improvement can be achieved.

Formalizing local mixing time. We define the local mixing time as a measure of the random walk length for a specific node to achieve a certain distance to stationarity. Using real-world large-scale social graphs, we observe that it only takes a few hops for random walks starting from a majority of nodes to approach the stationary distribution, whereas there also exists a small group of nodes with a longer local mixing time. This is due to the fact that random walk length is closely related to the community structure within a social network. Most of the communities are well-connected to each other in a social network, resulting in a mostly homogeneous random walk length. However, the existence of some small communities that are poorly connected to the rest of the network can greatly prolong the time to approach stationarity. This heterogeneity property is not fully utilized in the design of current social network based security systems. Hence, by adaptively choosing the walk length for different nodes, we are able to significantly reduce the number of uncessarily long random walks and improve system security.

System architecture. Fig. 1 shows the overall architecture of the SmartWalk system, which reads a social graph and produces adaptive random walks upon requests from social network based security systems. Specifically, the SmartWalk system consists of two components: the local mixing time prediction algorithm and the random walk usage model. Given a node index, the prediction algorithm employs supervised learning techniques to predict the local mixing time of the given node based on its local features and some limited global knowledge of the graph. The random walk usage model is responsible for generating adaptive random walks, which are later leveraged in the security systems, based on the results from the prediction algorithm.

The usage model implements either the node-adaptive algorithm or the path-adaptive algorithm. In other words, we can simply make the walk length specific to a starting node, or alternatively, modify the walk length every time a follow-up hop is taken as the path is extended. The node-adaptive scheme only needs the predicted length of the starting node, but fail to take into account the structrual features of the intermidate nodes. The path-adaptive scheme significantly reduces unnecessarily long walks by dynamically updating the number of remaining hops every time the walk reaches a new node, but consequently requires more prediction inputs.

The SmartWalk system substitutes the fixed-length random walks in the security applications with adaptive random walks to achieve fine-grained control over the trade-off between security/privacy and other metrics.

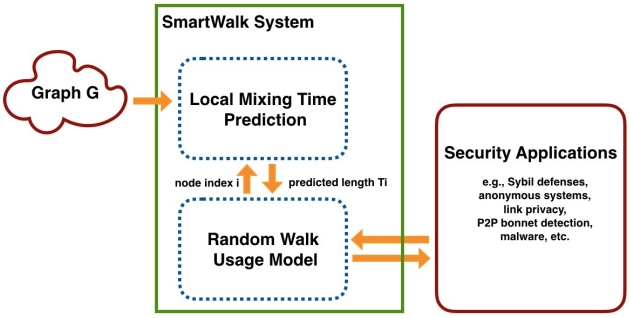


Figure 1: SmartWalk system architecture.

3. LOCAL MIXING TIME

Network model. We model a social network by a finite undirected connected graph G with node set V and edge set E [2]. In pratice, the nodes in V can be the users of a Facebook social graph, and the edges in E can be the frienship relationships between Facebook users represented by the endpoints of these edges. The size of the graph G is $n = |V|$ and the number of edges in G is $m = |E|$.

Consider a random walk of length k on G: it starts from node v_0 [3], and if it is at some node v_t at the t-th hop, the probability of moving to each neighbor of v_t is $1/deg(v_t)$, where $deg(\cdot)$ is the node degree. After k hops, it arrives at the terminal node v_k. The sequence of random walk nodes $\{v_t\}_{t=0}^{k}$ forms a Markov chain with a transition probability matrix $P = [p_{ij}]$, where $i, j \in V$ and the $(i,j)^{th}$ entry in P is given by

$$p_{ij} = \begin{cases} \frac{1}{deg(i)}, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Denote the probability distribution of the t-th node with the starting node i by $\pi_i(t)$, which is a row vector in $\mathbb{R}^n$. The random walk is thus characterized by $\pi_i(t+1) = \pi_i(t) \cdot P$. It follows that $\pi_i(t) = \pi_i(0) \cdot P^t$. For a random walk of length k starting from node i, it finally reaches the distribution $\pi_i(k)$. For irreducible and aperiodic graphs, the corresponding Markov chain is *ergodic*. In this case, for any starting node i, as walk length $k \to \infty$, the distribution $\pi_i(k)$ converges to a unique stationary distribution π, which satisfies $\pi = \pi P$.

For undirected and connected graphs, it has been proven that the distribution $[\frac{deg(i)}{2m}]_{i=1}^{n}$ satisfies $\pi = \pi P$, and is the unique stationary distribution of random walks [25].

Local mixing time. Below we introduce a new concept *local mixing time* (parameterized by ϵ) which measures the minimal length required for a random walk starting from node i to be within an ϵ-distance to the stationarity.

Definition 1 (Local Mixing Time). *The local mixing time (parameterized by ϵ) of a Markov chain with an initial distribution concentrated at node i is defined as*

$$T_i(\epsilon) = \min\{t : |\pi - \pi_i(0)P^t|_1 < \epsilon\},$$

[2]We presented our analysis in terms of undirected graphs for illustration simplicity.

[3]We use v_t to denote the t-th node on a random walk, which can take a value from $\{0, 1, ..., n-1\}$.

where π is the stationary distribution, $\pi_i(0)$ is the initial distribution concentrated at node i, P is the transition matrix, t is a non-negative integer, $\epsilon > 0$ is an arbitrary small constant value, and $|\cdot|_1$ is the total variation distance. [4]

ϵ is an application-specific parameter, and smaller ϵ indicates stronger convergence. Then the conventional mixing time $T(\epsilon)$ of graph G can be obtained by finding the maximum local mixing time over all nodes in G, i.e., $T(\epsilon) = \max\{T_i(\epsilon), \ \ i \in V\}$ [36]. Note that even for a single node, the computation of its local mixing time requires the knowledge of the entire social graph.

Given a node i, denote the set of its neighboring nodes as $N(i)$. It can be proven that the local mixing time of node i atisfies $T_i(\epsilon) \leq \max\{T_j(\epsilon), j \in N(i)\} + 1$. The proof is deferred in Appendix A.

Evaluation: local mixing time in social graphs. We use 10 various large-scale real-world social network topologies that mainly come from the Stanford Large Network Dataset Collection [23] and other sources [45] to evaluate the local mixing time for nodes in social graphs. The used datasets are listed in Table 1. Since the local mixing time is defined for undirected and connected graphs, we first convert the directed graphs to undirected by preserving only bidirectional edges [5]. Then, we extract the largest connected component from each graph to compute the local mixing time. In Table 1 we show some statistics of the social datasets [6] including the numbers of nodes and edges, average node degree $\bar{d}$ and the average clustering coefficient $\bar{cc}$.

Fig. 2a depicts the Cumulative Distribution Function (CDF) of local mixing time for every node in the Facebook1, Facebook2, Twitter, Epinions and DBLP datasets, with parameter $\epsilon = 0.25$ [7]. The local mixing time is computed for every node in a social dataset, using brute force. From Fig. 2a, we can see that over 80% Facebook1 nodes have a small local mixing time (less than 20), whereas the rest nodes (less than 20%) have a larger local mixing time (with the maximum value around 90). For Twitter, over 80% of its nodes have a local mixing time below 100 while the rest nodes may reach 400. Similar results are observed for other three datasets. Based on Fig. 2a, we conclude that the distribution of local mixing time over nodes has a heterogeneous and long-tail property.

Fig. 2b illustrates the CDF of the local mixing time for a random sample of 5% nodes in each of the ten datasets in Table 1. For Facebook1, Facebook2, Twitter, Epinions and DBLP, they have similar results in Fig. 2a and Fig. 2b, which implies that a random sample of 5% nodes are sufficient to demonstrate the basic property of the local mixing time CDF for all nodes. Hence, the heterogeneous and long-tail property exists for all the ten datasets – a large portion of their nodes have much faster local mixing time than the rest, which implies that the required length of random walks to approach the stationarity is heterogeneous across nodes. Even though a small number of nodes may need a long random walk, most nodes only need a relatively small random walk length to reach the stationary distribution. However, the classical paradigm only considers random walks of a fixed length for all nodes. As a result, every node performs long random walks to meet the application requirement, which is in fact unnecessary and inefficient for a majority of nodes based on our observation. Furthermore, the mixing time of the datasets in Table 1 is generally greater than $O(\log n)$, validating the observation of Mohaisen *et al.* [36]: the worst-case mixing time of real-world social networks is much larger than expected and being used in literature. Hence, we are interested in predicting the local mixing time for any given node in social graphs and proposing usage models of adaptive random walks to utilize the heterogeneity.

Table 1: Datasets and their properties

Datasets	Nodes	Edges	$\bar{d}$	$\bar{cc}$
Facebook1[45]	63,392	816,886	25.8	0.22
Facebook2[45]	43,953	182,384	8.5	0.11
Google+[23]	107,614	12,238,285	227.4	0.5
Twitter[23]	81,306	1,342,296	33.0	0.6
Epinions[23]	75,877	405,739	10.7	0.14
LiveJournal[23]	4,843,953	42,845,684	17.8	0.27
Pokec[23]	1,632,803	22,301,964	27.3	0.11
DBLP[23]	317,080	1,049,866	6.6	0.63
Youtube[23]	1,134,890	2,987,624	5.3	0.08
Flickr[23]	80,513	5,899,882	146.6	0.16

4. SMARTWALK: PREDICTION

In this section, we apply a set of supervised machine learning techniques to predict the local mixing time for any given node in a social graph.

Goals. Since it requires the global information of a social network to compute the exact local mixing time for a given node (recall Definition 1), it is infeasible for decentralized social network based systems such as SybilLimit [51] to get the metric directly (for all nodes). Even for the case with global information accessible, it is computationally expensive and time consuming for large-scale networks to calculate the exact local mixing time (with the complexity of $O(n^{2.3729}T(\epsilon))$) [46]. Therefore, we are interested in proposing a fast and distributed algorithm to estimate the local mixing time given a specific node. Based on supervised machine learning techniques, our algorithm only leverages local characteristics of nodes and limited knowledge of the local mixing time for a small subset of nodes. Specifically, we first compute the local mixing time for a small subset of nodes and use it as training labels to fit a regression model. Then, the model is used to predict the local mixing time for any given node in a social graph.

Features & Prediction. Given node i, we aim to predict its local mixing time only based on the local characteristics within its k-hop neighborhood (k is relatively small compared to $T(\epsilon)$). We show the prediction algorithm in Algorithm 1. In Algorithm 1, we choose the probability distribution of a k-hop random walk from node i, denoted by $\pi_i(k)$, as the features. The intuition is that the random walk length of a node is mainly determined by its neighbors, and

[4]The total variation distance between two probability measures θ_1 and θ_2 on a sigma-algebra $\mathcal{F}$ of subsets of the sample space Ω is defined via $|\theta_1, \theta_2)|_1 = \sup_{A \in \mathcal{F}} |\theta_1(A) - \theta_2(A)|$ [10].

[5]In most security and privacy systems that leverage social trust, bidirectional links between users are considered as an indicator for stronger trust than unidirectional ones.

[6]Facebook1 is a Facebook friendship (user-to-user link) network at the New Orleans area and Facebook2 is a Facebook interaction (wall post) network at the New Orleans area.

[7]The variation distance parameter ϵ is typically set as 0.25 [25]. Setting ϵ to be other values gives similar results.

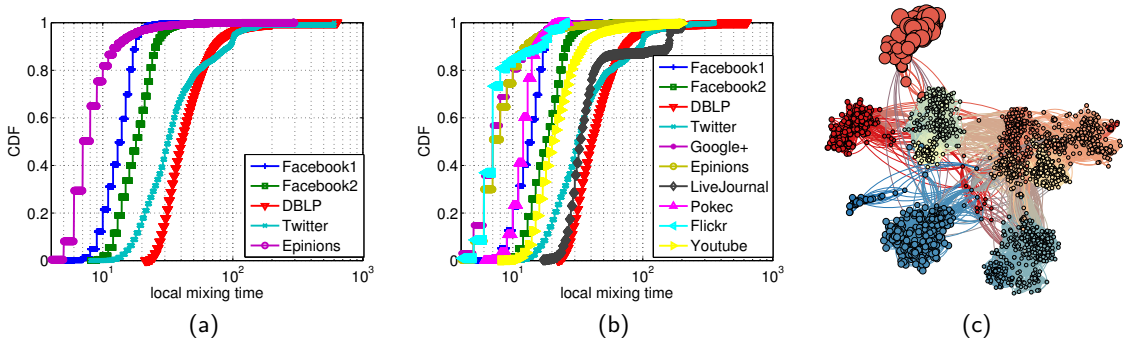


Figure 2: (a) The CDF of local mixing time for every node in Facebook1, Facebook2, Twitter, Epinions and DBLP (x-axis is in the logarithmic scale). (b) The CDF of local mixing time for a sample of 5% nodes in each of the ten datasets in Table 1 (x-axis is in the logarithmic scale). (c) Illustration of the relationship between the community structure and local mixing time using the Facebook1 graph.

hence the probabilities over neighbors could be used as features for estimation (see Remark 1). In centralized systems, $\pi_i(k)$ can be obtained by $\pi_i(k) = \pi_i(0) \cdot P^k$. In distributed systems, $\pi_i(k)$ can be approximated by the terminal node distribution after performing a sufficiently large number of k-hop random walks starting from node i. Each node uses its local mixing time as the label.

We randomly select a subset of M nodes as the training set (M is relatively small compared to n). After collecting the training feature matrix and training labels, we use them to fit a Random Forest regression model [7]. We also compare the results with those under a k-Nearest Neighbors (KNN) regression model [9]. Random Forest fits a number of decision trees on sub-samples of the dataset and then averages the obtained labels. KNN finds a set of the closest training samples to the target point and predicts the label by assigning weights to the set's labels. Then we predict the local mixing time for the target node using structural features associated with the node. In the evaluation part, we show that when k is carefully chosen, we can obtain a good estimation of the local mixing time only using characteristics within the local neighborhood and a training set with a relatively small size M.

REMARK 1. *The intuition behind our prediction algorithm can be illustrated by the Facebook1 graph in Fig. 2c, where nodes belonging to the same community are marked by the same color, and the size of each node is proportional to its local mixing time. We can see that the local mixing time for nodes residing in the same community does not vary greatly. The transitional change of local mixing time between different communities usually occurs at* marginal nodes *that connect two communities. Note that most nodes in the same community tend to share similar local neighborhood characteristics. For marginal nodes, their local characteristics result from a combination of several communities. Since the number of communities is small, a small number of training nodes are sufficient to map each node to its community and give a good prediction of its local mixing time.*

Evaluation. To evaluate the fitness of a regression model for a dataset, we employ two metrics. Given a dataset of n values denoted by $\{x_i | i = 1, 2, , n\}$, each associated with a predicted value y_i, the first metric is Root Mean Squared Error ($RMSE$), which is defined as $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2}$. It is the total root average squared difference between the predicted and the true response values. Lower $RMSE$ indicates a better prediction. The second metric is the coefficient of determination (r^2), defined as $r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2}$. It characterizes the correlation between the predicted and true response values. Higher r_2 indicates a better prediction.

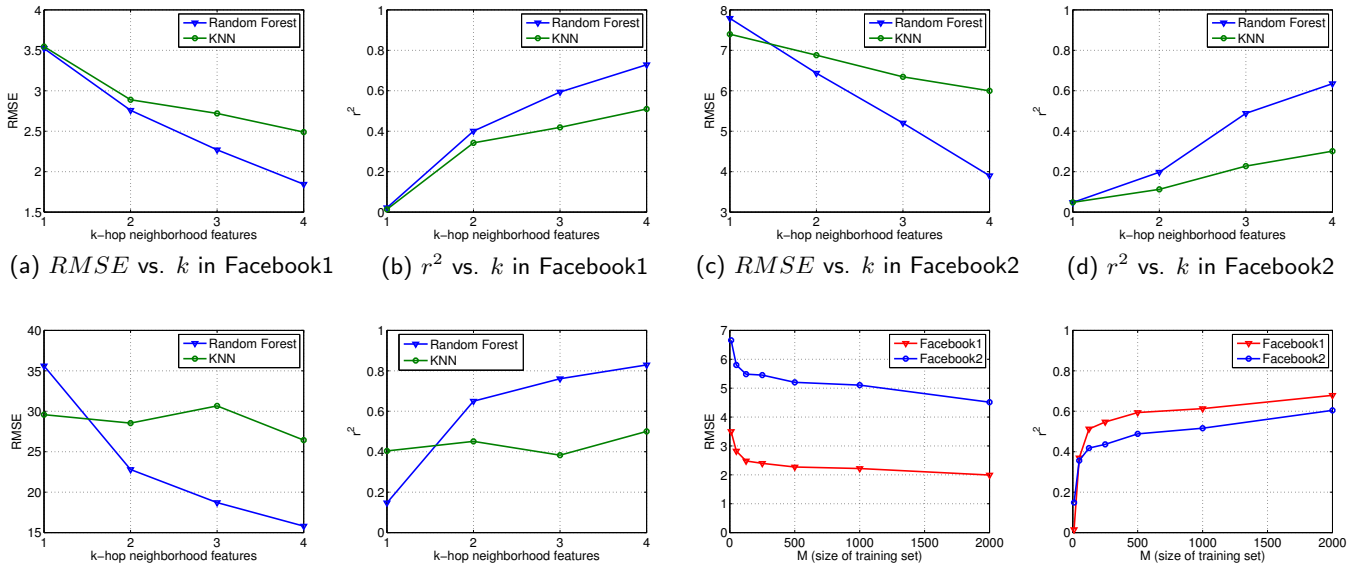
Algorithm 1 Local Mixing Time Prediction Algorithm

Step 1. Randomly select M nodes as training samples from graph G.
Step 2. (a) Starting from each training (target) node i, perform k-hop random walks and get the probability distribution $\pi_i(k)$ as the training (target) feature vector.
(b) Form the M-by-n training feature matrix $F_{train}(k)$ and the target feature martrix $F_{target}(k)$.
Step 3. (a) For each training node i, compute its local mixing time $T_i(\epsilon)$ as the training label.
(b) Form the M-by-1 training label vector $T_{train}(\epsilon)$.
Step 4. Fit a Random Forest regression model $\mathcal{M} = RF(F_{train}(k), T_{train}(\epsilon))$.
Step 5. For each target nodes, predict the local mixing time via the regression model, i.e., $T_{target}(\epsilon) = \mathcal{M}(F_{target}(k))$.

In the following, we present experimental results on Facebook1, Facebook2 and Twitter, and show that based merely on the local characteristics (i.e., features) and limited global information of the graph (i.e., training labels), it is possible to give a good estimation of the local mixing time for each node in the graph. Hence, we can estimate the approximate local mixing time for any user in distributed systems provided that the local mixing time of a small subset of nodes is broadcast to other users.

To evaluate our prediction algorithm, we set $\epsilon = 0.25$, $k = 1, 2, 3, 4$ and $M = 10, 50, 125, 250, 500, 1000, 2000$. We compare the results using two supervised learning methods, i.e., Random Forest regression and KNN regression. In our experiments, the number of estimators in Random Forest is set to 20, and the number of neighbors in KNN is set to 10.

Fig. 3a to Fig. 3f depict $RMSE$ and r^2 averaged over 100 iterations, respectively, using features within the k-hop neighborhood ($k = 1, 2, 3, 4$) at M=500. It can be seen that using Random Forest regression, $RMSE$ decreases and r^2



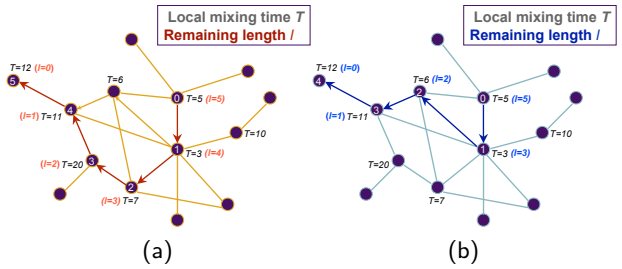
(a) $RMSE$ vs. k in Facebook1 (b) r^2 vs. k in Facebook1 (c) $RMSE$ vs. k in Facebook2 (d) r^2 vs. k in Facebook2

(e) $RMSE$ vs. k in Twitter (f) r^2 vs. k in Twitter (g) $RMSE$ vs. M (h) r^2 vs. M

Figure 3: (a) - (f) $RMSE$ *and* r_2 *using features within* k*-hop neighborhood* (M=500, averaged over 100 iterations) in Facebook1, Facebook2 and Twitter. (g) - (h) $RMSE$ *and* r_2 *versus the size of the training set* M (k=3, averaged over 100 iterations) in Facebook1 and Facebook2.

increases explicitly as k gets larger. This is because that using a wider neighborhood around node i gives a better match among the training nodes, which consequently results in the considerable improvement of the prediction performance. In contrast, $RMSE$ and r^2 using KNN regression do not vary significantly as k grows. For Facebook1 and Facebook2 graphs, the increase of k produces slightly better $RMSE$ and r^2. However, this observation does not hold for the Twitter graph. The weak impact of k on KNN can be explained by KNN's dependence upon the distance between feature vectors. Since the distance between the set of closest training samples to the new point almost remains unchanged with respect to k, we get similar prediction results. In general, Random Forest outperforms KNN from the perspective of both $RMSE$ and r^2, and its advantage gets more obvious with a larger value of k.

Fig. 3g and Fig. 3h depicts $RMSE$ and r^2 averaged over 100 iterations versus the number of training samples using Facebook1 and Facebook2 graphs at $k = 3$. It can be seen that $RMSE$ decreases and r^2 increases sharply when M grows from 10 to 125. This is because that a larger number of training samples implies more global knowledge of the graph. The resulting improvement gets slower when M exceeds 125. The choice of parameters k and M is dependent on the application requirement and varies among different datasets. According to the results in Section 3, we know that the mixing time is 90 for Facebook1, 179 for Facebook2, and 638 for Twitter. For these three social graphs, with $k = 3$ and $M = 500$, our prediction method produces acceptable performance with relatively small $RMSE$ and large r^2. In this case, M is approximately one-hundredth of the size of these three datasets, and k is about one-tenth or one-hundredth of the mixing time.



(a) (b)

Figure 4: Illustration of (a) Node-adaptive Random Walks and (b) Path-adaptive Random Walks.

5. SMARTWALK: USAGE MODEL

In this section, based on the results of the prediction algorithm in Section 4, we propose two usage models of adaptive random walks.

Adaptive across nodes. Given a random walk starting from node v_0, its local mixing time $T_{v_0}(\epsilon)$ measures the length required to converge to the stationary distribution. Hence, it is fairly straightforward to take the prediction value $\tilde{T}_{v_0}(\epsilon)$ as the random walk length when starting from node v_0. The parameter ϵ indicates the closeness between the terminus distribution and stationarity, and can be determined by the requirement of applications. Algorithm 2 determines the random walk length by the predicted $\tilde{T}_{v_0}(\epsilon)$, and thus is adaptive to v_0 (*node-adaptive*). As illustrated in Fig. 4a, the local mixing time of vertex v_0 (marked by 0 in Fig. 4a) is predicted as $T = 5$. Thus, any random walk starting from v_0 is a 5-hop random node sequence with successive nodes being neighbors. In this usage model, the walk length l only relys on the starting node, and is independent of any intermediate node along the path.

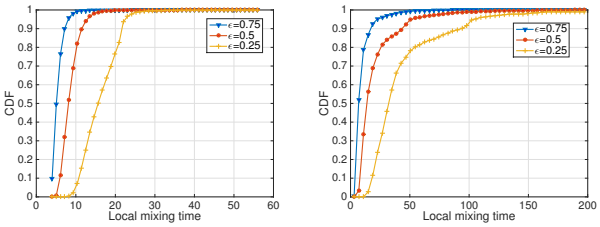
Algorithm 2 Node-adaptive Random Walks

Input: $\mathcal{G}, F_{train}(k), T_{train}(\epsilon), v_0$
Output: $\mathcal{W}$
Step 1. fit a Random Forest regression model $\mathcal{M} = RF(F_{train}(k), T_{train}(\epsilon))$.
Step 2. compute $F_{v_0}(k)$, a vector of features within the k-hop neightborhod of the initial node v_0.
Step 3. predict the local mixing time of v_0 via the regression model, i.e., $\tilde{T}_{v_0}(\epsilon) = \mathcal{M}(F_{v_0}(k))$.
Step 4. $\mathcal{W} = \{v_0\}$, $t = \tilde{T}_{v_0}(\epsilon)$, $v_p = v_0$.
Step 5. **while** $t > 0$
select a neighboring node v_{p+1} of v_p with probability $\frac{1}{deg(v_p)}$.
add v_{p+1} to the set $\mathcal{W}$.
$v_p = v_{p+1}$, $t = t - 1$.
end while

Algorithm 3 Path-adaptive Random Walks

Input: $\mathcal{G}, F_{train}(k), T_{train}(\epsilon), v_0$
Output: $\mathcal{W}$
Step 1. fit a Random Forest regression model $\mathcal{M} = RF(F_{train}(k), T_{train}(\epsilon))$.
Step 2. compute $F_{v_0}(k)$, a vector of features within the k-hop neightborhod of the initial node v_0.
Step 3. predict the local mixing time of v_0 via the regression model, i.e., $\tilde{T}_{v_0}(\epsilon) = \mathcal{M}(F_{v_0}(k))$.
Step 4. $\mathcal{W} = \{v_0\}$, $t = \tilde{T}_{v_0}(\epsilon)$, $v_p = v_0$.
Step 5. **while** $t > 0$
select a neighboring node v_{p+1} of v_p with probability $\frac{1}{deg(v_p)}$.
add v_{p+1} to the set $\mathcal{W}$.
repeat Steps 2 & 3 for node v_{p+1}, and get $\tilde{T}_{v_{p+1}}(\epsilon) = \mathcal{M}(F_{v_{p+1}}(k))$.
$v_p = v_{p+1}$, $t = \min\{\tilde{T}_{v_{p+1}}(\epsilon), t - 1\}$.
end while

Adaptive across nodes and paths. For node-adaptive random walks, the length of random walks starting from node v_0 is set as the predicted local mixing time of node v_0, i.e., $\tilde{T}_{v_0}(\epsilon)$. However, for a random walk initiated from the same node v_0, the remaining random walk length required to approach stationarity also depends on the path it has already covered. Specifically, if the random walk arrives at some intermediate node v_p, it might take no more than $T_{v_p}(\epsilon)$ additional hops to be within the ϵ-distance to the stationary distribution. Hence, to make the usage model adaptive both across nodes and across paths, each time an intermediate node v_p is reached, we update the value of the remaining random walk length if it is greater than the predicted local mixing time $\tilde{T}_{v_p}(\epsilon)$ of node v_p. The improved *path-adaptive* usage model is given in Algorithm 3. Fig. 4b illustrates an example of a path adaptive walk. At the beginning, the local mixing time of the initial vertex v_0 is predicted to be $T = 5$. After one hop, we arrive at node v_1 with its local mixing time $T = 3$. According to the new information provided by v_1, the remaining necessary walk length l is updated to 3 instead of $5 - 1 = 4$. The second hop takes us to node v_2 with $T = 6$, which is greater than the number of remaining hops. Thus we keep $l = 2$. Repeat this process, and eventually we terminate at node v_4. The total walk length is 4, which is determined by considering all the nodes along the walk.



(a) Facebook wall post (b) Twitter

Figure 5: CDF of local mixing time with respect to ϵ.

6. SECURITY APPLICATIONS

In this section, we demonstrate the applicability of our two usage models to social network based security systems, including Sybil defense, anonymous communication and link privacy preserving systems..

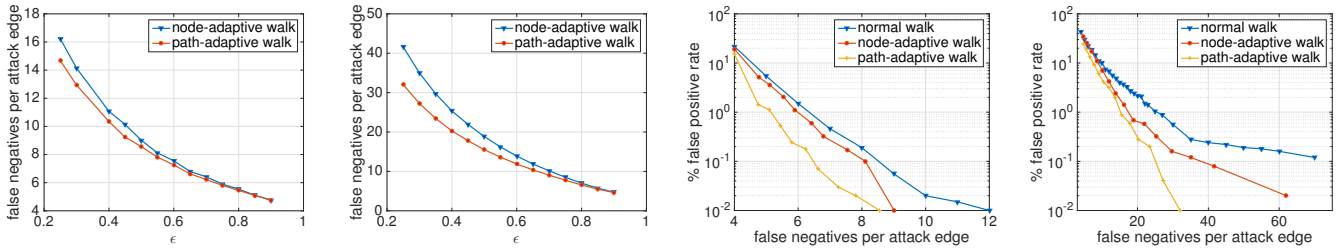
6.1 Sybil defense

A Sybil attack is an attack wherein a single user forges a large number of pseudonymous identities. Recent work has proposed Sybil defense mechanisms by leveraging the trust relationships in social networks [51, 52, 35, 24, 13, 44, 43]. The key insight is that it is difficult for an adversary to establish trust relationships with honest users (attack edges), particularly when interaction networks are used to incur a higher cost for adversaries to set up an attack edge [47, 16]. SybilLimit [51] is a Sybil defense protocol that performs random walk based routes on social graphs and registers public keys with the tails (terminus points of random routes) to differentiate Sybil users from benign users.

Security/false positive rate trade-off. (1) We evaluate the security performance of SybilLimit in terms of the number of Sybils that an adversary can insert in the honest region (false negatives). Note that the expected number of Sybil nodes that an adversary can insert in the honest region is given by $E[S] = E[g \cdot W] = g \cdot E[W]$, where g is the number of attack edges and W is the random walk length. Then the false negatives per attack edge is $E[W]$. (2) Another critical metric to evaluate the accuracy of SybilLimit is the *false positive rate* (percentage of benign users misclassified as Sybils). In SybilLimit, only random walks of fixed length are performed, i.e., $W = w$ for all walks, and the value of w is usually chosen to be the graph mixing time to ensure a low false postive rate [51], which is unnecessarily large for most nodes and only severely degrades the security performance. Hence, we are interested in applying adaptive random walks when generating random routes in SybilLimit. We compare the security/false positive rate trade-off when generating random routes based on three different random walks, i.e., fixed-length random walks (normal random walks), node-adapative random walks and path-adaptive random walks. We show that among these three usage models, our node-adaptive and path-adaptive random walks guarantee stronger system security by decreasing false negatives for any desired false positive rate.

Evaluation. We use the Facebook wall post dataset in [34] (with 29,060 nodes and 169,752 edges) and the Twitter dataset in Table 1 (with 81,306 nodes and 1,342,296 edges). [8] Using two adaptive random walk models, Fig. 6a and Fig.

[8]The Facebook wall post dataset is an interaction network



(a) Facebook wall post (b) Twitter (c) Facebook wall post (d) Twitter

Figure 6: (a) - (b) *false negatives per attack edge* as a function of ϵ; (c) - (d) *false positive rate* as a function of *false negatives per attack edge.*

6b depict the false negatives per attack edge as a function of ϵ for the Facebook wall post graph and the Twitter graph, respectively. Thus, false negatives are tunable by setting the variation distance parameter ϵ to various values from 0 to 1 (from strong to weak convergence). Also note that to achieve a certain distance ϵ, adaptive models result in a significantly smaller false negatives per attack edge than the mixing time $w = T(\epsilon)$ used by fixed-length systems. For instance, in Fig. 6a, at $\epsilon = 0.25$, the false negatives (≈ 15) are reduced by a *factor* of 3.7 compared to the mixing time (≈ 56) in Fig. 5a.

Fig. 6c and Fig. 6d illustrate the false positive rate versus false negatives per attack edge ($E[W]$) using three random walk models for the Facebook wall post graph and the Twitter graph, respectively. It can be seen that the path-adaptive random walks achieve the best security/false positive rate trade-off among the three, while the node-adaptive random walks come to be the second best. Specifically, for the Facebook wall post graph (Fig. 6c), the false positive rate after the adoption of path-adaptive walks shows a decline from 1.3% to 0.2% at $E[w] = 6$ and from 0.2% to 0.015% at $E[W] = 8$, i.e., the false positive rate is reduced by an order of magnitude compared to the classical fixed-length walks (y-axis is in log-scale). In other words, the accuracy of classifying benign users is considerably improved. For the Twitter graph, we observe that *the false positive rate can be reduced by up to two orders of magnitude (at false negatives = 30) using the path-adaptive walks.*

We conclude that both path-adaptive and node-adaptive random walk models outperform the classical fixed-length model. This is because our adaptive walk algorithms reduce the walk length of most nodes to a large extent while still ensuring that their distance to stationarity is sufficiently close. The path-adaptive walk model works better than the node-adaptive one since it leverages the information of nodes along the path to further decrease the number of unnecessary hops. As discussed above, the path-adaptive random walk model results in significant improvements in accuracy and security trade-offs (by up to two orders of magnitude).

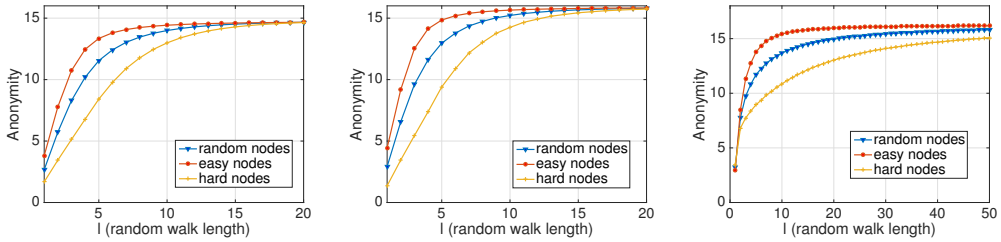
6.2 Anonymous systems

Anonymous communication systems preserve users' privacy by hiding the communication link between the user and the remote communicating entity. Nagaraja *et al.* and others [37, 34, 12, 15] proposed several anonymous system designs that enhance the security properties by leveraging trust relationships to select proxies which are more likely to be honest. The Pisces protocol [34] is a low-latency anonymity system that leverages social links. Similar to the Tor protocol, users in Pisces rely on proxy servers and onion routing for anonymous communication. Specifically, the relays involved in the onion routing path are chosen by performing a random walk on a trusted social network topology. In [34], the anonymity performance is evaluated based on the *Shannon entropy*, which considers the probability distribution of nodes being possible initiators as computed by the attackers.

Anonymity/latency trade-off. Both shannon entropy and latency are significantly influenced by the length of random walks l. Given a node i, as the random walk gets longer, the node's entropy increases and eventually converges to some value (indicating stronger system anonymity), meanwhile the latency gets larger. Since latency is roughly proportional to the walk length, we use the *expected random walk length* as the latency metric. In prior works, all random walks have the same length. Specifically, in [34], the random walk length l is set to a fixed value such that the expected entropy of a random sample of 5% nodes is above a threshold. We first show that nodes with greater local mixing time usually require a longer random walk to achieve the same level of anonymity as other nodes. As a result, using Jain's fairness index [21], we show that the fixed length method used in [34] has poor *fairness* of anonymity due to its ignorance of a minority of nodes that needs larger l. Then we demonstrate that using an adaptive length method instead enhances the fairness for any given expected random walk length.

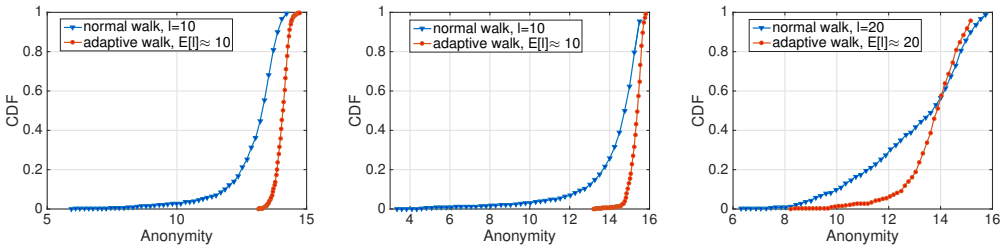
Evaluation. We use the Facebook wall post dataset in [34], along with the Facebook link (Facebook1) and Twitter datasets in Table 1. We rank the nodes according to their local mixing time in a descending order, and compare the anonymity (entropy) averaged over the top 5% nodes (*hard nodes*), the last 5% nodes (*easy nodes*) and random 5% nodes, as illustrated in Fig. 7. It can be seen that the convergence rate of easy nodes' anonymity to the upper bound is much faster than that of hard nodes. However, in a normal random walk scheme, l is set to be identical for every node. Note that in social graphs, a majority of nodes are easy nodes while hard nodes take only a small portion (see Fig. 5a and Fig. 5b). Consequently, as illustrated in Fig. 7, the expected anonymity/entropy for a random sample only reflects the behaviour of most easy nodes rather than that of hard nodes. In Fig. 7a, if we set the threshold as 14, we

and thus implies stronger social ties than the Facebook link dataset. For the Twitter dataset, we only preserve a link between two users if they follow each other such that a link indicates a close relationship between the two users.



(a) Facebook wall post (b) Facebook link (c) Twitter

Figure 7: Anonymity as a function of random walk length.



(a) Facebook wall post (b) Facebook link (c) Twitter

Figure 8: CDF of anonymity for the hard nodes.

will choose $l = 10$ for all nodes, which is in fact insufficient for over 5% of the nodes (i.e., hard nodes) that need l to be at least 13. The approach for determining a fixed walk length based on average anonymity can lead to a more severe anonymity loss for hard nodes in larger social graphs, as indicated in Fig. 7b and Fig. 7c.

To illustrate the anonymity loss of hard nodes, Fig. 8 depicts the CDF of anonymity over the hard nodes, using different random walk schemes. For the Facebook wall post graph, in the case of normal random walks, we choose $l = 10$ based on Fig. 7a so that the expected anonymity reaches 14. Then about 90% of hard nodes fail to reach 13.5, and their minimum anonymity even drops to 6. To ensure that more than 99% nodes meet the threshold requirement, we have to assign an unnecessarily large value to l (around 20 in this case), which incurs long latency. For adaptive random walks [9], we are able to adaptively perform short random walks for the majority of nodes and relatively long walks for the rest. Using the prediction algorithm, we can detect the existence of hard nodes. In the Facebook wall post graph, the necessary walk length is predicted for a set of different ϵ's. We choose $\epsilon = 0.65$, which produces an average walk length that is close to the fixed length used in normal walks, i.e., 10. From Fig. 8a, we can see that after the adaptive walk model is applied, the percentage of hard nodes with anonymity greater than 13.5 rises to 90%, whereas the expected length $E[l]$ is still small (≈ 10). Our adaptive random walk algorithm also results in a significant increase of the minimum anonymity from 6 to 13. *Note that the entropy metric characterizes the anonymity using a logarithmic scale; thus an increase of entropy from 6 to 13 results in 2 orders of magnitude larger anonymity set size.*

To quantify the fairness of anonymity among nodes, we introduce the Jain's fairness index [21] given by $\mathcal{F}(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i)(x_i)^2}$, which measures the fairness of a set of values where there are n users with each assigned with the throughput x_i. The fairness metric ranges from $\frac{1}{n}$ (the worst case) to 1 (the best case), with the maximum value obtained at the uniform allocation over all users. In the scenario of anonymous communications, we take x_i as the anonymity set of each node v_i. Fig. 9 illustrates the fairness versus the average walk length in two random walk models. We conclude that the adaptive walk scheme significantly strengthens the anonymity of hard nodes and thus enhances the fairness.

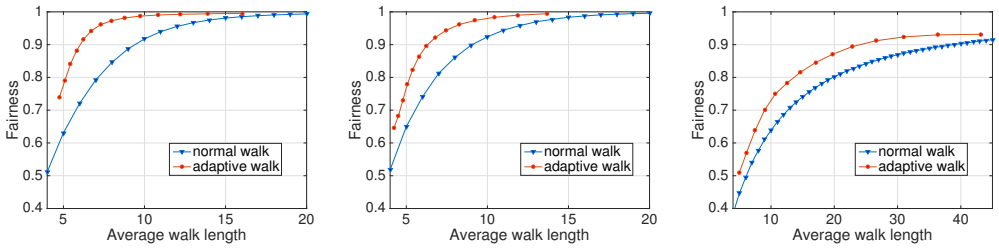
[9]We mainly consider applying node-adaptive random walks to the anonymous communication systems.

6.3 Link Privacy

Extensive research has been carried out to protect the privacy of trust relationships between any pair of users (link privacy) [19, 20, 50, 54, 33, 27]. The challenge of preserving link privacy lies in causing no significant losses on the utility of applications that leverage the social trust relationships. Specifically, link privacy is preserved by adding extra noise to the local structure of a social network. At the same time, global structural characteristics are maintained to ensure that the utility of the social network is not severely reduced. This can be implemented by replacing a real link between two users with a fake link generated by a random walk[33].

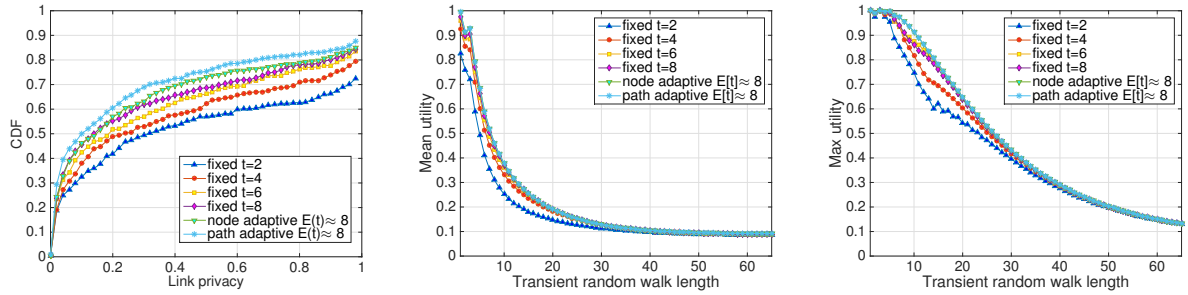
Link privacy/utility trade-off. Mittal *et al.* in [33] considered that the length of random walks for all nodes has a fixed value. As the length increases (more noise), the perturbed social graph converges to a random graph and its utility declines drastically. Our key insight is that instead of adding identical amount of noise to all users, perturbation can be unevenly distributed according to the local mixing time such that privacy can be protected with less perturbation on average. In other words, we can perform node-adaptive random walks rather than random walks with a fixed length for every user when generating fake links.

To evaluate different perturbation algorithms, we use the definitions of utility and link privacy in [33], which are based



(a) Facebook wall post (b) Facebook link (c) Twitter

Figure 9: Fairness versus average random walk length.



(a) Link privacy (b) Mean utility (c) Max utility

Figure 10: (a) CDF of link privacy; (b) *Mean utility* and (c) *max utility of the perturbed graph* G' versus transient random walk length.

on the transition matrices and the Bayesian inference, respectively.

Definition 2. *The overall mean utility of a perturbed graph G' with respect to the original graph G and an application parameter l is defined as the mean utility for all nodes in G, i.e., $U_{mean}(G,G',l) = \frac{1}{|V|}\sum_{i\in V}|\pi_i(0)(P^l(G) - P^l(G'))|_1$. Similarly, the maximum utility (worst case) of a perturbed graph G' is defined by computing the maximum of the utility over all nodes in G, i.e., $U_{max}(G,G',l) = \max_{i\in V}\{|\pi_i(0)(P^l(G) - P^l(G'))|_1\}$.*

Definition 3. *The link privacy of a link L is defined as the probability of the existence of the link in the original graph G under the assumption that the adversary has access to the perturbed graph G' and prior information H, i.e., $LP(L,G',H) = \Pr[L = 1|G',H]$.*

Note that smaller distances indicate higher utility performance, and smaller probabilities provide higher privacy protection. We consider the worst-case link privacy by assuming that the adversary has the information of the entire original graph without the link L, i.e., $H = G - L$.

Evaluation. We use the Facebook link graph. Fig. 10a illustrates the CDF of link privacy $\Pr[L|G',H]$ under the worst case prior $H = G - L$. From Fig. 10a, we can see that as the perturbation t gets larger, the percentage of links with lower link privacy increases, indicating higher privacy. By making t adaptive to different nodes (ranging from 5 to 36 with $E(t) \approx 8$), a larger portion of nodes have low probabilities compared to the fixed perturbation algorithm at $t = 8$, which indicates that privacy is better preserved. This is because we make every node get its minimum required perturbation using our adaptive random walk models, which offer a higher level of privacy for a given expected walk length. Fig. 10b and Fig. 10c illustrate the mean utility and the max utility of the perturbed graph G' versus the transient random walk length, respectively. We can see that by making t adaptive to different nodes (ranging from 5 to 36) with the average value around 8, the utility degradation is minimal compared to $t = 8$. Combining Fig. 10a, Fig. 10b and Fig. 10c, our adaptive perturbation algorithms improve the privacy performance at the cost of slight degradation in utility.

7. FURTHER DISCUSSION

We leverage supervised machine learning techniques to predict the local mixing time of a given node, which requires the knowledge of k-hop neighborhood features. In centralized systems where the graph is globally known, features can be directly computed and the total computation time is $O(\Gamma(k))$, where $\Gamma(k)$ is the number of k-hop neighbors. We benchmark the computational overhead on a machine running a Linux 2.6.32 kernel with a 2.5 GHz Intel Xeon core. The average computational time for 3-hop features is about 100 milliseconds using Facebook1, 30 milliseconds using Facebook2 and 170 miliseconds using Twitter. For distributed systems and a given node i, its k-hop features can approximated by performing a sufficiently large number of k-hop random walks from i and obtaining the frequency of different terminus nodes.

Random walks are naturally resilient to Sybil attacks [51, 52], since the Sybil users have limited power in corrupting the close neighborhood of honest users. However, the effect of poisoning attacks [26] on the probe method is still an interesting research question. Possible defenses against poisoning attacks are to provide more robust training node

selection by performing short random walks from prior trust seeds, or to develop detection methods for nodes to examine potentially poisoned features and labels. We will leave the safe adoption of machine learning techniques in adversarial settings to future work.

8. RELATED WORK

Random Walks in Security Systems. Danezis *et al.* [12] proposed Drac, a decentralized protocol for anonymous communications that leverages users' social links. Random walks are used in the circuit creation process. Mittal *et al.* [33] presented a random walk based perturbation algorithm, which anonymizes the social trust relationships by replacing real edges with edges between initial and terminal nodes of random walks. Many Sybil defense mechanisms leverage random walks to detect Sybil users from benign users, such as SybilLimit [51], SybilGuard [52], SybilRank [8], SybilInfer [13] and Criminal account Inference Algorithm [49]. Integro [5] changes the transition probabilities associated with random walks, but still uses fixed-length random walks. Using multiple real world social network datasets, we experimentally verify the applicability of adaptive random walk models in Sybil defense, anonymous systems and link privacy preserving systems. We show that our proposal has the potential to improve the security and privacy of these applications by an order of magnitude. We note that our approach has broad potential to impact security-performance trade-offs in applications even outside the context of social networks; this includes graph-theoretic detection mechanisms for P2P botnets [38], spamming botnets [53], malicious online communities [42], and malware [22].

Random Walks in Networked Systems. Lovász *et al.* [29] describes the connection of mixing time to the second largest eigenvalue modulus (SLEM) of graphs. A lot of works have studied the impact of network topology on the mixing times of random walks experimentally [11, 37, 36]. To the best of our knowledge, all previous works use uniform length random walks, and our work is the first to adapt the random walk length depending on structural characteristics of nodes, and apply the concept to improve system security and privacy. Even though our analysis was presented from the perspective of undirected graphs, our idea can be extended to both weighted and directed networks, and such a quantitative study would be an interesting direction of future work.

9. CONCLUSIONS

In this paper, we observe that in various social topologies, the walk length required to converge to stationarity has a heterogeneous and long-tail property across nodes. Using a set of supervised machine learning techniques, we show that the walk length for a specific node can be well predicted given the local characteristics and limited knowledge for a small set of training nodes. Based on the heterogeneous property and prediction algorithm, we propose two usage models of random walks that can adaptively change the random walk length, i.e., node-adaptive and path-adaptive random walks. Finally, we present experimental results using two usage models in real world social network based security applications, and show up to two orders of magnitude improvement in performance.

10. ACKNOWLEDGMENTS

This work was supported in part by NSF awards number CNS-1409415, CNS-1423139, CNS-1553437, and CNS-1617286, and by CCF-Tencent Open Research Fund.

11. REFERENCES

[1] Google unveils uproxy, an anti-censorship browser extension. http://thetechjournal.com/internet/google-unveils-uproxy.xhtml.

[2] Lantern - open internet for everyone. http://www.getlantern.org.

[3] D. Aldous. Random walks of finite groups and rapidly mixing Markov chains. In J. Azéma and M. Yor, editors, *Séminaire de Probabilités XVII 1981/82*, volume 986 of *Lecture Notes in Mathematics*, pages 243–297. Springer-Verlag, 1983.

[4] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *HICSS 2006*. IEEE Computer Society.

[5] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov. Integro: Leveraging victim prediction for robust fake account detection in osns. In *NDSS*, 2015.

[6] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review, problems and techniques section*, 46(4):667–689, Dec. 2004.

[7] L. Breiman. Random forests. *Machine Learning*, 2001.

[8] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI 2012*, pages 197–210.

[9] R. Caruana. Multitask learning. *Machine Learning*, 1997.

[10] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions, 2007.

[11] M. Chen. Mixing time of random walks on graphs. Master's thesis, University of York, 2004.

[12] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. *PET*, 2010.

[13] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, 2009.

[14] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *The Annals of Applied Probability*, 1:36–61, 1991.

[15] R. Dingledine, N. Mathewson, and P. Tor: The second-generation onion router. *USENIX*, 2004.

[16] E. Gilbert and K. Karahalios. Predicting tie strength with social media. *CHI*, 2009.

[17] D. Gkorou, T. Vinko, J. Pouwelse, and D. Epema. Leveraging node properties in random walks for robust reputations in decentralized networks. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, 2013.

[18] S. J. Hardiman and L. Katzir. Estimating clustering coefficients and size of social networks via random walk. In *WWW*, 2013.

[19] M. Hay, G. Miklau, D. Jensen, D. Towsley, and C. Li. Resisting structural re-identification in anonymized

social networks. *the International Journal on Very Large Data Bases*, 2008.

[20] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *the International Journal on Very Large Data Bases*, 2007.

[21] R. Jain, D.-M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. 1998.

[22] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitras. The dropper effect: Insights into malware distribution with downloader graph analytics. In *ACM SIGSAC*, pages 1118–1129. ACM, 2015.

[23] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[24] C. Lesniewski-Laas and M. F. Kaashoek. Whanau: A sybil-proof distributed hash table. *NSDI*, 2012.

[25] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2009.

[26] C. Liu, P. Gao, M. Wright, and P. Mittal. Exploiting temporal dynamics in sybil defenses. In *ACM SIGSAC*, 2015.

[27] C. Liu and P. Mittal. Linkmirage: Enabling privacy-preserving analytics on social relationships. 2016.

[28] W. Liu and L. Lu. Link prediction based on local random walk. *Epl*, 2010.

[29] L. LovÃąsz. *Random Walks on Graphs: A Survey*, volume 2. János Bolyai Mathematical Society, 1996.

[30] J. Lu and D. Li. Sampling online social networks by random walk. In *ACM IWHTISNR*, 2012.

[31] S. Marti, P. Ganesan, and H. Garcia-Molina. *Sprout: P2P Routing with Social Networks*. Springer Berlin Heidelberg, 2004.

[32] P. Mittal, M. Caesar, and N. Borisov. X-vine: Secure and pseudonymous routing using social networks. *Network and Distributed System Security Symposium*, 2012.

[33] P. Mittal, C. Papamanthou, and D. Song. Preserving link privacy in social network based systems. In *NDSS*, 2013.

[34] P. Mittal, M. Wright, and N. Borisov. Pisces: Anonymous communication using social networks. *NDSS*, 2013.

[35] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. In *INFOCOM*, 2011.

[36] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010.

[37] S. Nagaraja. Anonymity in the wild: Mixes on unstructured networks. *PET*, 2007.

[38] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. Botgrep: Finding p2p bots with structured graph analysis. In *USENIX*, pages 95–110, 2010.

[39] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 2005.

[40] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*. Springer, 2005.

[41] Y. Sovran, J. Li, and L. Subramanian. Unblocking the internet: Social networks foil censors.

[42] G. Stringhini, P. Mourlanne, G. Jacob, M. Egele, C. Kruegel, and G. Vigna. Evilcohort: detecting communities of malicious accounts on online services. In *USENIX*, pages 563–578, 2015.

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

[44] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. *NSDI*, 2009.

[45] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *WOSN*, 2009.

[46] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *STOC*, 2012.

[47] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. *Acm Eurosys*, 2009.

[48] Y. Xie, Z. Chen, A. Agrawal, A. Choudhary, and L. Liu. Random walk-based graphical sampling in unbalanced heterogeneous bipartite social graphs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, 2013.

[49] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *WWW*, pages 71–80. ACM, 2012.

[50] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. *SDM*, 2008.

[51] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE SP*, 2008.

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

[53] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. Botgraph: Large scale spamming botnet detection. In *USENIX*, NSDI'09, pages 321–334, 2009.

[54] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, 2007.

APPENDIX

A. PROOF OF INEQUALITY

PROOF. Given node i, the probability distribution at time t is given by $\pi_i(t) = \pi_i(0)P^t$. Assume $t > 1$. $\pi_i(t) = \frac{1}{deg(i)} \sum_{j \in N(i)} \pi_j(0)P^{t-1}$. Let $T_{max} = \max_{j \in N(i)} T_j(\epsilon)$. Denote the total variation distance to stationarity at time t as $\Delta_i(t)$. Then for any neighbouring node j of node i, $\Delta_j(T_{max}) < \epsilon$. Thus, we have $\Delta_i(T_{max}+1) = |\pi - \frac{1}{deg(i)} \sum_{j \in N(i)} \pi_j(0)P^{T_{max}}|_1 = |\frac{1}{deg(i)} \sum_{j \in N(i)} (\pi - \pi_j(0)P^{T_{max}})|_1 \leq \frac{1}{deg(i)} \sum_{j \in N(i)} \Delta_j(T_{max}) < \epsilon$. Since $\Delta_i(T_{max}+1) < \epsilon$, the local mixing time of node i must not exceed $T_{max}+1$. The proof is completed. □