LinkMirage: Enabling Privacy-preserving Analytics on Social Relationships

Changchang Liu, Prateek Mittal

Email: cl12@princeton.edu, pmittal@princeton.edu

Department of Electrical Engineering, Princeton University

Abstract—Social relationships present a critical foundation for many real-world applications. However, both users and online social network (OSN) providers are hesitant to share social relationships with untrusted external applications due to privacy concerns. In this work, we design LinkMirage, a system that mediates privacy-preserving access to social relationships. LinkMirage takes users' social relationship graph as an input, obfuscates the social graph topology, and provides untrusted external applications with an obfuscated view of the social relationship graph while preserving graph utility.

Our key contributions are (1) a novel algorithm for obfuscating social relationship graph while preserving graph utility, (2) theoretical and experimental analysis of privacy and utility using real-world social network topologies, including a large-scale Google+ dataset with 940 million links. Our experimental results demonstrate that LinkMirage provides up to 10x improvement in privacy guarantees compared to the state-of-the-art approaches. Overall, LinkMirage enables the design of real-world applications such as recommendation systems, graph analytics, anonymous communications, and Sybil defenses while protecting the privacy of social relationships.

I. INTRODUCTION

Online social networks (OSNs) have revolutionized the way our society interacts and communicates with each other. Under the hood, OSNs can be viewed as a special graph structure composed of individuals (or organizations) and connections between these entities. These *social relationships* represent sensitive relationships between entities, for example, trusted friendships or important interactions in Facebook, Twitter, or Google+, which users want to preserve the security and privacy of.

At the same time, an increasing number of third party applications rely on users' social relationships (these applications can be external to the OSN). E-commerce applications can leverage social relationships for improving sales [21], and data-mining researchers also rely on the social relationships for functional analysis [31], [33]. Social relationships can be used to mitigate spam [26]. Anonymous communication systems can improve client anonymity by leveraging users' social relationships [11], [28], [29]. State-of-the-art Sybil defenses

rely on social trust relationships to detect attackers [8], [43].

However, both users and the OSN providers are hesitant to share social relationships/graphs with these applications due to privacy concerns. For instance, a majority of users are exercising privacy controls provided by popular OSNs such as Facebook, Google+ and LinkedIn to limit access to their social relationships [9]. Privacy concerns arise because external applications that rely on users' social relationships can either explicitly reveal this information to an adversary, or allow the adversary to perform inference attacks [14], [20], [24], [30], [32], [37]. These concerns hinder the deployment of many real-world applications. Thus, there exist fundamentally conflicting requirements for any link obfuscation mechanism: protecting privacy for the sensitive links in social networks and preserving utility of the obfuscated graph for use in real-world applications.

In this work, we design LinkMirage, a system that mediates privacy-preserving access to social relationships. LinkMirage takes users' social relationship graph as an input, either via an OSN operator or via individual user subscriptions. Next, LinkMirage *obfuscates* the social graph topology to protect the privacy of users' social contacts (edge/link privacy, not vertex privacy). LinkMirage then provides external applications such as graph analytics and anonymity systems [11], [28], [29] with an obfuscated view of the social relationship graph. Thus, LinkMirage provides a trade-off between securing the confidentiality of social relationships, and enabling the design of social relationship based applications.

We present a novel obfuscation algorithm that first clusters social graphs, and then anonymizes intra-cluster links and inter-cluster links, respectively. We obfuscate links in a manner that preserves the key structural properties of social graphs. While our approach is of interest even for static social graphs, we go a step further in this paper, and consider the evolutionary dynamics of social graphs (node/link addition or deletion). We design LinkMirage to be resilient to such evolutionary dynamics, by *consistently* clustering social graphs across time instances. Consistent clustering improves both the privacy and utility of the obfuscated graphs. We show that LinkMirage provides strong privacy properties. Even a strategic adversary with full access to the obfuscated graph and prior information about the original social graph is limited in its ability to infer information about users' social relationships. LinkMirage provides up to 3x privacy improvement in static settings, and up to 10x privacy improvement in dynamic settings compared to the state-of-the-art approaches.

Overall, our work makes the following contributions.

- First, we design LinkMirage to mediate privacy-preserving access to users' social relationships. LinkMirage obfuscates

links in the social graph (link privacy) and provides untrusted external applications with an obfuscated view of the social graph. LinkMirage can achieve a good balance between privacy and utility, under the context of both static and dynamic social network topologies.

- Second, LinkMirage provides rigorous privacy guarantees to defend against strategic adversaries with prior information of the social graph. We perform link privacy analysis both theoretically as well as using real-world social network topologies. The experimental results for both a *Facebook* dataset (with 870K links) and a large-scale *Google+* dataset (with 940M links) show up to 10x improvement in privacy over the state-of-the-art research.

- Third, we experimentally demonstrate the applicability of LinkMirage in real-world applications, such as privacy-preserving graph analytics, anonymous communication and Sybil defenses. LinkMirage enables the design of social relationships based systems while simultaneously protecting the privacy of users' social relationships.

- Finally, we quantify a general utility metric for LinkMirage. We analyze our utility measurement provided by LinkMirage both theoretically and using real-world social graphs (*Facebook* and *Google+*).

II. BACKGROUND

A. Motivating Applications

In this paper, we focus our research on protecting the *link* privacy between *labeled* vertices in social networks [16], [27], [42]. Mechanisms for graph analytics, anonymous communication, and Sybil defenses can leverage users' social relationships for enhancing security, but end up revealing users' social relationships to adversaries. For example, in the Tor network [11], the relays' IP addresses (labels) are already publicly known (vertex privacy in [25], [34], [45] is not useful). Tor operators are hesitant to utilize social trusts to set up the Tor circuit as recommended by [28], [29] since the circuit construction protocol would reveal sensitive social contact information about the users. Our proposed link-privacy techniques can thus be utilized by the Tor relay operators to enhance system security while preserving link privacy. Overall, our work focuses on protecting users' trust relationships while enabling the design of such systems.

LinkMirage supports three categories of social relationship based applications: 1) Global access to the obfuscated graph: Applications such as social network based anonymity systems [11], [28], [29] and peer-to-peer networks [8] can utilize LinkMirage (described in Section IV-B) to obtain a global view of privacy-preserving social graph topologies; 2) Local access to the obfuscated graph: an individual user can query LinkMirage for his/her obfuscated social relationships (local neighborhood information), to facilitate distributed applications such as SybilLimit [43]; 3) Mediated data analytics: LinkMirage can enable privacy-preserving data analytics by running desired functional queries (such as computing graph modularity and pagerank score) on the obfuscated graph topology and only returning the result of the query, since immediate analytics operated on the original graph topology would leak sensitive information about users' social relationships. Existing work [12], [13] demonstrated that the implementation of graph analytics algorithms could leak certain information. Instead of repeatedly adding perturbations to the output of each graph analytics algorithm as in differential privacy [12], [13], which
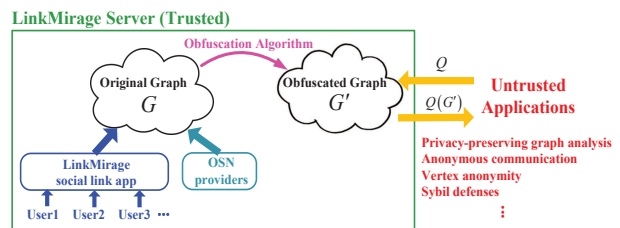

Fig. 1. LinkMirage architecture. LinkMirage first collects social link information through our social link app or directly through the OSN providers, and then applies an obfuscation algorithm to perturb the original social graph(s). The obfuscated graph(s) would be utilized to answer the query of the untrusted applications in a privacy-preserving manner. The third-party application (which queries the social link information) is considered an adversary which aims to obtain sensitive link information from the perturbed query results.

would be rather costly, LinkMirage can obtain the perturbed graph just once to support multiple graph analytics. Such an approach protects the privacy of users' social relationships from inference attacks using query results. There exists a plethora of attacks against vertex anonymity based mechanisms [20], [30], [32], [37]. Ji et al. [19] recently showed that no single vertex anonymization technique was able to resist all the existing attacks. Note that these attacks are not applicable to link privacy schemes. Therefore, a sound approach to vertex anonymity must start with improvements in our understanding of link privacy. When used as first step in the design of vertex privacy mechanisms, our approach can protect the privacy of social contacts and graph links even when the vertices are de-anonymized using state-of-the-art approaches [20], [30], [32], [37]. Furthermore, our method can even improve the resilience of vertex anonymity mechanisms against de-anonymization attacks when applied to *unlabelled* graphs (will be shown in Section VI-B).

B. System Architecture and Threat Model

Fig. 1 shows the overall architecture for LinkMirage. For link privacy, we consider the third-party applications (which can query the social link information) as adversaries, which aim to obtain sensitive link information from the perturbed query results. A sophisticated adversary may have access to certain prior information such as partial link information of the original social networks, and such prior information can be extracted from publicly available sources, social networks such as Facebook, or other application-related sources as stated in [6]. The adversary may leverage Bayesian inference to infer the probability for the existence of a link. We assume that LinkMirage itself is trusted, in addition to the social network providers/users who provide the input social graph.

In Section V-B, V-C, we define our Bayesian privacy metric (called *anti-inference privacy*) and an information theoretic metric (called *indistinguishability*) to characterize the privacy offered by LinkMirage against adversaries with prior information. In addition, the evolving social topologies introduce another serious threat where sophisticated adversaries can combine information available in multiple query results to infer users' social relationships. We define *anti-aggregation privacy* in Section V-D, for evaluating the privacy performance of LinkMirage against such adversaries.

C. Basic Theory

Let us denote a time series of social graphs as $G_0, \cdots, G_T$. For each temporal graph $G_t = (V_t, E_t)$, the set of vertices is V_t and the set of edges is E_t. For our theoretical analysis, we focus on undirected graphs where all the $|E_t|$ edges are

symmetric, i.e. $(i,j) \in E_t$ iff $(j,i) \in E_t$. Note that our approach can be generalized to directed graphs with asymmetric edges. P_t is the transition probability matrix of the Markov chain on the vertices of G_t. P_t measures the probability that we follow an edge from one vertex to another vertex, where $P_t(i,j) = 1/\deg(i)$ ($\deg(i)$ denotes the degree of vertex i) if $(i,j) \in E_t$, otherwise $P_t(i,j) = 0$. A random walk starting from vertex v, selects a neighbor of v at random according to P_t and repeats the process.

III. SYSTEM OVERVIEW AND ROADMAP

A. LinkMirage System

Our objective for LinkMirage is to obfuscate social relationships to strike a balance between privacy for users' social relationships and the usability for large-scale real-world applications, as will be discussed in Section IV-A. We deploy LinkMirage as a Facebook application that implements graph construction and obfuscation, as will be discussed in Section IV-B.

For our perturbation mechanism of LinkMirage, we take both the static and the temporal social network topology into consideration, as will be discussed in Section IV-C. Our obfuscation mechanism consists of the following conceptual steps:

- Dynamic clustering which finds community structures in evolving graphs by simultaneously considering consecutive graphs. Our dynamic clustering utilizes an effective back-tracking strategy to cluster a graph based on the clustering result of the previous graph.
- Selective perturbation which perturbs the minimal amount of edges in the evolving graphs according to the dynamic clustering result, where we only perturb the changed communities between consecutive graphs. In this manner, it is possible to use a very high privacy parameter in the perturbation process, while preserving structural properties of the social network topologies.

We further discuss the scalability of our perturbation algorithm on the real world large-scale Google+ dataset in Section IV-D and visually show the effectiveness of our algorithm on the real world Facebook dataset in Section IV-E.

B. Privacy Evaluation

In Section V, we rigorously analyze the privacy advantage of our LinkMirage over the state-of-the-art approaches, by considering three adversarial scenarios where sophisticated adversaries can combine information available in multiple query results to infer users' social relationships. The privacy advantage of our LinkMirage will be demonstrated as

- LinkMirage shows significant privacy advantages in *anti-inference privacy* (will be defined in Section V-B). The difference between the posterior probability and the prior probability of the existence of an link is smaller for LinkMirage than the state-of-the-art methods.
- LinkMirage achieves higher *indistinguishability* from an information theoretic perspective (will be defined in Section V-C) where the obfuscated graph of LinkMirage contains less information of users' social relationships than the state-of-the-art methods.
- LinkMirage also shows significant privacy advantages in *anti-aggregation privacy* (will be defined in Section V-D),

where the adversary's estimation for users' social relationships is less accurate for LinkMirage than the existing methods.

C. Utility Evaluation

In Section VI, we apply our perturbation algorithm to various real world applications such as graph analytics, anonymous communications, and Sybil defenses. Compared to previous methods, LinkMirage results in significantly lower attack probabilities when applied to anonymous communications and higher resilience to de-anonymization attacks when applied to vertex anonymity systems. LinkMirage even surprisingly improves the Sybil detection performance when applied to the distributed SybilLimit systems. LinkMirage also outperform existing methods in preserving the utility for multiple graph analytics applications, such as pagerank score and modularity. In Section VII, we further analyze LinkMirage's ability to preserve general graph-theoretic characteristics.

IV. LINKMIRAGE SYSTEM

A. Design Goals

We envision that applications relying on social relationships between users can bootstrap this information from online social network operators such as Facebook, Google+, Twitter with access to the users' social relationships. To enable these applications in a privacy-preserving manner, a perturbed social graph topology (by adding noise to the original graph topology) should be available.

Social graphs evolve over time, and the third-party applications would benefit from access to the most current version of the graph. A baseline approach is to perturb each graph snapshot independently. However, the sequence of perturbed graphs provide significantly more observations to an adversary than just a single perturbed graph. We argue that an effective perturbation method should consider the evolution of the original graph sequence. Therefore, we have the overall design goals for our system as:

1) We aim to obfuscate social relationships while balancing privacy for users' social relationships and the usability for real-world applications.
2) We aim to handle both the static and dynamic social network topologies.
3) Our system should provide rigorous privacy guarantees to defend against adversaries who have prior information of the original graphs, and adversaries who can combine multiple released graphs to infer more information.
4) Our method should be scalable to be applied in real-world large-scale social graphs.

B. LinkMirage: Deployment

To improve the usability of our proposed obfuscation approach (which will be described in detail in Section IV-C), and to avoid dependance on the OSN providers, we developed a Facebook application (available: https://apps.facebook.com/linkmirage/) that implements graph construction (via individual user subscriptions) and obfuscation. The work flow of the LinkMirage deployment is as follows: (i) When a user visits the above URL, Facebook checks the credentials of the user, asks whether to grant the *user's friends* permission, and then gets redirected to the application hosting server. (ii) The application server authenticates itself,

and then queries Facebook for the information of the user's friends, and returns their information such as *user's id*. The list of user's friends can then be collected by the application server to construct a Facebook social graph for the current timestamp. Leveraging LinkMirage, a perturbed graph for this *timestamp* would be available which preserves the link privacy of the users' social relationships.

Real-world systems such as Uproxy, Lantern, Kaleidoscope [17], anonymity systems [11], [28], [29], Sybil defenses systems [8], [43] can directly benefit from our protocol through automatically obtaining the perturbed social relationships. Furthermore, our protocol can enable privacy-preserving graph analytics for OSN providers.

C. LinkMirage: Perturbation Algorithm

Social networks evolve with time and publishing a time series of perturbed graphs raises a serious privacy challenge: an adversary can combine information available from multiple perturbed graphs over time to compromise the privacy of users' social contacts [5], [10], [38]. In LinkMirage, we take a time series of graph topologies into consideration, to account for the evolution of the social networks. Intuitively, the scenario with a static graph topology is just a special situation of the temporal graph sequence, and is thus inherently incorporated in our model.

Consider a social graph series $G_0 = (V_0, E_0), \cdots, G_T = (V_T, E_T)$. We want to transform the graph series to $G'_0 = (V_0, E'_0), \cdots, G'_T = (V_T, E'_T)$, such that the vertices in G'_t remain the same as in the original graph G_t, but the edges are perturbed to protect link privacy. Moreover, while perturbing the current graph G_t, LinkMirage has access to the past graphs in the time series (i.e., $G_0, \cdots, G_{t-1}$). Our perturbation goal is to balance the utility of social graph topologies and the privacy of users' social contacts, across time.

Approach Overview: Our perturbation mechanism for LinkMirage is illustrated in Fig. 2.

Static scenario: For a static graph G_{t-1}, we first cluster it into several communities, and then perturb the links within each community. The inter-cluster links are also perturbed to protect their privacy.

Dynamic scenario: Let us suppose that G_t evolves from G_{t-1} by addition of new vertices (shown in blue color). To perturb graph G_t, our intuition is to consider the similarity between graphs G_{t-1} and G_t.

First, we partition G_{t-1} and G_t into subgraphs, by clustering each graph into different communities. To avoid randomness (guarantee consistency) in the clustering procedure and to reduce the computation complexity, we dynamically cluster the two graphs *together* instead of clustering them independently. Noting that one green node evolves by connecting with a new blue node, we free [1] all the nodes located within $m = 2$ hops of this green node (the other two green nodes and one red node) and merge the remaining three red nodes to a big virtual node. Then, we cluster these new nodes, the freed nodes and the remaining virtual node to detect communities in G_t.

Next, we compare the communities within G_{t-1} and G_t, and identify the *changed* and *unchanged* subgraphs. For the *unchanged* subgraphs C_1, C_2, we set their perturbation at time t to be identical to their perturbation at time $t - 1$,

---

[1] We free the nodes from the previously clustering hierarchy.
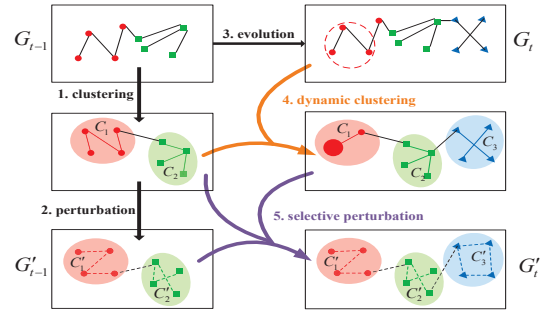

Fig. 2. Our perturbation mechanism for G_t. Assume that G_{t-1} has already been dynamically obfuscated, based on dynamic clustering (step 1) and selective perturbation (step 2). Our mechanism analyzes the evolved graph G_t (step 3) and dynamically clusters G_t (step 4) based on the freed m hop neighborhood ($m = 2$) of new links (between green and blue nodes), the merging virtual node (the large red node in step 4), and the new nodes. By comparing the communities in G_{t-1} and G_t, we can implement selective perturbation (step 5), i.e. perturb the changed blue community independently and perturb the unchanged red and green communities in the same way as G'_{t-1}, and then perturb the inter-cluster links.

denoted by C'_1, C'_2. For the *changed* subgraph C_3, we perturb it independently to obtain C'_3. We also perturb the links between communities to protect privacy of these inter-cluster links. Finally, we publish G'_t as the combination of C'_1, C'_2, C'_3 and the perturbed inter-cluster links. There are two key steps in our algorithm: dynamic clustering and selective perturbation, which we describe in detail as follows.

1) Dynamic Clustering: Considering that communities in social networks change significantly over time, we need to address the inconsistency problem by developing a dynamic community detection method. Dynamic clustering aims to find community structures in evolving graphs by simultaneously considering consecutive graphs in its clustering algorithms. There are several methods in the literature to cluster evolving graphs [3], but we found them to be unsuitable for use in our perturbation mechanism. One approach to dynamic clustering involves performing community detection at each timestamp independently, and then establishing relationships between communities to track their evolution [3]. We found that this approach suffers from performance issues induced by inherent randomness in clustering algorithms, in addition to the increased computational complexity.

Another approach is to combine multiple graphs into a single coupled graph [3]. The coupled graph is constructed by adding edges between the same nodes across different graphs. Clustering can be performed on the single coupled graph. We found that the clustering performance is very sensitive to the weights of the added links, resulting in unstable clustering results. Furthermore, the large dimensionality of the coupled graph significantly increases the computational overhead.

For our perturbation mechanism, we develop an adaptive dynamic clustering approach for clustering the graph G_t using the clustering result for the previous graph G_{t-1}. This enables our perturbation mechanism to (a) exploit the link correlation/similarity in consecutive graph snapshots, and (b) reduce computation complexity by avoiding repeated clustering for unchanged links.

Clustering the graph G_t from the clustering result of the previous graph G_{t-1} requires a *backtracking* strategy. We use the maximum-modularity method [31] for clustering, which is

TABLE I. Temporal Statistics of the Facebook Dataset.

Time	0	1	2	3	4	5	6	7	8
# of nodes	9,586	9,719	11,649	13,848	14,210	16,344	18,974	26,220	35,048
# of edges	48,966	38,058	47,024	54,787	49,744	58,099	65,604	97,095	142,274
Average degree	5.11	3.91	4.03	3.96	3.50	3.55	3.46	3.70	4.06

hierarchical and thus easy to backtrack. Our backtrack strategy is to first maintain a history of the merge operations that led to the current clustering. When an evolution occurs, the algorithm backtracks over the history of merge operations, in order to incorporate the new additions and deletions in the graph.

More concretely, if the link between node x and node y is changed (added or deleted), we omit all the m-hop neighborhoods of x and y as well as x and y themselves from the clustering result of the previous timestamp, and then perform re-clustering. All the new nodes, the changed nodes and their m-hop neighbors, and the remaining merged nodes in the previous clustering result would be considered as basic elements for clustering G_t (recall Figure 2).

For efficient implementation, we store the intermediate results of the hierarchical clustering process in a data structure. Upon link changes between x, y, we free the m-hop neighborhood of x, y from the stored data structure.

2) Selective perturbation:
Intra-cluster Perturbation: After clustering G_t based on G_{t-1} using our dynamic clustering method, we perturb G_t based on G_{t-1} and the perturbed G'_{t-1}. First, we compare the communities detected in G_{t-1} and G_t, and classify them as *changed* or *unchanged*. Our unchanged classification does not require that the communities are exactly the same, but that the overlap among vertices/links exceeds a threshold. Our key idea is to keep the perturbation process for links in the *unchanged* communities to be identical to their perturbation in the previous snapshot. In this manner, we can preserve the privacy of these unchanged links to the largest extent; it is easy to see that alternate approaches would leak more information. For the communities which are classified as changed, our approach is to perturb their links independently of the perturbation in the previous timestamp. For independent perturbations, we leverage the static perturbation method of Mittal et al. in [27]. Their static perturbation deletes all the edges in the original graph, and replaces each edge (v, u) with a fake edge (v, w) selected from the k-hop random walk starting from v. Larger perturbation parameter k corresponds to better privacy and leads to worse utility.

Inter-cluster Perturbation: Finally, we need to interconnect the subgraphs identified above. Suppose that $|v_a|$ nodes and $|v_b|$ nodes are connecting communities a and b respectively, and they construct an inter-community subgraph. For each marginal node $v_a(i) \in v_a$ and $v_b(j) \in v_b$ (here the marginal node in community a (resp.b) refers to the node that has neighbors in the other community b (resp.a)) , we randomly connect them with probability $\frac{\deg(v_a(i)) \deg(v_b(j))|v_a|}{|E_{ab}|(|v_a|+|v_b|)}$.[2] Here, all the computations for $\deg(\cdot), |v.(\cdot)|, |E.|$ only consider the marginal nodes. We can combine the perturbed links corresponding to the unchanged communities, changed communities, and inter-community subgraphs, to compute the output of our algorithm,

---

[2]This probability is set for the preservation of degree distributions as analyzed in Section VII.

Algorithm 1 LinkMirage, with dynamic clustering (steps 1-2) and selective perturbation (steps 3-6). The parameter k denotes the perturbation level for each community. Here, *ch, un, in* are short for *changed, unchanged, intercommunity*, respectively.

Input: $\{G_t, G_{t-1}, G'_{t-1}\}$ if $t \geq 1$ or $\{G_t\}$ if $t = 0$;
Output: G'_t;
  $G'_t, C_t =$ null;
  if t=0;
    cluster G_0 to get C_0;
    label C_0 as changed, i.e. $C_{0-\mathrm{ch}} = C_0$;
  endif
  /*Begin Dynamic Clustering*/
  1. free the nodes within m hops of the changed links;
  2. re-cluster the new nodes, the freed nodes, the remaining merged virtual nodes in $C_{(t-1)}$ to get C_t;
  /*End Dynamic Clustering*/
  /*Begin Selective Perturbation*/
  3. find the unchanged communities $C_{t-\mathrm{un}}$ and the changed communities $C_{t-\mathrm{ch}}$;
  4. let $G'_{t-\mathrm{un}} = G'_{(t-1)-\mathrm{un}}$;
  5. perturb $C_{t-\mathrm{ch}}$ for $G'_{t-\mathrm{ch}}$ by the static method;
  6. **foreach** community pair a and b;
    if both of the communities belong to $C_{t-\mathrm{un}}$
      $C'_{t-\mathrm{in}}(a, b) = C'_{(t-1)-\mathrm{in}}(a, b)$;
    else
      foreach marginal node $v_a(i)$ in a and $v_b(j)$ in b randomly add an edge $(v_a(i), v_b(j))$ with probability $\frac{\deg(v_a(i)) \deg(v_b(j))|v_a|}{|E_{ab}|(|v_a|+|v_b|)}$ to $G'_{t-\mathrm{in}}(a, b)$;
  /*End Selective Perturbation*/
  return $G'_t = [G'_{t-\mathrm{ch}}, G'_{t-\mathrm{un}}, G'_{t-\mathrm{in}}]$;

i.e., G'_t.

LinkMirage not only preserves the structural characteristics of the original graph series, but also protects the privacy of the users by randomizing the original links. As compared to prior work, our method provides stronger privacy and utility guarantees for evolving graphs. Detailed procedures are stated in Algorithm. 1.

LinkMirage improves the state-of-the-art methods such as [27] by incorporating the temporal graph topology. Surprisingly, our approach of first isolating communities and then selectively perturbing them provides benefits even in a static context! This is because previous static approaches use a single parameter to control the privacy/utility trade-off. Thus, if we apply them to the whole graph using high privacy parameters, it would destroy graph utility (e.g. community structures). On the other hand, LinkMirage applies perturbations selectively to communities; thus it is possible to use a very high privacy parameter in the perturbation process, while preserving structural properties such as community structures.
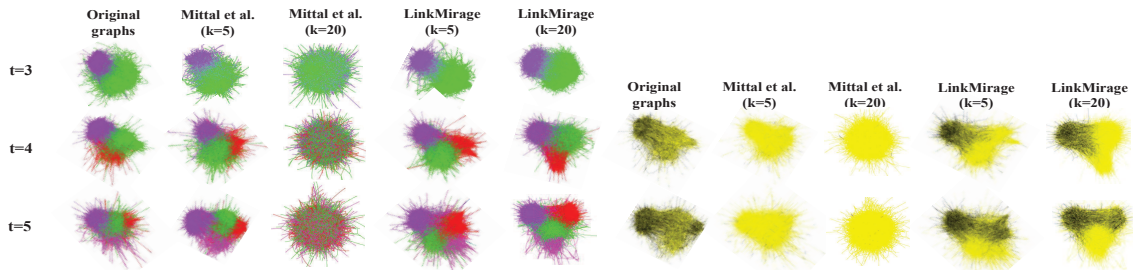
5

Fig. 3. Dynamic Facebook interaction dataset topology, for $t = 3, 4, 5$. On the left, we can see that LinkMirage has superior utility than the baseline approach (Mittal et al.), especially for larger values of k (due to dynamic clustering). On the right, we show the overlapped edges (black) and the changed edges (yellow) between consecutive graphs: t=(3, 4) and t=(4, 5). We can see that in LinkMirage, the perturbation of unchanged communities is correlated across time (selective perturbation), minimizing information leakage and enhancing privacy.

D. Scalable Implementation

Our algorithm relies on two key graph theoretical techniques: *community detection* (serves as a foundation for the dynamic clustering step in LinkMirage) and *random walk* (serves as a foundation for the selective perturbation step in LinkMirage). The computational complexity for both *community detection* and *random walk* is $O(|E_t|)$ [3], [27] where $|E_t|$ is the number of edges in graph G_t, therefore the overall computational complexity of our approach is $O(|E_t|)$. Furthermore, our algorithms are parallelizable. We adopt the GraphChi parallel framework in [22] to implement our algorithm efficiently using a commodity workstation (3.6 GHz, 24GB RAM). Our parallel implementation scales to very large social networks; for example, the running time of LinkMirage is less than 100 seconds for the large scale Google+ dataset (940 million links) (will be described in Section V-A) using our commodity workstation.

E. Visual Depiction

For our experiments, we consider a real world *Facebook social network* dataset [40] among *New Orleans regional network*, spanning from September 2006 to January 2009. Here, we utilize the wall post interaction data which represents stronger trust relationships and comprises of 46,952 nodes (users) connected by 876,993 edges. We partitioned the dataset using three month intervals to construct a total of 9 graph instances as shown in Table I. Fig. 3 depicts the outcome of our perturbation algorithm on the partitioned Facebook graph sequence with timestamp $t = 3, 4, 5$ (out of 9 snapshots), for varying perturbation parameter k (perturbation parameter for each community). For comparative analysis, we consider a baseline approach [27] that applies static perturbation for each timestamp independently. In the dynamic clustering step of our experiments, we free the two-hop neighborhoods of the changed nodes, i.e. $m = 2$.

The maximum-modularity clustering method yields two communities for G_3, three communities for G_4, and four communities for G_5. For the perturbed graphs, we use the same color for the vertices as in the original graph and we can see that fine-grained structures (related to utility) are preserved for both algorithms under small perturbation parameter k, even though links are randomized. Even for high values of k, LinkMirage can preserve the macro-level (such as community-level) structural characteristics of the graph. On the other hand, for high values of k, the static perturbation algorithm results in the loss of structure properties, and appears

to resemble a random graph. Thus, our approach of first isolating communities and applying perturbation at the level of communities has benefits even in a static context.

Fig. 3 also shows the privacy benefits of our perturbation algorithm for timestamps $t = 4, 5$. We can see that LinkMirage reuses perturbed links (shown as black unchanged links) in the unchanged communities (one unchanged community for $t = 4$ and two unchanged communities for $t = 5$). Therefore, LinkMirage preserves the privacy of users' social relationships by considering correlations among the graph sequence, and this benefit does not come at the cost of utility. In the following sections, we will formally quantify the privacy and utility properties of LinkMirage.

V. PRIVACY ANALYSIS

We now address the question of understanding link privacy of LinkMirage. We propose three privacy metrics: *anti-inference privacy, indistinguishability, anti-aggregation privacy* to evaluate the link privacy provided by LinkMirage. Both theoretical analysis and experimental results with a *Facebook* dataset (870K links) and a large-scale *Google+* dataset (940M links) show the benefits of LinkMirage over previous approaches. We also illustrate the relationship between our privacy metric and differential privacy.

A. Experimental Datasets

To illustrate how the temporal information degrades privacy, we consider two social network datasets. The first one is a large-scale Google+ dataset [14]. whose temporal statistics are illustrated in Table II. To the best of our knowledge, this is the largest temporal dataset of social networks in public domain. The Google+ dataset is crawled from July 2011 to October 2011 which has 28,942,911 nodes and 947,776,172 edges. The dataset only considers link additions, i.e. all the edges in the previous graphs exist in the current graph. We partitioned the dataset into 84 timestamps. The second one is the 9-timestamp Facebook wall posts dataset [40] as we stated in Section IV-E. with temporal characteristics shown in Table I. It is worth noting that the wall-posts data experiences tremendous churn with only 45% overlap for consecutive graphs. Since our dynamic perturbation method relies on the correlation between consecutive graphs, the evaluation of our dynamic method on the Facebook wall posts data is conservative. To show the improvement in performance of our algorithm for graphs that evolve at a slower rate, we also consider a sampled graph sequence extracted from the Facebook wall posts data with 80% overlap for consecutive graphs.

TABLE II. Temporal Statistics of the Google+ Dataset.

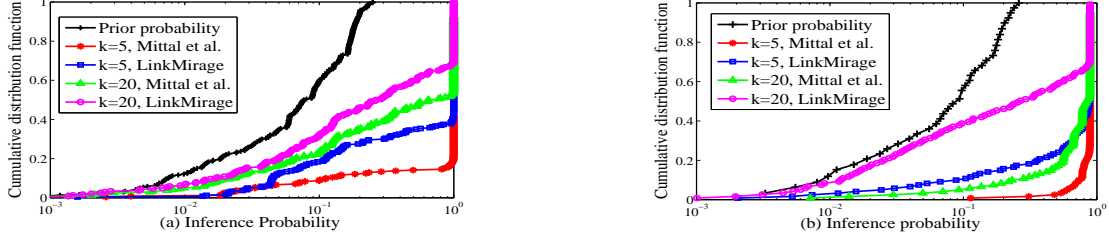| Time	Jul.29	Aug.8	Aug.18	Aug.28	Sep.7	Sep.17	Sep.27	Oct.7
# of nodes	16,165,781	17,483,936	17,850,948	19,406,327	19,954,197	24,235,387	28,035,472	28,942,911
# of edges	505,527,124	560,576,194	575,345,552	654,523,658	686,709,660	759,226,300	886,082,314	947,776,172
Average degree	31.2714	32.0624	32.2305	33.7273	34.4143	31.3272	31.6058	32.7464

Fig. 4. (a),(b) represent the link probability distributions for the whole Facebook interaction dataset and the sampled Facebook interaction dataset with 80% overlap. We can see that the posterior probability of LinkMirage is more similar to the prior probability than the baseline approach.

B. Anti-Inference Privacy

First, we consider adversaries that aim to infer link information by leveraging Bayesian inference. We define the privacy of a link L_t (or a subgraph) in the t-th graph instance, as the difference between the posterior probability and the prior probability of the existence of the link (or a subgraph), computed by the adversary using its prior information W, and the knowledge of the perturbed graph sequence $\{G'_i\}_{i=0}^t$. Utilizing Bayesian inference, we have

Definition 1: For link L_t in the original graph sequence $G_0, \cdots, G_t$ and the adversary's prior information W, the anti-inference privacy Privacy_{ai} for the perturbed graph sequence $G'_0, \cdots, G'_t$ is evaluated by the similarity between the posterior probability $P(L_t|\{G'_i\}_{i=0}^t, W)$ and the prior probability $P(L_t|W)$, where the posterior probability is

$$P(L_t|\{G'_i\}_{i=0}^t, W) = \frac{P(\{G'_i\}_{i=0}^t|L_t, W) \times P(L_t|W)}{P(\{G'_i\}_{i=0}^t|W)} \quad (1)$$

Higher similarity implies better anti-inference privacy.

The difference between the posterior probability and the prior probability represents the information leaked by the perturbation mechanism. Similar intuition has been mentioned in [23]. Therefore, the posterior probability should not differ much from the prior probability.

In the above expression, $P(L_t|W)$ is the prior probability of the link, which can be computed based on the known structural properties of social networks, for example, by using link prediction algorithms [24]. Note that $P(\{G'_i\}_{i=0}^t|W)$ is a normalization constant that can be analyzed by sampling techniques. The key challenge is to compute $P(\{G'_i\}_{i=0}^t|L_t, W)$[3].

For evaluation, we consider a special case where the adversary's prior is the entire time series of original graphs except the link L_t (which is the link we want to quantify privacy for, and $L_t = 1$ denotes the existence of this link while $L_t = 0$ denotes the non-existence of this link). Such prior information can be extracted from personal public information, Facebook related information or other application-related information as

---

[3]The detailed process for computing the posterior probability can be found in [27].
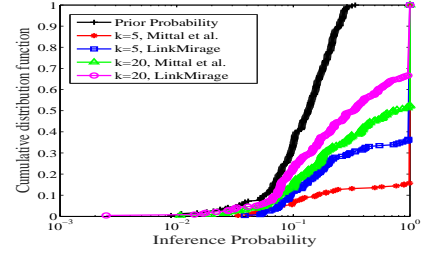

Fig. 5. Link probability distribution for the Google+ dataset under the adversary's prior information extracted from the social-attribute network model in [14].

stated in [6]. Note that this is a very strong adversarial prior, which would lead to the worst-case analysis of link privacy. Denoting $\{\widetilde{G}_i(L_t)\}_{i=0}^t$ as the prior which contains all the information except L_t, we have the posterior probability of link L_t under the worst case is

$$P(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t)$$
$$= \frac{P(\{G'_i\}_{i=0}^t|, L_t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \times P(L_t|\{\widetilde{G}_i(L_t)\}_{i=0}^t)}{P(\{G'_i\}_{i=0}^t|\{\widetilde{G}_i(L_t)\}_{i=0}^t)}$$

where

$$P(\{G'_i\}_{i=0}^t|L_t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) = P(G'_0|\widetilde{G}_0(L_t)) \times$$
$$P(G'_1|G'_0, \widetilde{G}_0(L_t), \widetilde{G}_1(L_t)) \cdots P(G'_t|G'_{t-1}, \widetilde{G}_{t-1}(L_t), \widetilde{G}_t(L_t))$$

Therefore, the objective of perturbation algorithms is to make $P(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t)$ close to $P(L_t|\{\widetilde{G}_i(L_t)\}_{i=0}^t)$.

Comparison with previous work: Fig. 4 shows the posterior probability distribution for the whole Facebook graph sequence and the sampled Facebook graph sequence with 80% overlapping ratio, respectively. We computed the prior probability using the link prediction method in [24]. We can see that the posterior probability corresponding to LinkMirage is closer to the prior probability than that of the method of Mittal et al. [27]. In Fig. 4(b), taking the point where the link probability equals 0.1, the distance between the posterior CDF and the prior CDF for the static approach is a factor of 3 larger than LinkMirage ($k = 20$). Larger perturbation degree k improves privacy and leads to smaller difference with the prior probability. Finally, by comparing Fig. 4(a) and (b), we can see that larger overlap in the graph sequence improves the
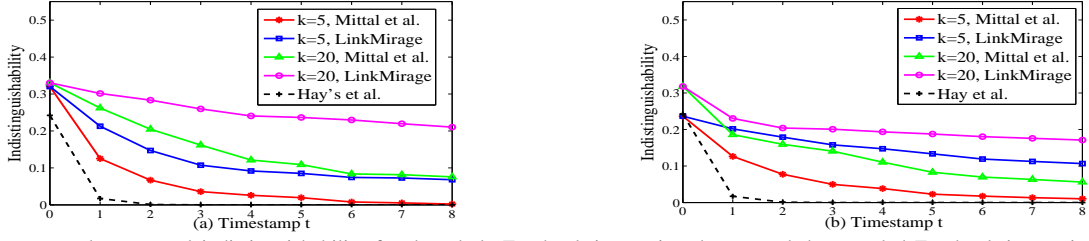
Fig. 6. (a),(b) represent the temporal indistinguishability for the whole Facebook interaction dataset and the sampled Facebook interaction dataset with 80% overlap. Over time, the adversary has more information, resulting in decreased indistinguishability. We can also see that LinkMirage has higher indistinguishability than the static method and the Hay's method in [16], although it still suffers from some information leakage.

privacy benefits of LinkMirage. We also compare with the work of Hay et al. in [16], which randomizes the graph with r real links deleted and another r fake links introduced. The probability for a real link to be preserved in the perturbed graph is $1 - r/m$, which should not be small otherwise the utility would not be preserved. Even considering $r/m = 0.5$ (which would substantially hurt utility [16]), the posterior probability for a link using the method of Hay et al. would be 0.5, *even without prior information*. In contrast, our analysis for LinkMirage considers a worst-case prior, and shows that the posterior probability is smaller than 0.5 for more than 50% of the links when $k = 20$ in Fig. 4. Therefore, our LinkMirage provides significantly higher privacy than the work of Hay et al.

Adversaries with structural and contextual information: Note that our analysis so far focuses on quantifying link-privacy under an adversary with prior information about the original network structure (including link prediction capabilities). In addition, some adversaries may also have access to contextual information about users in the social network, such as user attributes, which can also be used to predict network links (e.g., social-attribute network prediction model in [14]). We further computed the prior probability using such social-attribute network prediction model in [14] and showed the link probability for the Google+ dataset in Fig. 5. The posterior probability of our LinkMirage is closer to the prior probability and thus LinkMirage achieves better privacy performance than previous work.

C. Indistinguishability

Based on the posterior probability of a link under the worst case $P(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t)$, we need to qualify the privacy metric for adversaries who aim to distinguish the posterior probability with the prior probability. Since our goal is to reduce the information leakage of L_t based on the perturbed graphs $\{G'_i\}_{i=0}^t$ and the prior knowledge $\{\widetilde{G}_i(L_t)\}_{i=0}^t$, we consider the metric of *indistinguishability* to quantify privacy, which can be evaluated by the conditional entropy of a private message given the observed variables [7]. The objective for an obfuscation scheme is to maximize the indistinguishability of the unknown input I given the observables O, i.e. $H(I|O)$ (where H denotes entropy of a variable [7]). Here, we define our metric for link privacy as

Definition 2: The *indistinguishability* for a link L_t in the original graph G_t that the adversary can infer from the perturbed graph G'_t under the adversary's prior information $\{\widetilde{G}_i(L_t)\}_{i=0}^t$ is defined as $\text{Privacy}_{\text{id}} =$

$H(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t)$.

Furthermore, we quantify the behavior of indistinguishability over time. For our analysis, we continue to consider the worst case prior of the adversary knowing the entire graph sequence except the link L_t. To make the analysis tractable, we add another condition that if the link L exists, then it exists in all the graphs (link deletions are rare in real world social networks). For a large-scale graph, only one link would not affect the clustering result. Then, we have

Theorem 1: The *indistinguishability* decreases with time,

$$H(L|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L)\}_{i=0}^t) \geq H(L|\{G'_i\}_{i=0}^{t+1}, \{\widetilde{G}_i(L)\}_{i=0}^{t+1}) \tag{2}$$

The inequality follows from the theorem *conditioning reduces entropy* in [7]. Eq.2 shows that the *indistinguishability* would not increase as time evolves. The reason is that over time, multiple perturbed graphs can be used by the adversary to infer more information about link L.

Next, we theoretically show why LinkMirage has better privacy performance than the static method. For each graph G_t, denote the perturbed graphs using LinkMirage and the static method as $G'_t, G'^{,s}_t$, respectively.

Theorem 2: The *indistinguishability* for LinkMirage is greater than that for the static perturbation method, i.e.

$$H(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \geq H(L_t|\{G'^{,s}_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \tag{3}$$

Proof: In LinkMirage, the perturbation for the current graph G_t is based on perturbation for G_{t-1}. Let us denote the *changed* subgraph between G_{t-1}, G_t as $G_{t-\text{ch}}$, then

$$
\begin{aligned}
&H(L_t|\{G'_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \\
=&H(L_t|\{G'_i\}_{i=0}^{t-2}, G'_{t-1}, G'_t - G'_{t-\text{ch}}, G'^{,s}_{t-\text{ch}}, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \\
=&H(L_t|\{G'_i\}_{i=0}^{t-1}, G'^{,s}_{t-\text{ch}}, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \\
\geq&H(L_t|\{G'_i\}_{i=0}^{t-1}, G'^{,s}_t, \{\widetilde{G}_i(L_t)\}_{i=0}^t) \\
\geq&H(L_t|\{G'^{,s}_i\}_{i=0}^t, \{\widetilde{G}_i(L_t)\}_{i=0}^t)
\end{aligned}
$$

where the first inequality also comes from the theorem *conditioning reduces entropy* in [7]. The second inequality generalizes the first inequality from a snapshot t to the entire sequence. From Eq.3, we can see that LinkMirage may offer superior *indistinguishability* compared to the static perturbation, and thus provides higher privacy. ∎

Comparison with previous work: Next, we experimentally analyze our *indistinguishability* metric over time. Fig. 6 depicts
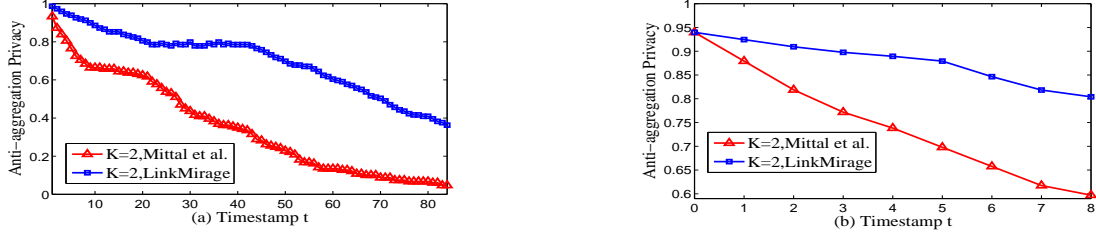
Fig. 7. (a)(b) show the temporal anti-aggregation privacy for the Google+ dataset and the Facebook dataset, respectively. The anti-aggregation privacy decreases as time evolves because more information is leaked with more perturbed graphs available. Leveraging selective perturbation, LinkMirage achieves much better anti-aggregation privacy than the static baseline method.

the *indistinguishability* metric using the whole Facebook graph sequence and the sampled Facebook graph sequence with 80% overlap. We can see that the static perturbation leaks more information over time. In contrast, the selective perturbation achieves significantly higher *indistinguishability*. In Fig. 6(a), after 9 snapshots, and using $k = 5$, the *indistinguishability* of the static perturbation method is roughly $1/10$ of the indistinguishability of LinkMirage. This is because selective perturbation explicitly takes the temporal evolution into consideration, and stems privacy degradation via the selective perturbation step. Comparing Fig. 6(a) and (b), LinkMirage has more advantages for larger overlapped graph sequence.

We also compare with the work of Hay et al. in [16], For the first timestamp, the probability for a real link to be preserved in the anonymized graph is $1 - r/m$. As time evolves, the probability would decrease to $(1 - r/m)^t$. Combined with the prior probability, the corresponding *indistinguishability* for the method of Hay et al. is shown as the black dotted line in Fig. 6, which converges to 0 very quickly (we also consider $r/m = 0.5$ which would substantially hurt utility [16]) Compared with the work of Hay et al, LinkMirage significantly improves privacy performance. Even when $t = 1$, LinkMirage with $k = 20$ achieves up to 10x improvement over the approach of Hay et al. in the *indistinguishability* performance.

D. Anti-aggregation Privacy

Next, we consider the adversaries who try to aggregate all the previously published graphs to infer more information. Recall that after community detection in our algorithm, we anonymize the links by leveraging the k-hop random walk. Therefore, the perturbed graph G' is actually a sampling of the k-hop graph G^k, where the k-hop graph G^k represents graph where all the k-hop neighbors in the original graph are connected. It is intuitive that a larger difference between G^k and G' represents better privacy. Here, we utilize the distance between the corresponding transition probability matrices $\|P_t^k - P_t'\|_{\text{TV}}$ [4] to measure this difference. And we extend the definition of total variance [18] from vector to matrix by averaging total variance distance of each row in the matrix, i.e. $\|P_t^k - P_t'\|_{\text{TV}} = \frac{1}{|V_t|}\sum_{v=1}^{|V_t|} \|P_t^k(v) - P_t'(v)\|_{\text{TV}}$, where $P_t^k(v), P_t'(v)$ denotes the v-th row of P_t^k, P_t'. We then formally define the anti-aggregation privacy as

Definition 3: The anti-aggregation privacy for a perturbed

graph G_t' with respect to the original graph G_t and the perturbation parameter k is $\text{Privacy}_{\text{aa}}(G_t, G_t', k) = \|P_t^k - P_t'\|_{\text{TV}}$.

The adversary's final objective is to obtain an estimated measurement of the original graph, e.g. the estimated transition probability matrix $\hat{P}_t$ which satisfies $\hat{P}_t^k = P_t'$. A straightforward manner to evaluate privacy is to compute the estimation error of the transition probability matrix i.e. $\|P_t - \hat{P}_t\|_{\text{TV}}$. We can derive the relationship between the anti-aggregation privacy and the estimation error as (we defer the proofs to the Appendix to improve readability.)

Theorem 3: The anti-aggregation privacy is a lower bound of the estimation error for the adversaries, and

$$\|P_t^k - P_t'\|_{\text{TV}} \le k\|P_t - \hat{P}_t\|_{\text{TV}} \tag{4}$$

We further consider the network evolution where the adversary can combine all the perviously perturbed graphs together to extract more k-hop information of the current graph. Under this situation, a strategic methodology for the adversary is to combine the perturbed graph series $G_0', \cdots, G_t'$, to construct a new perturbed graph $\breve{G}_t'$, where $\breve{G}_t' = \bigcup_{i=0,1\cdots,t} G_i'$. The combined perturbed graph $\breve{G}_t'$ contains more information about the k-hop graph $G_t'^k$ than G_t'. Correspondingly, the transition probability matrix $\breve{P}_t'$ of the combined perturbed graph $\breve{G}_t'$ would provide more information than P_t'. That is to say, the *anti-aggregation privacy* decreases with time.

Comparison with previous work: We evaluate the anti-aggregation privacy of LinkMirage on both the Google+ dataset and the Facebook dataset. Here we perform our experiments based on a conservative assumption that a link always exists after it is introduced. The anti-aggregation privacy decreases with time since more information about the k-hop neighbors of the graph is leaked as shown in Fig. 7. Our selective perturbation preserves correlation between consecutive graphs, therefore leaks less information and achieves better privacy than the static baseline method. For the Google+ dataset, the anti-aggregation privacy for the method of Mittal et al. is only $1/10$ of LinkMirage after 84 timestamps.

E. Relationship with Differential Privacy

Our anti-inference privacy analysis considers the worst-case adversarial prior to infer the existence of a link in the graph. Next, we uncover a novel relationship between this anti-inference privacy and differential privacy.

---

[4]We choose the total variance distance to evaluate the statistical distance between P_t^k and P_t' as in [27].
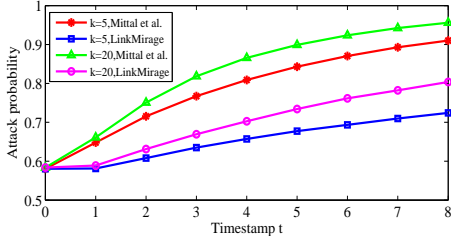
Fig. 8. The worst case probability of deanonymizing users' communications ($f = 0.1$). Over time, LinkMirage provides better anonymity compared to the static approaches.

Differential privacy is a popular theory to evaluate the privacy of a perturbation scheme [12], [13]. The framework of differential privacy defines *local sensitivity* of a query function f on a dataset D_1 as the maximal $|f(D_1) - f(D_2)|_1$ for all D_2 differing from D_1 in at most one element $df = \max_{D_2} \|f(D_1) - f(D_2)\|_1$. Based on the theory of differential privacy, a mechanism that adds independent Laplacian noise with parameter df/ϵ to the query function f, satisfies ϵ-differential privacy. The degree of added noise, which determines the utility of the mechanism, depends on the local sensitivity. To achieve a good utility as well as privacy, the local sensitivity df should be as small as possible. The following lemma demonstrates the effectiveness of our worst-case Bayesian analysis by showing that the objective for good utility-privacy balance under our worst-case Bayesian analysis is equivalent to that under differential privacy.

Remark 1: The requirement for good utility-privacy balance in differential privacy is equivalent to the objective of our Bayesian analysis under the worst case. (We defer the proofs to Appendix to improve readability.)

F. Summary for Privacy Analysis

- LinkMirage provides rigorous privacy guarantees to defend against adversaries who have prior information about the original graphs, and the adversaries who aim to combine multiple released graphs to infer more information.
- LinkMirage shows significant privacy advantages in *anti-inference privacy, indistinguishability and anti-aggregation privacy*, by outperforming previous methods by a factor up to 10.

VI. APPLICATIONS

Applications such as anonymous communication [11], [28], [29] and vertex anonymity mechanisms [25], [34], [45] can utilize LinkMirage to obtain the entire obfuscated social graphs. Alternatively, each individual user can query LinkMirage for his/her perturbed neighborhoods to set up distributed social relationship based applications such as SybilLimit [43]. Further, the OSN providers can also leverage LinkMirage to perturb the original social topologies only once and support multiple privacy-preserving graph analytics, e.g., privately compute the pagerank/modularity of social networks.

A. Anonymous Communication [11], [28], [29]

As a concrete application, we consider the problem of anonymous communication [11], [28], [29]. Systems for anonymous communication aim to improve user's privacy by hiding the communication link between the user and the remote destination. Nagaraja et al. and others [11], [28], [29] have suggested that the security of anonymity systems can be improved by leveraging users' trusted social contacts.

We envision that our work can be a key enabler for the design of such social network based systems, while preserving the privacy of users' social relationships. We restrict our analysis to low-latency anonymity systems that leverage social links, such as the Pisces protocol [28].

Similar to the Tor protocol, users in Pisces rely on proxy servers and onion routing for anonymous communication. However, the relays involved in the onion routing path are chosen by performing a random walk on a trusted social network topology. Recall that LinkMirage better preserves the evolution of temporal graphs in Fig. 3. We now show that this translates into improved anonymity over time, by performing an analysis of the degradation of user anonymity over multiple graph snapshots. For each graph snapshot, we consider a worst case anonymity analysis as follows: if a user's neighbor in the social topology is malicious, then over multiple communication rounds (within that graph instance) its anonymity will be compromised using state-of-the-art traffic analysis attacks [41]. Now, suppose that all of a user's neighbors in the first graph instance are honest. As the perturbed graph sequence evolves, there is further potential for degradation of user anonymity since in the subsequent instances, there is a chance of the user connecting to a malicious neighbor. Suppose the probability for a node to be malicious is f. Denote $n_t(v)$ as the distinct neighbors of node v at time t. For a temporal graph sequence, the number of the union neighbors $\cup_{k=0}^{t} n_k(v)$ of v increases with time, and the probability for v to be attacked under the worst case is $P_t^{attack}(v) = 1 - (1-f)^{|\cup_{k=0}^{t} n_k(v)|}$. Note that in practice, the adversary's prior information will be significantly less than the worst-case adversary.

Fig. 8 depicts the degradation of the worst-case anonymity with respect to the number of perturbed topologies. We can see that the attack probability for our method is lower than the static approach with a factor up to 2. This is because over consecutive graph instances, the users' social neighborhood has higher similarity as compared to the static approach, reducing potential for anonymity degradation. Therefore, LinkMirage can provide better security for anonymous communication, and other social trust based applications.

B. Vertex Anonymity [25], [34], [45]

Previous work for vertex anonymity [25], [34], [45] would be defeated by de-anonymization techniques [20], [30], [32], [37]. LinkMirage can serve as a formal first step for vertex anonymity, and even improve its defending capability against de-anonymization attacks. We apply LinkMirage to anonymize vertices, i.e. to publish a perturbed topology without labeling any vertex. In [20], Ji et al. modeled the anonymization as a sampling process where the sampling probability p denotes the probability of an edge in the original graph G_o to exist in the anonymized graph G'. LinkMirage can also be applied for
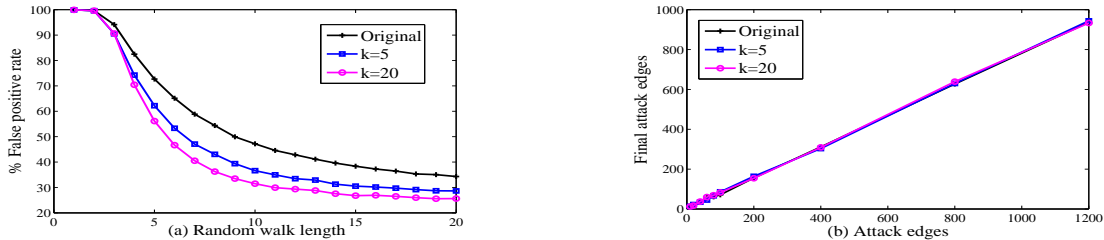
Fig. 9. (a) shows the false positive rate for Sybil defenses. We can see that the perturbed graphs have lower false positive rate than the original graph. Random walk length is proportional to the number of Sybil identities that can be inserted in the system. (b) shows that the final attack edges are roughly the same for the perturbed graphs and the original graphs.

such model, where the perturbed graph G' is sampled from the k-hop graph G^k (corresponding to G_o).

They also derived a theoretical bound of the sampling probability p for perfect de-anonymization, and found that a weaker bound is needed with a larger value of the sampling probability p. Larger p implies that G' is topologically more similar to G, making it easier to enable a perfect de-anonymization. When considering social network evolution, the sampling probability p can be estimated as $|E(G'_0, \cdots, G'_t)|/|E(G^k_0, \cdots, G^k_t)|$, where $E(G'_0, \cdots, G'_t)$ are the edges of the perturbed graph sequence, and $E(G^k_0, \cdots, G^k_t)$ are the edges of the k-hop graph sequence. Compared with the static baseline approach, LinkMirage selectively reuses information from previously perturbed graphs, thus leading to smaller overall sampling probability p, which makes it harder to perfectly de-anonymize the graph sequence. For example, the average sampling probability p for the Google+ dataset (with $k = 2$) is 0.431 and 0.973 for LinkMirage and the static method respectively. For the Facebook temporal dataset (with $k = 3$), the average sampling probability p is 0.00012 and 0.00181 for LinkMirage and the static method respectively. Therefore, LinkMirage is more resilient against de-anonymization attacks even when applied to vertex anonymity, with up to 10x improvement.

C. Sybil Defenses [43]

Next, we consider Sybil defenses systems which leverage the published social topologies to detect fake accounts in the social networks. Here, we analyze how the use of a perturbed graph changes the Sybil detection performance of SybilLimit [43], which is a representative Sybil defense system. Each user can query LinkMirage for his/her perturbed friends to set up the implementation of SybilLimit. Fig. 9(a) depicts the false positives (honest users misclassified as Sybils) with respect to the random walk length in the Sybillimit protocol. Fig. 9(b) shows the final attack edges with respect to the attack edges in the original topology. We can see that the false positive rate is much lower for the perturbed graphs than for the original graph, while the number of the attack edges stay roughly the same for the original graph and the perturbed graphs. The number of Sybil identities that an adversary can insert is given by $S = g' \cdot w'$ (g' is the number of attack edges and w' is the random walk parameter in the protocol). Since g' stays almost invariant and the random walk parameter w' (for any desired false positive rate) is reduced, LinkMirage improves Sybil resilience and provides the privacy of the social relationships such that Sybil defense protocols continue to be applicable (similar to static approaches whose Sybil-resilience

TABLE III. Modularity of Perturbed Graph Topologies

Google+	Original Graph	LinkMirage $k = 2$	LinkMirage $k = 5$	Mittal et al. $k = 2$	Mittal et al. $k = 5$
Modularity	0.605	0.601	0.603	0.591	0.586
Facebook	Original Graph	LinkMirage $k = 5$	LinkMirage $k = 20$	Mittal et al. $k = 5$	Mittal et al. $k = 20$
Modularity	0.488	0.479	0.487	0.476	0.415

performance have been demonstrated in previous work).

D. Privacy-preserving Graph Analytics [31], [33]

Next, we demonstrate that LinkMirage can also benefit the OSN providers for privacy-preserving graph analytics. Previous work in [12], [13] have demonstrated that the implementation of graph analytic algorithms would also result in information leakage. To mitigate such privacy degradation, the OSN providers could add perturbations (noises) to the outputs of these graph analytics. However, if the OSN providers aim to implement multiple graph analytics, the process for adding perturbations to each output would be rather complicated. Instead, the OSN providers can first obtain the perturbed graph by leveraging LinkMirage and then set up these graph analytics in a privacy-preserving manner.

Here, we first consider the pagerank [33] as an effective graph metric. For the Facebook dataset, we have the average differences between the perturbed pagerank score and the original pagerank score as 0.0016 and 0.0018 for $k = 5$ and $k = 20$ respectively in LinkMirage. In comparison, the average differences are 0.0019 and 0.0087 for $k = 5$ and $k = 20$ in the approach of Mittal et al. LinkMirage preserves the pagerank score of the original graph with up to 4x improvement over previous methods. Next, we show the modularity [31] (computed by the timestamp $t = 3$ in the Google+ dataset and the Facebook dataset, respectively) in Table III. We can see that LinkMirage preserves both the pagerank score and the modularity of the original graph, while the method of Mittal et al. degrades such graph analytics especially for larger perturbation parameter k (recall the visual intuition of LinkMirage in Fig. 3).

E. Summary for Applications of LinkMirage

- LinkMirage preserves the privacy of users' social contacts while enabling the design of social relationships based applications. Compared to previous methods, LinkMirage results in significantly lower attack probabilities (with a

TABLE IV. Graph Metrics of the Original and the Perturbed Graphs for the Google+ Dataset.

	Clustering Coefficient	Assortativity Coefficient
Original Graph	0.2612	-0.0152
LinkMirage $k = 2$	0.2263	-0.0185
LinkMirage $k = 5$	0.1829	-0.0176
LinkMirage $k = 10$	0.0864	-0.0092
LinkMirage $k = 20$	0.0136	-0.0063

factor up to 2) when applied to anonymous communications and higher resilience to de-anonymization attacks (with a factor up to 10) when applied to vertex anonymity systems.

- LinkMirage even surprisingly improves the Sybil detection performance when applied to the distributed SybilLimit systems.
- LinkMirage preserves the utility performance for multiple graph analytics applications, such as pagerank score and modularity with up to 4x improvement.

VII. Utility Analysis

Following the application analysis in Section VI, we aim to develop a general metric to characterize the utility of the perturbed graph topologies. Furthermore, we theoretically analyze the lower bound on utility for LinkMirage, uncover connections between our utility metric and structural properties of the graph sequence, and experimentally analyze our metric using the real-world Google+ and Facebook datasets.

A. Metrics

We aim to formally quantify the utility provided by LinkMirage to encompass a broader range of applications. One intuitive global utility metric is the degree of vertices. It is interesting to find that the expected degree of each node in the perturbed graph is the same as the original degree and we defer the proof to Appendix to improve readability.

Theorem 4: The expected degree of each node after perturbation by LinkMirage is the same as in the original graph: $\forall v \in V_t, \mathbb{E}(\deg'(v)) = \deg(v)$, where $\deg'(v)$ denotes the degree of vertex v in G'_t.

To understand the utility in a fine-grained level, we further define our utility metric as

Definition 4: The Utility Distance (UD) of a perturbed graph sequence $G'_0, \cdots, G'_T$ with respect to the original graph sequence $G_0, \cdots, G_T$, and an application parameter l is defined as

$$\text{UD}(G_0, \cdots G_T, G'_0, \cdots G'_T, l)$$
$$= \frac{1}{T+1} \sum_{t=0}^{T} \sum_{v \in V_t} \frac{1}{|V_t|} \|P_t^l(v) - (P'_t)^l(v)\|_{\text{TV}} \quad (5)$$

Our definition for utility distance in Eq. 5 is intuitively reasonable for a broad class of real-world applications, and captures the behavioral differences of l-hop random walks between the original graphs and the perturbed graphs. We note that random walks are closely linked to the structural properties of social networks. In fact, a lot of social network based security applications such as Sybil defenses [43] and

anonymity systems [28] directly perform random walks in their protocols. The parameter l is application specific; for applications that require access to fine grained local structures, such as recommendation systems [2], the value of l should be small. For other applications that utilize coarse and macro structure of the social graphs, such as Sybil defense mechanisms, l can be set to a larger value (typically around 10 in [43]). Therefore, this utility metric can quantify the utility performance of LinkMirage for various applications in a general manner.

Note that LinkMirage is not limited to only preserving the community structure of the original graphs. We evaluate two representative graph theoretic metrics *clustering coefficient* and *assortativity coefficient* [14] as listed in Table IV. We can see that LinkMirage well preserves such fine-grained structural properties for smaller perturbation parameter k. Therefore, the extent to which the utility properties are preserved depends on the perturbation parameter k.

B. Relationships with Other Graph Structural Properties

The mixing time $\tau_\epsilon(G_t)$ measures the time required for the Markov chain to converge to its stationary distribution, and is defined as $\tau_\epsilon(G_t) = \min_r \max_v(r\||P_t^r(v) - \pi_t|_{\text{TV}} < \epsilon)$. Based on the Perron-Frobenius theory, we denote the eigenvalues of P_t as $1 = \mu_1(G_t) \geq \mu_2(G_t) \geq \cdots \mu_{|V_t|}(G_t) \geq -1$. The convergence rate of the Markov chain to π_t is determined by the second largest eigenvalue modulus (SLEM) as $\mu(G_t) = \max\left(\mu_2(G_t), -\mu_{|V_t|}(G_t)\right)$.

Since our utility distance is defined by using the transition probability matrix P_t, this metric can be proved to be closely related to structural properties of the graphs, as shown in Theorem 5 and Theorem 6.

Theorem 5: Let us denote the utility distance between the perturbed graph G'_t and the original graph G_t by $\text{UD}(G_t, G'_t, l)$, then we have $\tau_{G'_t}\left(\text{UD}\left(G_t, G'_t, \tau_{G_t}(\epsilon)\right) - \epsilon\right) \geq \tau_{G_t}(\epsilon)$.

Theorem 6: Let us denote the second largest eigenvalue modulus (SLEM) of transition probability matrix P_t of graph G_t as μ_{G_t}. We can bound the SLEM of a perturbed graph G'_t using the mixing time of the original graph, and the utility distance between the graphs as $\mu_{G'_t} \geq 1 - \frac{\log n + \log \frac{1}{\text{UD}(G_t, G'_t, \tau_{G_t}(\epsilon)) - \epsilon}}{\tau_{G_t}(\epsilon)}$.

C. Upper Bound of Utility Distance

LinkMirage aims to limit the degradation of link privacy over time. Usually, mechanisms that preserve privacy trade-off application utility. In the following, we will theoretically derive an upper bound on the utility distance for our algorithm. This corresponds to a lower bound on utility that LinkMirage is guaranteed to provide.

Theorem 7: The utility distance of LinkMirage is upper bounded by $2l$ times the sum of the utility distance of each community ϵ and the ratio cut δ_t for each G_t, i.e.

$$\text{UD}(G_0, \cdots G_T, G'_0, \cdots G'_T, l) \leq \frac{1}{T+1} \sum_{t=0}^{T} 2l(\epsilon + \delta_t) \quad (6)$$

where δ_t denotes the number of inter-community links over the number of vertices, and each community $C_{k(t)}$ within G_t satisfies $\|C_{k(t)} - C'_{k(t)}\|_{TV} \leq \epsilon$. We defer the proofs to the
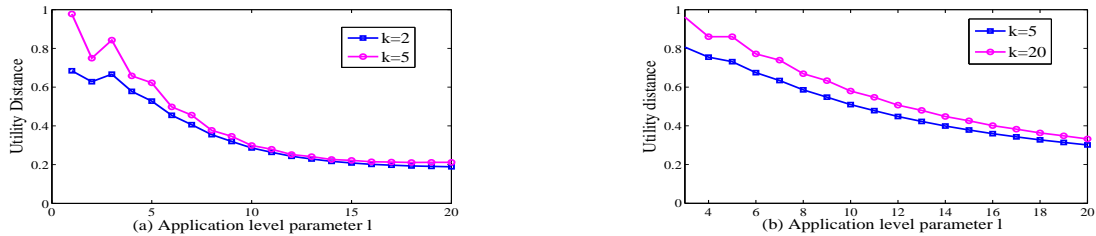
Fig. 10. (a), (b) show the utility distances using the Google+ dataset and the Facebook dataset, respectively. Larger perturbation parameter k results in larger utility distance. Larger application parameter l decreases the distance, which shows the effectiveness of LinkMirage in preserving global community structures.

Appendix to improve readability.

Note that an upper bound on utility distance corresponds to a lower bound on utility of our algorithm. While better privacy usually requires adding more noise to the original sequence to obtain the perturbed sequence, thus we can see that LinkMirage is guaranteed to provide a minimum level of utility performance.

In the derivation process, we do not take specific evolutionary pattern such as the overlapped ratio into consideration, therefore our theoretical upper bound is rather loose. Next, we will show that in practice, LinkMirage achieves smaller utility distance (higher utility) than the baseline approach of independent static perturbations.

D. Utility Experiments Analysis

Fig. 10(a)(b) depict the utility distance for the Google+ and the Facebook graph sequences, for varying perturbation degree k and the application level parameter l. We can also see that as k increases, the distance metric increases. This is natural since additional noise increase the distance between probability distributions computed from the original and the perturbed graph series. As the application parameter l increases, the distance metric decreases. This illustrates that LinkMirage is more suited for security applications that rely on macro structures, as opposed to applications that require exact information about one or two hop neighborhoods. Furthermore, our experimental results in Table III also demonstrate the utility advantage of our LinkMirage over the approach of Mittal et al. [27] in real world applications.

VIII. RELATED WORK

Privacy with labeled vertices An important thread of research aims to preserve link privacy between labeled vertices by obfuscating the edges, i.e., by adding /deleting edges [16], [27], [42]. These methods aim to randomize the structure of the social graph, while differing in the manner of adding noise. Hay et al. [16] perturb the graph by applying a sequence of r edge deletions and r edge insertions. The deleted edges are uniformly selected from the existing edges in the original graph while the added edges are uniformly selected from the non-existing edges. However, neither the edge deletions nor edge insertions take any structural properties of the graph into consideration. Ying and Wu [42] proposed a new perturbation method for preserving spectral properties, without analyzing its privacy performance.

Mittal et al. proposed a perturbation method in [27], which serves as the foundation for our algorithm. Their method deletes *all* edges in the original graph, and replaces each edge with a fake edge that is sampled based on the structural prop-

erties of the graph. In particular, random walks are performed on the original graph to sample fake edges. As compared to the methods of Hay et al. [16] and Mittal et al. [27], LinkMirage provides up to 3x privacy improvement for static social graphs and up to 10x privacy improvement for dynamic social graphs.

Another line of research aims to preserve link privacy [15] [44] by aggregating the vertices and edges into super vertices. Therefore, the privacy of links within each super vertex is naturally protected. However, such approaches do not permit fine grained utilization of graph properties, making it difficult to be applied to applications such as social network based anonymous communication and Sybil defenses.

Privacy with unlabeled vertices While the focus of our paper is on preserving link privacy in context of labeled vertices, an orthogonal line of research aims to provide privacy in the context of unlabeled vertices (vertex privacy) [4], [25], [34]. Liu et al. [25] proposed k-anonymity to anonymize unlabeled vertices by placing at least k vertices at an equivalent level. Differential privacy provides a theoretical framework for perturbing aggregate information, and Sala et al. [34] leveraged differential privacy to privately publish social graphs with unlabeled vertices. We note that LinkMirage can also provide a foundation for preserving vertex privacy as stated in Section VI-B. Shokri et al. [35] addresses the privacy-utility trade-off by using game theory, which can be generalized to consider the temporal effects by updating the prior after each iteration. In [39], they take the correlation over time into account on the utility-privacy game designed in [36].

We further consider anonymity in temporal graphs with unlabeled vertices. *The time series data should be seriously considered*, since the adversaries can combine multiple published graph to launch enhanced attacks for inferring more information. [5], [10], [38] explored privacy degradation in vertex privacy schemes due to the release of multiple graph snapshots. These observations motivate our work, even though we focus on labeled vertices.

De-anonymization In recent years, the security community has proposed a number of sophisticated attacks for de-anonymizing social graphs [20], [30], [32], [37]. While most of these attacks are not applicable to link privacy mechanisms (their focus is on vertex privacy), they illustrate the importance of considering adversaries with prior information about the social graph[5]. We perform a rigorous privacy analysis of LinkMirage (Section V) by considering a worst-case (strongest) adversary that knows the entire social graph except one link, and show that even such an adversary is limited in

---

[5]Burattin et al [6] exploited inadvertent information leaks via Facebook's graph API to de-anonymize social links; Facebook's new graph API (v2.0) features stringent privacy controls as a countermeasure.

its inference capability.

IX. DISCUSSION

Privacy Utility Tradeoffs: LinkMirage mediates privacy-preserving access to users' social relationships. In our privacy analysis, we consider the worst-case adversary who knows the entire social link information except one link, which conservatively demonstrates the superiority of our algorithm over the state-of-the-art approaches. LinkMirage benefits many applications that depend on graph-theoretic properties of the social graph (as opposed to the exact set of edges). This also includes recommendation systems and E-commerce applications.

Broad Applicability: While our theoretical analysis of LinkMirage relies on undirected links, the obfuscation algorithm itself can be generally applied to directed social networks. Furthermore, our underlying techniques have broad applicability to domains beyond social networks, including communication networks and web graphs.

X. CONCLUSION

LinkMirage effectively mediates privacy-preserving access to users' social relationships, since 1) LinkMirage preserves key structural properties in the social topology while anonymizing intra-community and inter-community links; 2) LinkMirage provides rigorous guarantees for the anti-inference privacy, indistinguishability and anti-aggregation privacy, in order to defend against sophisticated threat models for both static and temporal graph topologies; 3) LinkMirage significantly outperforms baseline static techniques in terms of both link privacy and utility, which have been verified both theoretically and experimentally using real-world Facebook dataset (with 870K links) and the large-scale Google+ dataset (with 940M links). LinkMirage enables the deployment of real-world social relationship based applications such as graph analytic, anonymity systems, and Sybil defenses while preserving the privacy of users' social relationships.

REFERENCES

[1] L. J. Almeida and A. de Andrade Lopes, "An ultra-fast modularity-based graph clustering algorithm," in *Proceedings 14th Portuguese Conference on Artificial Intelligence (EPIA)*, 2009.

[2] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz, "Trust-based recommendation systems: an axiomatic approach," in *WWW*, 2008.

[3] T. Aynaud, E. Fleury, J.-L. Guillaume, and Q. Wang, "Communities in evolving networks: Definitions, detection, and analysis techniques," in *Dynamics On and Of Complex Networks*, 2013.

[4] F. Beato, M. Conti, and B. Preneel, "Friend in the middle (fim): Tackling de-anonymization in social networks," in *PERCOM*, 2013.

[5] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Privacy in dynamic social networks," in *WWW*, 2010.

[6] A. Burattin, G. Cascavilla, and M. Conti, "Socialspy: Browsing (supposedly) hidden information in online social networks," *arXiv preprint*, 2014.

[7] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

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

[9] R. Dey, Z. Jelveh, and K. Ross, "Facebook users have become much more private: a large scale study," in *IEEE SESOC*, 2012.

[10] X. Ding, L. Zhang, Z. Wan, and M. Gu, "De-anonymizing dynamic social networks," in *GLOBECOM*, 2011.

[11] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security*, 2004.

[12] C. Dwork, "Differential privacy," in *Automata, languages and programming*. Springer, 2006.

[13] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *STOC*, 2009.

[14] N. Z. Gong, W. Xu, L. Huang, P. Mittal, E. Stefanov, V. Sekar, and D. Song, "Evolution of social-attribute networks: measurements, modeling, and implications using google+," in *IMC*, 2012.

[15] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *Proceedings of the VLDB Endowment*, 2008.

[16] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing social networks," *Technical Report, University of Massachusetts, Amherst*, 2007.

[17] H. Hodson, "Google software lets you get online via your friends," *New Scientist*, 2013.

[18] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[19] S. Ji, W. Li, P. Mittal, X. Hu, and R. Beyah, "Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization," in *Usenix Security*, 2013.

[20] S. Ji, W. Li, M. Srivatsa, and R. Beyah, "Structural data de-anonymization: Quantification, practice, and implications," in *CCS*, 2014.

[21] Y. Kim and J. Srivastava, "Impact of social influence in e-commerce decision making," in *Proceedings of the ninth international conference on Electronic commerce*, 2007.

[22] A. Kyrola, G. E. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a pc." in *OSDI*, 2012.

[23] N. Li, W. Qardaji, D. Su, Y. Wu, and W. Yang, "Membership privacy: a unifying framework for privacy definitions," in *CCS*, 2013.

[24] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, 2007.

[25] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *SIGMOD*, 2008.

[26] A. Mislove, A. Post, P. Druschel, and P. K. Gummadi, "Ostra: Leveraging trust to thwart unwanted communication." in *NSDI*, 2008.

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

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

[29] S. Nagaraja, "Anonymity in the wild: mixes on unstructured networks," in *PETS*, 2007.

[30] A. Narayanan and V. Shmatikov, *De-anonymizing social networks*. IEEE S&P, 2009.

[31] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, 2006.

[32] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn, "Community-enhanced de-anonymization of online social networks," in *CCS*, 2014.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." 1999.

[34] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing graphs using differentially private graph models," in *IMC*, 2011.

[35] R. Shokri, "Privacy games: Optimal user-centric data obfuscation," in *PETS*, 2015.

[36] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: optimal strategy against localization attacks," in *CCS*, 2012.

[37] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social network as a side-channel," in *CCS*, 2012.

[38] C.-H. Tai, P.-J. Tseng, P. S. Yu, and M.-S. Chen, "Identities anonymization in dynamic social networks," in *ICDE*, 2011.

[39] G. Theodorakopoulos, R. Shokri, C. Troncoso, J.-P. Hubaux, and L. B. Jean-Yves, "Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services," in *WPES*, 2014.

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

[41] M. Wright, M. Adler, B. N. Levine, and C. Shields, *Defending Anonymous Communications Against Passive Logging Attacks*. IEEE S&P, 2003.

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

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

[44] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *PinKDD*, 2008.

[45] B. Zhou, J. Pei, and W. Luk, "A brief survey on anonymization techniques for privacy preserving publishing of social network data," *ACM SIGKDD Explorations Newsletter*, 2008.

APPENDIX

A. Proof of the Upper Bound of Anti-aggregation Privacy

$\|P_t^k - P_t'\|_{\text{TV}} = \|P_t^k - \hat{P}_t^k\|_{\text{TV}} \leq \frac{1}{2|V_t|} \sum_{v=1}^{|V_t|} \|P_t(v)P_t^{k-1} - \hat{P}_t(v)P_t^{k-1}\|_1 + \frac{1}{2|V_t|} \sum_{v=1}^{|V_t|} \|\hat{P}_t(v)P_t^{k-1} - \hat{P}_t(v)\hat{P}_t^{k-1}\|_1 = \|P_t - \hat{P}_t\|_{\text{TV}} + \|P_t^{k-1} - \hat{P}_t^{k-1}\|_{\text{TV}} \leq k\|P_t - \hat{P}_t\|_{\text{TV}}$.

B. Relationships with Differential Privacy

When considering differential privacy for a time series of graph sequence $\{G_i\}_{i=0}^t$, we have $f(D) = P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 1), f(D') = P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 0)$. For a good privacy performance, we need $P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 1) \approx P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 0)$. Since the probability of $\{G_i'\}_{i=0}^t$ given $\{\tilde{G}_i(L_t)\}_{i=0}^t$ as $P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t) = P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 1)P(L_t = 1|\{\tilde{G}_i(L_t)\}_{i=0}^t) + P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t = 0)P(L_t = 0|\{\tilde{G}_i(L_t)\}_{i=0}^t)$, it is easy to see that if the condition for a good privacy performance holds, we have $P(L_t|\{G_i'\}_{i=0}^t, \{\tilde{G}_i(L_t)\}_{i=0}^t) = \frac{P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t, L_t) \times P(L_t|\{\tilde{G}_i(L_t)\}_{i=0}^t)}{P(\{G_i'\}_{i=0}^t|\{\tilde{G}_i(L_t)\}_{i=0}^t)} \approx P(L_t|\{\tilde{G}_i(L_t)\}_{i=0}^t)$, which is the same as in Definition 1 and means that the posterior probability is similar to the prior probability, i.e., the adversary is bounded in the information it can learn from the perturbed graphs.

C. Proof of Theorem 4: Expectation of Perturbed Degree

According to Theorem 3 in [27], we have $\mathbb{E}(\deg_{com}'(v)) = \deg(v)$, where $\deg_{com}'(v)$ denotes the degree of v after perturbation within community. Then we consider the random perturbation for inter-community subgraphs. Since the probability for an edge to be chosen is $\frac{\deg(v_a(i)) \deg(v_b(j))|v_a|}{|E_{ab}|(|v_a| + |v_b|)}$, the expected degree after inter-community perturbation satisfies $\mathbb{E}(\deg_{inter}'(v_a(i))) = \sum_j \frac{\deg(v_a(i)) \deg(v_b(j))(|v_a| + |v_b|)}{|E_{ab}|(|v_a| + |v_b|)} = \deg(v_a(i))$. Combining with the expectations under static scenario, we have $\mathbb{E}(\deg'(v)) = \deg(v)$.

D. Proof of the Upper Bound for the Utility Distance

We first introduce some notations and concepts. We consider two perturbation methods in the derivation process below. The first method is our dynamic perturbation method, which takes the graph evolution into consideration. The second method is the intermediate method, where we only implement dynamic clustering without selective perturbation. That is to say, we cluster G_t, then perturb each community by the static method and each inter-community subgraphs by randomly connecting the marginal nodes, independently. We denote the perturbed graphs corresponding to the dynamic, the intermediate method by $G_t', G_t'^{,i}$ respectively. Similarly, we denote the perturbed TPM for the two approaches by $P_t', P_t'^{,i}$. To simplify the derivation process, we partition the proof into two stages. In the first stage, we derive the UD upper bound for the intermediate perturbation method. In the second stage, we derive the relationship between $G_t'^{,i}$ and G_t'. Results from the two stages can be combined to find the upper bound for the utility

distance of LinkMirage. Denoting the communities as $C_1, C_2, \cdot, C_{K_t}$ and the inter-community subgraphs as $C_{12}, C_{13}, \cdots$, we have

$$\|P_t - P_t'^{,i}\|_{\text{TV}}$$
$$= \left\| \begin{matrix} P_{t(1,1)} - P_{t(1,1)}' & \cdots & P_{t(1,K_t)} - P_{t(1,K_t)}' \\ P_{t(2,1)} - P_{t(2,1)}' & \cdots & P_{t(2,K_t)} - P_{t(2,K_t)}' \\ \vdots & \vdots & \ddots \\ P_{t(K_t,1)} - P_{t(K_t,1)}' & \cdots & P_{t(K_t,K_t)} - P_{t(K_t,K_t)}' \end{matrix} \right\|_{\text{TV}} \quad (7)$$
$$= \frac{1}{|V_t|} \sum_{k=1}^{K_t} |V_t(k)| \|P_{t(k,k)} - P_{t(k,k)}'\|_{\text{TV}}$$
$$+ \frac{1}{|V_t|} \sum_{k,j=1, k \neq j}^{K_t} |E_t(k,j)| \|P_{t(k,j)} - P_{t(k,j)}'\|_{\text{TV}}$$
$$\leq \epsilon + \delta_t$$

Here, δ_t is the ratio cut of the graph [1], and $\delta_t = |E_{t-\text{in}}|/|V_t| = \sum_{k,j=1, k \neq j}^{K_t} |E_t(k,j)|/|V_t|$. For arbitrary matrix P and Q, we have $\|P^l - Q^l\|_{\text{TV}} \leq l\|P - Q\|_{\text{TV}}$. Combining the above results, we have $\text{UD}(G_t, G_t'^{,i}, l) \leq l\|P_t - P_t'^{,i}\|_{\text{TV}} \leq l(\epsilon + \delta_t)$. Then, we generalize the utility analysis of intermediate perturbation to our dynamic perturbation. Assume that there are K_t^c out of K_t clusters that are considered as changed, which would be perturbed independently, and K_t^u out of K_t clusters are considered as unchanged, i.e., their perturbation would follow the perturbation manner in G_{t-1}'. To simplify derivation, we use $P_{t(k)}$ instead of $P_{t(k,k)}$ to represent the TPM of the k-th community. Then, we have

$$\text{UD}(G_t, G_t', 1) = \|P_t - P_t'\|_{\text{TV}}$$
$$\leq \frac{\sum_{k=1}^{K_t^c} \|P_{t(k)} - P_{t(k)}'\|_{\text{TV}} + \sum_{j=1}^{K_t^u} \|P_{t(j)} - P_{t(j)}'\|_{\text{TV}}}{|K_t|} + \delta_t$$
$$\leq \frac{1}{|K_t|} \left(\sum_{k=1}^{K_t^c} \|P_{t(k)} - P_{t(k)}'\|_{\text{TV}} + \sum_{j=1}^{K_t^u} \left(\|P_{t(j)} - P_{(t-1)(j)}\|_{\text{TV}} \right. \right.$$
$$+ \|P_{(t-1)(j)} - P_{(t-1)(j)}'\|_{\text{TV}} + \|P_{(t-1)(j)} - P_{t(j)}'\|_{\text{TV}} \right) \bigg) + \delta_t \quad (8)$$
$$= \frac{\sum_{k=1}^{K_t} \|P_{t(k)} - P_{t}'(k)\|_{\text{TV}}}{|K_t|} + \frac{|K_t^u|\epsilon_0}{|K_t|} + \delta_t$$
$$\leq \text{UD}(G_t, G_t'^{,i}, 1) + \epsilon + \delta_t$$

where ϵ_0 denotes the threshold to classify a community as *changed* or *unchanged*. The last inequality comes from the fact that $\epsilon_0 \leq \epsilon$. Then, we can prove $\text{UD}(G_t, G_t', l) = \|P_t^l - (P_t')^l\|_{\text{TV}} \leq l\|P_t - P_t'\|_{\text{TV}} \leq l\|P_t - P_t'^{,i}\|_{\text{TV}} + l(\epsilon + \delta_t) = 2l(\epsilon + \delta_t)$ and $\text{UD}(G_0, \cdots G_T, G_0', \cdots G_T', l) \leq \frac{1}{T+1} \sum_{t=0}^T 2l(\epsilon + \delta_t)$.

E. Proof for Relating Utility Distance with Structural Metrics

From the definition of total variation distance, we have $\|P_v^r(G_t') - \pi\|_{TV} + \|P_v^r(G_t) - \pi\|_{TV} \geq \|P_v^r(G_t') - P_v^t(G)\|_{TV}$. Taking the maximum over all vertices, we have $\max \|P_v^r(G_t') + \pi\|_{TV} + \max \|P_v^r(G_t) - \pi\|_{TV} \geq \max \|P_v^r(G_t') - P_v^t(G)\|_{TV}$. Therefore, for $t \geq \tau_G(\epsilon)$, $\max \|P_v^r(G_t') - \pi\|_{TV} \geq \max \|P_v^r(G_t') - P_v^t(G)\|_{TV} + \max \|P_v^r(G_t) - \pi\|_{TV} \geq \frac{\sum_{v=1}^{|V_t|} \|P_v^r(G_t') - P_v^t(G)\|_{TV} - \pi\|_{TV}}{|V_t|} - \epsilon = \text{UD}(G_t, G_t', \tau_G(\epsilon)) - \epsilon$. Then, we have $\tau_{G_t'}(\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon)) - \epsilon) \geq \tau_{G_t}(\epsilon)$. It is known that the second largest eigenvalue modulus is related to the mixing time of the graph as $\tau_{G_t'}(\epsilon) \leq \frac{\log n + \log \frac{1}{\epsilon}}{1 - \mu_{G_t'}}$. From this relationship, we can bound the SLEM in terms of the mixing time as $1 - \frac{\log n + \log (\frac{1}{\epsilon})}{\tau_{G_t'}} \leq \mu_{G_t'}$. Replacing ϵ with $\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon)) - \epsilon$, we have $1 - \frac{\log n + \log \frac{1}{\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon)) - \epsilon}}{\tau_{G_t'}(\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon)) - \epsilon)} \leq \mu_{G_t'}$. Finally, we leverage $\tau_{G_t'}(\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon) - \epsilon)) \geq \tau_{G_t}(\epsilon)$ in the above equation, to obtain $\mu_{G_t'} \geq 1 - \frac{\log n + \log \frac{1}{\text{UD}(G_t, G_t', \tau_{G_t}(\epsilon)) - \epsilon}}{\tau_{G_t}(\epsilon)}$.