Change of Guard: The Next Generation of Social Graph De-anonymization Attacks

Kumar Sharad
University of Cambridge
kumar.sharad@cl.cam.ac.uk

ABSTRACT

Past decade has seen active research in social graph de-anonymization with a variety of algorithms proposed. Previous algorithms used handcrafted tricks and were locked in a co-evolution of attack and defense with design of anonymization systems. We present a radically improved algorithm for re-identifying social network data. We use a machine learning system based on random forests to identify nodes using their structural features. The algorithm can handle a variety of threat models and is agnostic to the de-anonymization scheme employed. This is substantiated by our evaluation using three real-world social graph datasets under four threat models. Our algorithm is consistently better than the previous generation of algorithms as confirmed by comparison with seven seed-based and seedless attacks based on two real-world social networks. It is time for attacks based on heuristics to be replaced by learning models.

1. INTRODUCTION

The anonymization of social graphs is becoming an important problem because of the growing secondary uses of social network data. The collection of personal data has become pervasive in the past decade. Data are often high-dimensional, content rich and very desirable for research community. Social network datasets have been a particular favorite of the researchers which are shared by data holders fairly openly [1, 2].

Privacy is often seen as a hindrance and applied as an afterthought. This manifests itself as poorly thought through data anonymization schemes. Social networks are specially hard to anonymize due to the interconnectivity of individuals. The literature suggests that it may not be possible to effectively anonymize high-dimensional data without destroying its utility [3]. Privacy risks can be alleviated to some extent by interactive mechanisms like differential privacy [4–7] where instead of releasing the data a query interface is provided and the replies are tightly controlled by adding noise to provide a privacy guarantee. Such mechanisms have their limitations as they are non-trivial to setup, expensive to maintain and introduce excessive noise where the data analyst's goals are unknown [6]. Non-interactive privacy mechanisms that release a perturbed version of the data have been widely proposed [8–15]. Privacy researchers routinely break such mechanisms [16–31]. Overwhelming evidence suggests either privacy or utility must be relinquished if high-dimensional datasets are to be released.

Ji *et al.* [27] show that most anonymized social network datasets can be largely de-anonymized purely based on structural knowledge but searching for the optimal attack is computationally infeasible. None of the approaches so far show performance close to what is possible [27]. All approaches rely on hand-picked heuristics to exploit structural similarity in social graphs. We overcome these hurdles by proposing a fresh approach using machine-learning models to obtain an appreciable improvement. In particular, we present a new generation of graph de-anonymization algorithm that is seedless and automatically discovers artefacts from data samples to construct a de-anonymization learning model thus doing away with the need for hand selected features. Learning from pertinent examples limits human intervention and creates models that are robust, reliable and nimble to change.

Our contributions. Our primary contributions are:

- We present a new generation of heuristic free seedless graph de-anonymization algorithm that uses machine learning to map nodes across graphs (Section 3.3).
- We show that our proposed scheme can handle a variety of threat models and is agnostic to the de-anonymization scheme employed (Section 4.1).
- The algorithm is evaluated on three real-world social graph datasets under four threat models (Section 4).
- We conduct a thorough comparison of our algorithm with seven seed-based and seedless attacks using two real-world social network datasets. Our algorithm's overall performance is better than all the others by a healthy margin (Section 5.1).

2. DE-ANONYMIZATION LANDSCAPE

Graph de-anonymization traditionally tries to obtain a mapping between two graphs using just their structure. The graph nodes represent individuals and the edges represent relations among them. After an anonymized graph is released the adversary tries to match it with a known graph at to compromise privacy. The adversary's knowledge is often imperfect, but may be sufficient to launch potent re-identification attacks [2, 18, 19]. Identification of individuals present in both graphs could lead to discovery of sensitive relations among them. The adversary could obtain side information from a number of sources such as another data release, scraping the web or by colluding with individuals who are part of a common social network. Most data releases aim to preserve some utility to facilitate analysis thus limiting possible perturbation of the data. This allows the attackers to try to unravel the anonymization using structural similarity and if available, any other side information.

AISec'16, October 28 2016, Vienna, Austria

ISBN 978-1-4503-4573-6/16/10. . . $15.00

DOI: http://dx.doi.org/10.1145/2996758.2996763

2.1 Anonymizing Social Networks

Several social network anonymization schemes try to protect node privacy, i.e., concealing an individual's presence in the graph [8–15]. Such schemes are mainly of two types [32–34] – clustering-based and perturbation-based. Clustering-based graph anonymization schemes release aggregate graph information instead of the raw graph while perturbation-based schemes introduce imperfections in the graph via a combination of edge additions/deletions to deter graph-structure-based de-anonymization attacks. Summarizing data limits its scope and usage but provides better privacy. Our work considers perturbation-based schemes, which are more popular due to their wider applicability and simplicity.

2.2 Heuristics vs. Machine Learning

Perfect recovery of common nodes across two graphs is a hard problem and computationally infeasible for large graphs in general [35]. The problem becomes even harder when the two graphs are noisy and the edges are perturbed. To make the problem tractable an imperfect and incomplete mapping is computed whose goal is to minimize the error and maximize the mappings. Modern graph de-anonymization algorithms are quite potent and can re-identify a large fraction of nodes with good accuracy [16–18, 21–31]. All such algorithms exploit graph heuristics to recover nodes based on structural properties. This approach, although potent, cannot produce the best performance with changing real-world anonymization schemes. Additionally there is no predefined manual to choose heuristics. The choice is based on judgment and prior knowledge of anonymization techniques and is a constant co-evolution of attack and defense.

Basic features based on neighborhood degree can be used to build complex and expressive models that capture the learning task very precisely [19, 36]. Handcrafting heuristics also suffers from the problem of tuning the parameters for various threat models, datasets and graph metrics which is done by trial and error. Pedarsani *et al.*'s [29] approach is a prime example of the limitations. The authors present a Bayesian model to capture the similarity of two individuals belonging to different graphs. The model requires knowledge of the anonymization scheme and can only handle simple features as it gets too complicated when feature complexity rises. Machine learning models can handle such complications and do not require manual tuning. In the rest of paper we show how to design such systems to breach privacy.

3. ALGORITHM DESIGN

In this section we describe the threat model under which our de-anonymization algorithm is evaluated. We also define the key components of the learning model and their role in deconstructing the anonymization of social graphs.

3.1 The Threat Model

We evaluate our attack under the well-studied threat model originally proposed by Narayanan and Shmatikov [18, 37]. The model assumes that the adversary has access to an auxiliary graph which is used as side information to re-identify individuals in a sanitized graph. Often data holders chose to sanitize the graphs prior to publishing them by removing all identifying information and adding noise via edge manipulation (see, Section 2.1).

The adversary is simulated by sampling overlapping graphs from a large social graph. The sampling generates auxiliary ($G_{aux} = (V_{aux}, E_{aux})$) and sanitized ($G_{san} = (V_{san}, E_{san})$) graphs from the original graph ($G = (V, E)$), where V and E represent nodes and edges respectively. The overlap between the node and edge sets of G_{aux} and G_{san} is measured in terms of Jaccard Coefficient, defined as $JC(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$ where X and Y are sets at least one of which is non-empty. Node set overlap of α_V is obtained by partitioning V into three sets V_A, V_B and V_C at random. The size of V_A, V_B and V_C is set as $\frac{1-\alpha_V}{2} \cdot |V|$, $\alpha_V \cdot |V|$ and $\frac{1-\alpha_V}{2} \cdot |V|$ respectively thus giving us $V_{aux} = V_A \cup V_B$ and $V_{san} = V_B \cup V_C$. Further noise is injected through edge perturbation – randomly deleting edges between nodes in V_{aux} and V_{san}; done by making two copies of E and independently deleting edges at random from each copy. The two copies are then projected on to V_{aux} and V_{san} to obtain E_{aux} and E_{san}. A fraction of $\frac{1-\alpha_E}{1+\alpha_E}$ edges are deleted at random from both graphs to get an edge set overlap of α_E. This overlap is only among the edges among the subgraph common to G_{aux} and G_{san}, the edge overlap for the entire graph is much lower. Deleting edges is widely used [18, 26, 27, 37] and one of the best ways to introduce noise as it produces an even degradation of the graph metrics and provides anonymity superior other schemes [20].

3.2 Learning to De-anonymize

We use a random-forests machine learning model to de-anonymize social graphs. The learning task is to classify a pair of nodes selected at random from G_{aux} and G_{san} as identical or non-identical. In this section we lay the foundation of the model and show how to construct it. The simple learning task of node pair classification is at the heart of our algorithm. We show how success in the learning task translates to success at de-anonymization.

3.2.1 Features

Degree distribution is a fundamental graph property [38] and a node can be uniquely identified by its neighborhood degree distribution [13]. This cannot be damaged too much without rendering the data useless [20]. Research in link prediction [39, 40] depicts strong benefits of using node similarity metrics to predict links. Such metrics show significant improvement over random guessing. The de-anonymization literature provides instances of node similarity metrics derived using neighborhood degree distribution to represent a node in a graph [19, 20]. In line with these approaches we define a node's feature vector using its neighborhood. These are generic graph features, and not specific to any anonymization scheme.

The neighbors of a graph node can be split into two categories, 1-hop nodes and 2-hop nodes. The shortest distance between a node and its 1-hop and 2-hop neighbors is one and two respectively. The degree distribution of the neighbors is quantized to represent a node [19, 36]. Figure 1 shows a feature vector composed of quantized neighborhood degree distribution. Each bin contains the count of nodes that have degree in a given range; for instance $c_0 = 23$ indicates that there are 23 nodes in the neighborhood with degree in the range (0 - b) whereas there are no nodes in the neighborhood with degree in the range ((2 $\cdot$ b + 1) - (3 $\cdot$ b)) for $c_2 = 0$. Such vectors can be created for both 1-hop and 2-hop neighbors of a node in an undirected graph – a total of two node categories. In a directed graph the 1-hop nodes are divided into successors and predecessors while the 2-hop nodes can be divided into successor-of-successor, successor-of-predecessor, predecessor-of-successor and predecessor-of-predecessor – a total of six node categories. A node feature vector is defined as a concatenation of the two degree distribution vectors for undirected graphs and 12 degree distribution vectors for directed graphs, counting both in-degree and out-degree for the six node categories.

Additionally, we calculate the Silhouette Coefficient of the number of 1-hop neighbors and 2-hop neighbors for a node pair. The Silhouette Coefficient of a node pair is defined as $\delta(d_1, d_2) = \frac{|d_1 - d_2|}{\max(d_1, d_2)}$, where d_1 and d_2 are the number of either 1-hop neighbors or 2-hop neighbors of nodes belonging to G_{aux} and G_{san}

respectively. This trains the model to predict the degree deviation for identical and non-identical node pairs. Modularity of features makes them nimble and adaptable; the features represent a node pair which is a data point for the learning model.

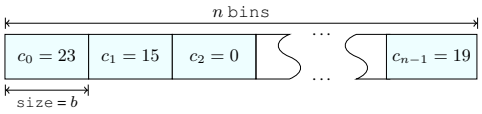

Figure 1: Feature vector of quantized node neighborhood

3.2.2 Training

Training a machine learning model in the absence of ground truth is challenging. However, this can be overcome by sampling training data by splitting G_{aux} and G_{san} [20]. To accomplish this the adversary just needs to have a rough estimate of the node set overlap between the two graphs. Using this knowledge both G_{aux} and G_{san} are split (see, Section 3.1) to create two sets of graphs and each set is used to sample identical and non-identical node pairs. After sampling the data from both sets is merged and used to train the model. The newly created graphs from splitting G_{aux} and G_{san} are not anonymized again since they have already undergone anonymization. The training pairs generated are used to train trees of the random forest thus helping the model to learn the features for classifying previously unseen node pairs.

3.2.3 Classification

After training the model, an unseen node pair can be classified. Classification proceeds by extracting features of the nodes in the pair and passing it through the random forest. Passing the node pair through a tree assigns it a probability of being identical. After the node pair has been through all the trees, the accumulated probabilities are averaged thus assigning the node pair a final prediction score which is a real number in [0,1]. The higher the score, the more likely the node pair to be identical.

3.3 Unraveling Anonymization

Classifying node pairs provides a measure of their structural similarity but it cannot be used directly to identify true mappings. Choosing a high classification threshold can be used to select identical node pairs with high likelihood. However, false positives cannot be completely avoided; too high a threshold will rule out a number of node pairs which in spite of being identical, will not be selected thus adversely affecting the number of mappings identified. The problem can be solved by two key observations [18]; first, high degree node pairs are easier to re-identify than low degree node pairs due to their higher information content and second, common friends of a node pair provide a valuable metric for identification [39]. Rest of the section describes how these observations can be used along with structural similarity as quantified by the classifier to produce actual node mappings.

3-phase re-identification. We carry out node re-identification in phases, the key idea is to re-identify high degree nodes early and then use them to attack low degree nodes. Since high degree nodes are easier to attack we divide the node pairs into three categories based on their degree. We pick three degree thresholds such that $t_1 > t_2 > t_3$ and divide the node pairs. All pairs with degree of both nodes greater than t_1 fall into *Phase 1*, all pairs with degree of both nodes greater than t_2 but not a part of *Phase 1* fall into *Phase 2* and finally all pairs with degree of both nodes greater than t_3 but not a part of either *Phase 1* or *Phase 2* fall into *Phase 3*. Nodes of degree lower than or equal to t_3 are left alone; t_3 is chosen as a low value and such nodes are hard to identify with high accuracy. Moreover, they are not influential and evoke little interest in an adversary. We train a separate random forest for each phase to focus the attack better. The attack begins with Phase 1 node pairs, moves on to Phase 2 and concludes with Phase 3 node pairs.

Initial mappings. To produce the initial set of mappings all node pairs in Phase 1 are classified and only pairs with classification score above 0.95 are selected. These mappings are *dirty* since at this stage a node might appear in multiple mappings thus making them contradictory.

Cleaning mappings. A set of dirty mappings are *cleaned* by greedily selecting node pairs starting with the node pair with the highest classification score. After a node pair is selected all further instances of both the nodes in the pair are discarded from the remaining mappings. Cleaning improves the overall accuracy of the mappings.

Filtering node pairs. The identified initial mappings are treated as true. At this stage, the mappings are bound to have errors but we have no way to differentiate between a true and a false mapping. However, the initial mappings are composed of node pairs which are structurally very similar, even the non-identical pairs, the nodes are highly likely to be in close proximity of each other. We leverage this knowledge to filter node pairs based on the friends they share. All node pairs (including those present in the initial mappings) are filtered based on the cosine similarity of their neighbors that have been identified in initial mappings. Cosine similarity between two non-empty sets X and Y is defined as $CS(X,Y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$. The filtering process only preserves node pairs that have a cosine similarity above a threshold. For a directed graph the cosine similarity is calculated as a sum of cosine similarity of common successors and predecessors. The filtered node pairs are cleaned again to discover new mappings, however, this time classification threshold is relaxed and node pairs self-select themselves till all pairs are exhausted. The new mappings discovered at the end of this step are preserved and previous mappings are forgotten.

Propagating the mappings. Filtering node pairs using the initial mappings increases the accuracy and coverage of subsequent mappings. Performance is further boosted by iterating the process. As new mappings are discovered, filtering node pairs become more reliable due to increase in accuracy and number of common friends; additionally, a larger set of mappings reaches higher number of nodes. This produces a snowball effect causing accuracy and coverage to grow till they stabilize. After this process concludes for Phase 1 the mappings are frozen and we continue with Phase 2. As we decrease the degree threshold to t_2 the number of node pairs rise sharply, making it inefficient to classify all node pairs. This is where the mappings from Phase 1 come in handy; the Phase 2 node pairs are pre-filtered before classification. Only the filtered mappings are classified and cleaned; additionally, we decrease the number of node pairs further by removing the nodes that have already been mapped in Phase 1. In Phase 2 node pairs are first filtered using Phase 1 mappings, after new mappings are discovered they are rolled in with the frozen Phase 1 mappings and the process is repeated. Whenever new mappings are found at a later phase they are rolled in with the frozen mappings, keeping only the latest mappings from the current phase. After the mappings from Phase 2 stabilize they are frozen and we repeat the same process for Phase 3. We observe that mappings stabilize a lot quicker at later phases as after a sufficient number of mappings has been discovered, new mappings do not influence the filtering much.

4. EVALUATION AND RESULTS

In this section we present the analysis of results obtained by running **3PSL** on the three datasets and four threat models.

4.1 Datasets and Implementation

The proposed 3-phase seedless attack (**3PSL**) is evaluated using three publicly available real-world social networks, Flickr [41] – a popular social network for sharing pictures, Epinions[1] – a who-trust-whom online social network of a general consumer review site and Enron[1] – an email communication network. The Flickr graph is undirected and provides group membership of the users, Epinions is directed and has no group information and Enron is an undirected graph. The number of nodes and edges in the original graphs are as follows: Flickr (nodes = 80 513, edges = 5 899 882), Epinions (nodes = 75 879, edges = 508 837) and Enron (nodes = 36 692, edges = 183 831). These graphs are used to produce overlapping auxiliary and sanitized graphs with varying node and edge overlaps (see, Section 3.1); the details are summarized in Table 1. The number of common nodes in each phase is depicted in in Table 2 and Table 3 shows the chosen degree thresholds (see, Section 3.3).

We analyze our attack with four different adversaries: (*i*) $\mathcal{A}_1$ – Flickr undirected graph and node group membership ($\alpha_E = \alpha_V = 0.33$) (*ii*) $\mathcal{A}_2$ – Epinions directed graph ($\alpha_E = 0.33, \alpha_V = 0.20$) (*iii*) $\mathcal{A}_3$ – Epinions directed graph ($\alpha_E = 0.50, \alpha_V = 0.35$) (*iv*) $\mathcal{A}_4$ – Enron undirected graph ($\alpha_E = 0.43, \alpha_V = 1$). This allows us to test our structural attack under varying graph densities, edge overlaps, node overlaps and side information like directionality and group membership. A Flickr node pair is considered identical only if the group membership matches exactly.

The code for the project is written in Python and run using CPython. We use a commodity laptop with a 2.8 GHz processor and 16 GB RAM to run our experiments. Table 4 shows the number of identical and non-identical pairs for the three phases used to train the random forest classifier.

Table 1: Graph details

	G_{aux}		G_{san}	
	Nodes	Edges	Nodes	Edges
Flickr ($\alpha_E = \alpha_V = 0.33$)	51 594	1 322 970	51 698	1 302 064
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	30 832	114 908	30 584	113 461
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	41 651	183 284	41 673	193 830
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	36 692	110 298	36 692	110 298

Table 2: Common nodes

	Phase 1	Phase 2	Phase 3	Total
Flickr ($\alpha_E = \alpha_V = 0.33$)	8684	6608	2779	18 071
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	759	1234	783	2776
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	1008	2727	1603	5338
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	1462	2214	8357	12 033

4.2 Phase 1 Classification

The classification of node pairs is successful with consistently high true positive and low false positive rate. This is important for the filtering to begin as it depends on the quality of initial node mappings. Additionally the classification produces good quality results overall as summarized by the area under the Receiver Operating Characteristic (ROC) curve (AUC). The ROC illustrates how close the classifier is to an ideal one. It does so by measuring the True Positive (TP) rate as the False Positive (FP) rate tolerated is varied in the range [0, 1]. An ideal classifier gives a TP rate of 1 at FP rate 0, whereas TP and FP are always the same for random guessing. In practice a classifier will always make errors (FP), our goal is to maximize the correct classification rate (TP) for the error tolerated. The Area Under the Curve (AUC) provides a summary of the quality of the classifier, an ideal classifier has an AUC = 1 where as random guessing produces a classifier with an AUC = 0.5. Classification success is an important factor behind high quality clean node pairs. Figure 2 shows the classification success and Table 5 provides a snapshot of the relation between TP and FP rates for Phase 1 for all adversaries. Even at a FP rate of 0.1% the TP rate is about 15-30% which allows us to mount an attack. Table 6 lists the number of mappings produced at various phases by **3PSL** as well as the total mappings. The sparser the graph the fewer high degree nodes it has and consequently it has fewer candidates for Phase 1, this in turn effects its propensity to be attacked. On the flip side sparse graphs limit the scope of analysis.

[1]https://snap.stanford.edu/data/index.html

Table 3: Degree thresholds for all phases

	t_1	t_2	t_3
Flickr ($\alpha_E = \alpha_V = 0.33$)	30	9	5
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	30	9	5
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	50	9	5
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	25	9	2

Table 4: Number of identical (**I**) and non-identical (**NI**) samples used for training

	Phase 1		Phase 2		Phase 3	
	I	**NI**	**I**	**NI**	**I**	**NI**
Flickr ($\alpha_E = \alpha_V = 0.33$)	8715	1 452 188	8765	1 988 697	4002	1 994 579
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	658	1 010 758	1303	1 998 397	920	1 998 940
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	1110	1 595 258	3729	1 996 012	2063	1 997 804
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	2219	3 338 583	3520	1 996 121	14 364	1 985 529

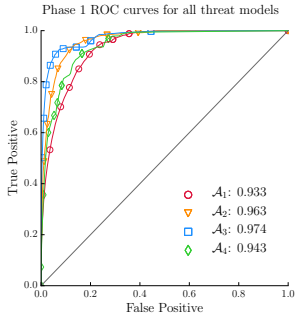

Figure 2: The ROC curve (AUC in legend)

4.3 Mapping Accuracy and Coverage

We use accuracy and coverage to study the performance of the de-anonymization algorithm. Accuracy and coverage are defined as $\frac{m_{true}}{m_{out}}$ and $\frac{m_{true}}{m_{tot}}$ respectively; where m_{out} is the number of mappings output by **3PSL**, m_{true} is the number of true mappings out of those output and m_{tot} is the total number of mappings between

Table 5: Phase 1: *False Positive* vs. *True Positive*

False Positive	0.001%	0.01%	0.1%	1%	10%	25%
Flickr ($\alpha_E = \alpha_V = 0.33$)	2.03	5.55	13.17	32.53	74.90	95.12
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	1.58	5.14	14.36	46.38	90.25	98.16
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	4.96	13.59	29.96	64.58	93.25	98.51
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	0.41	2.33	11.35	35.57	81.87	93.64

Table 6: Number of mappings produced

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	8534	5668	2743	16 945
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	721	1189	1774	3684
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	997	2791	1734	5522
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	1407	2077	7555	11 039

G_{aux} and G_{san}. Accuracy and coverage are important metrics to evaluate the quality of mappings and a balance is needed to produce good performance. Tables 7 and 8 show the accuracy and coverage of the mappings produced with a comparison presented in Figure 3. Results show that our algorithm performs well and produces high overall coverage and accuracy. The percentages for Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) are lower because of the graph being very sparse as well as a low edge and node overlap among auxiliary and sanitized graphs thus making it hard to attack. Figure 4 shows the behavior of accuracy and coverage with increasing node degree; as more information is available due to increase in degree the node pairs become easier to attack.

Table 7: Mapping accuracy (%)

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	95.85	92.36	66.10	89.87
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	74.06	37.85	6.82	29.99
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	96.69	85.78	55.36	78.20
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	86.99	71.74	25.31	41.91

Table 8: Mapping coverage for node degree above t_3 (%)

	Phase 1	Phase 2	Phase 3	Final
Flickr ($\alpha_E = \alpha_V = 0.33$)	94.20	79.22	65.24	84.27
Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)	70.36	36.47	15.45	39.81
Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)	95.63	87.79	59.89	80.89
Enron ($\alpha_E = 0.43, \alpha_V = 1$)	83.72	67.30	22.88	38.44

4.4 Evolution of Mappings

The mappings evolve as we iterate using the new mappings every time to filter node pairs based on cosine similarity starting with the initial mappings. Figure 5 shows that accuracy and coverage are always low at the start but after enough iterations they stabilize. Figure 6 depicts similar behavior for the total number of mappings. Discovery of mappings in Phase 1 ensures quick convergence in Phase 2 and Phase 3 with mappings stabilizing faster. The nature of the graph also effects the convergence; Flickr ($\alpha_E = \alpha_V = 0.33$) converges faster as group membership helps re-identify node pairs whereas Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) being sparser with low overlap takes the longest to converge. Increasing the Epinions overlap improves the adversary's side information and makes it

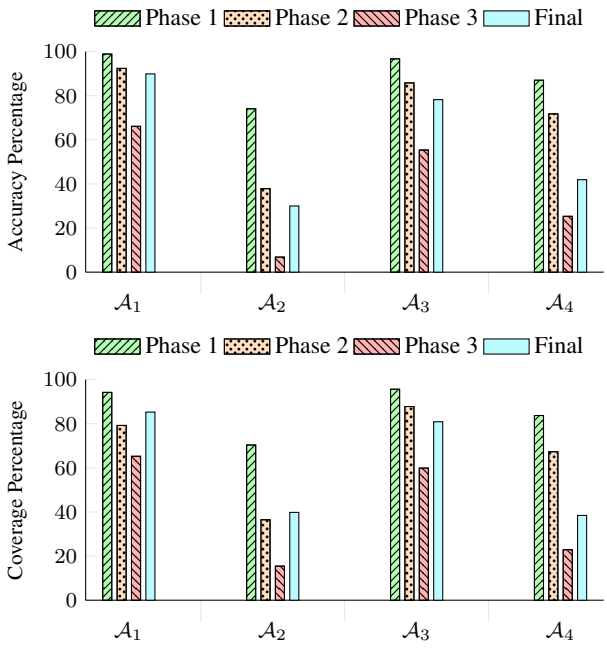

Figure 3: Success of adversary

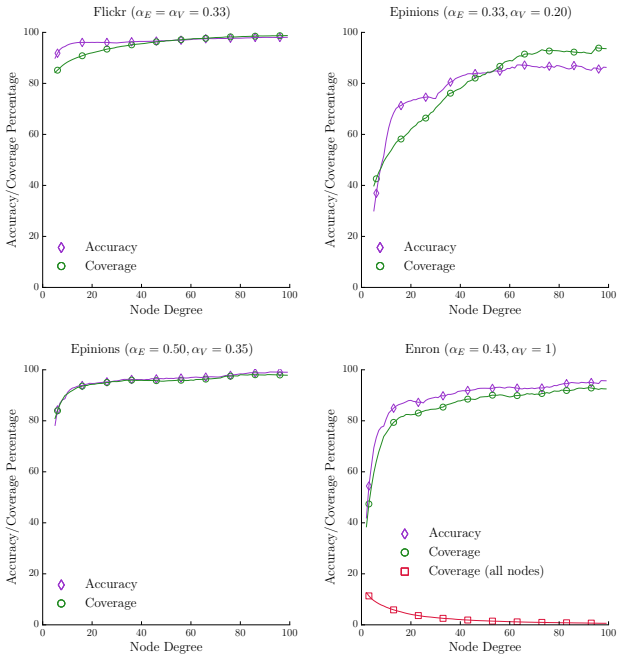

Figure 4: Variance of accuracy and coverage with degree

stronger and more successful as seen by quicker convergence. We perform an additional step for Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$) due to the low number of mappings; after completing Phase 2 we reuse the Phase 2 mappings along with Phase 1 mappings to run Phase 1 again, this improves the accuracy and coverage slightly after which we discard the old Phase 2 mappings and proceed as usual, only first run is reported here.

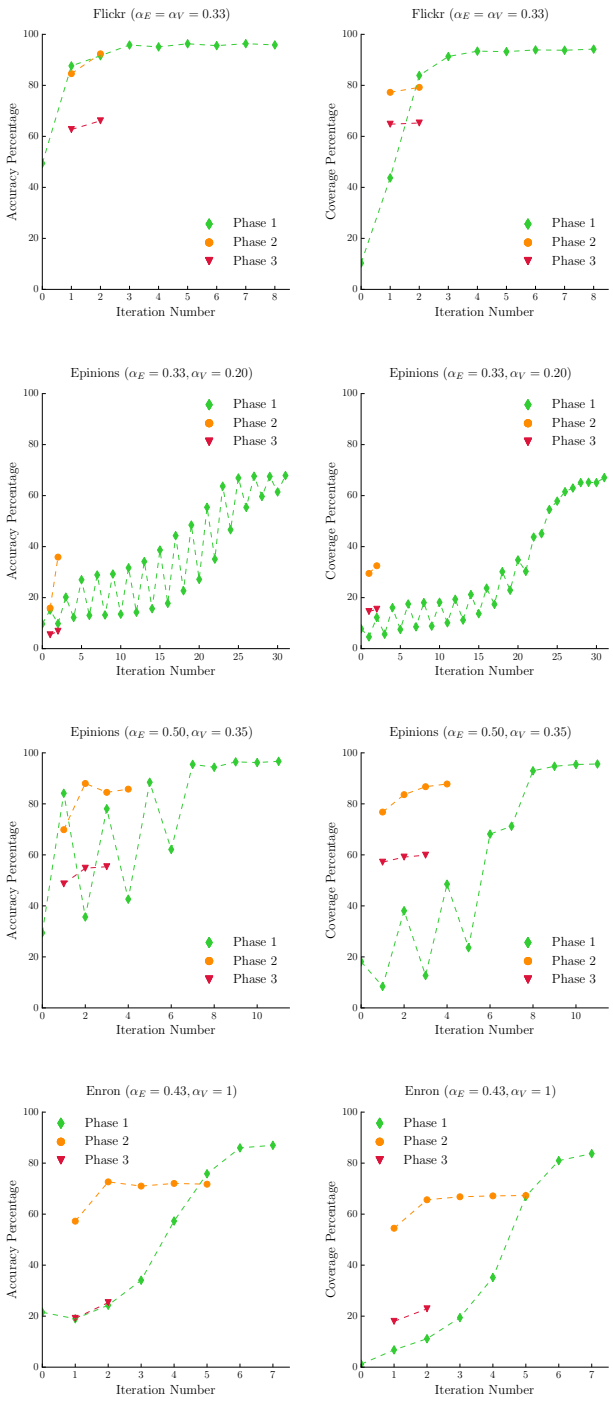

Figure 5: Variance of accuracy and coverage on iterating

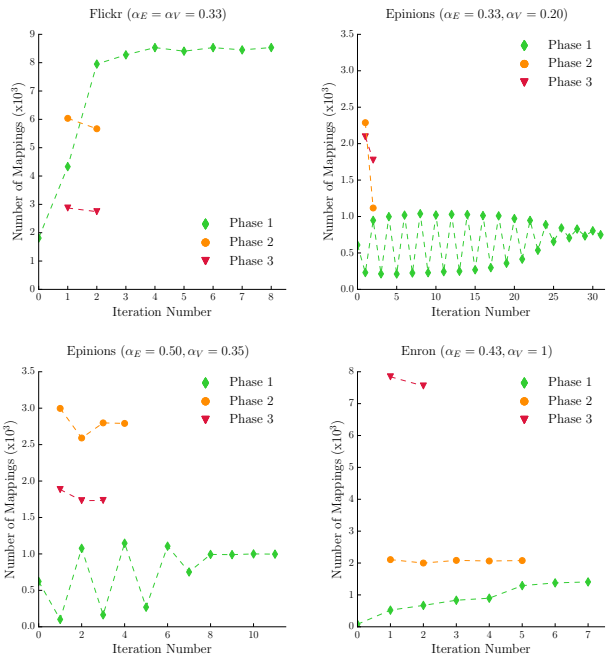

Figure 6: Variance of mapping count on iterating

fied are also concentrated among the low degree nodes with a light spread along the diagonal but their overall numbers are low compared to other mappings. A large number of true mappings are also concentrated among the low degree nodes since such node pairs are large in number and overwhelm the mappings in high node degree vicinity.

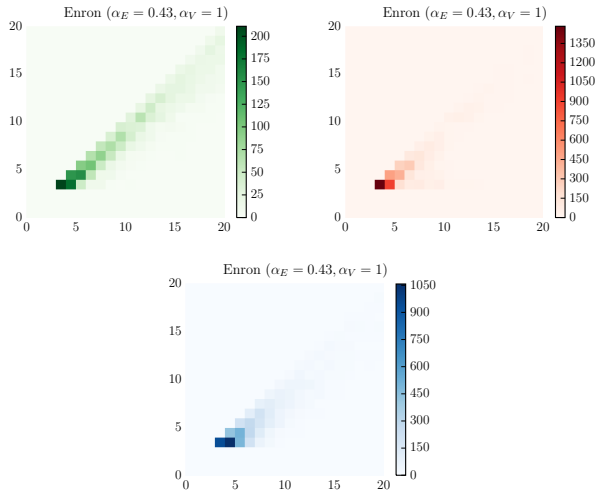

Figure 7: Joint Degree Distribution of node pairs

4.5 Error Analysis

In this section we analyze the errors that our algorithm makes. Phase 3 node pairs are the hardest to re-identify due to being low degree and consequently having insufficient information. Figure 7 illustrates a heat map of the joint degree distribution of true, false and unidentified node mappings for Enron (results for other datasets are similar). The figure shows that most false mappings are concentrated among low degree node pairs while the true mappings are spread out across the diagonal. Mappings that could not be identi-

We study the proximity of generated mappings by merging G_{aux} and G_{san} using the common nodes to create the complete graph G_{comp}. We measure the shortest path length between the generated mappings in G_{comp}. A true mapping produces a shortest path length of 0 due to the source and the target being identical. For each phase we plot the various path lengths as a percentage of the total mappings produced. Figures 8, 9, 10 and 11 reveal that the output mappings are always in close proximity, with almost all mappings

being within two hops of each other. Since the node features are extracted from 2-hop neighborhoods it is not surprising that almost all node mappings lie within this radius. It is highly unlikely for distant nodes to be matched and this is confirmed by the results. Our algorithm is most successful for Phase 1 node pairs and least successful for Phase 3 node pairs. The neighborhoods of low degree nodes are less diverse as compared to high degree nodes, moreover, the number of low degree node pairs at a distance of two hops from each other is a lot higher than node pairs at a distance of one hop. Hence, when the algorithm makes an error it is more likely to do so for low degree nodes at a distance of two hops from each other. This explains the reason for a larger percentage of errors in the region of two hop distance; the error percentage increases with graph sparsity as we lose diversity as observed in the difference between performance of $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$. Sparse graphs also have fewer Phase 1 and Phase 2 node mappings thus confounding filtering of low degree node pairs.

Node pairs with a higher neighborhood overlap are more likely to be identified. Figure 12 shows the cumulative percentage of node pairs above a given neighborhood overlap is considerably higher for node pairs that are correctly identified. The neighborhood overlap is measured as the Jaccard Coefficient of the 2-hop neighborhoods of the nodes of a pair. The mappings that are falsely identified as well as those that could not be identified share a very similar low neighborhood overlap which induces errors from the algorithm. An effective way to defeat structural attacks would be delete enough edges till overlap becomes low, the downside being such graphs would be unfit for any meaningful study.

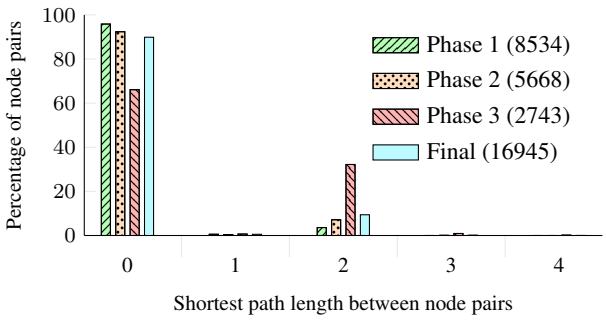

Figure 8: Flickr ($\alpha_E = \alpha_V = 0.33$)

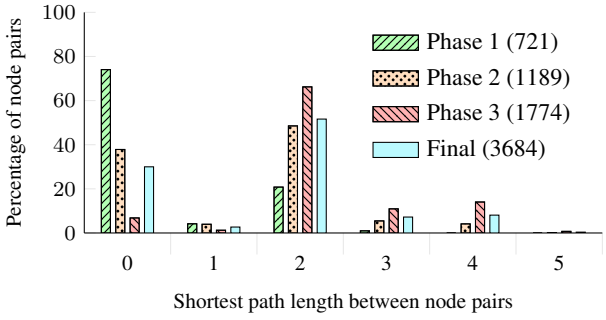

Figure 9: Epinions ($\alpha_E = 0.33, \alpha_V = 0.20$)

5. RELATED WORK

Graph de-anonymization attacks broadly fall into two categories – seed-based and seedless. Nilizadeh *et al.* [16] (**NKA**) exploit the community structure of social networks to augment de-anonymization. They propose a divide-and-conquer approach to de-anonymize nodes is two stages. First the communities across graphs are mapped, followed by mapping users within the communities.

Seed-based attacks start with a few known mappings between the graphs and aim to expand them using graph topology. Backstrom *et*

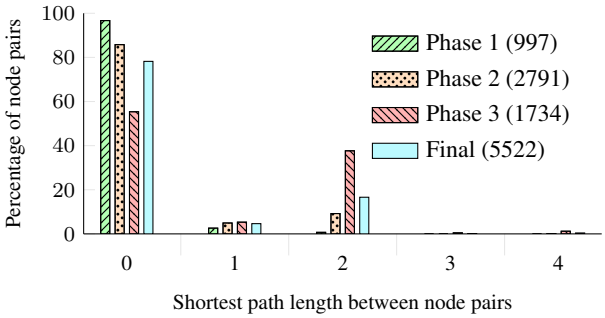

Figure 10: Epinions ($\alpha_E = 0.50, \alpha_V = 0.35$)

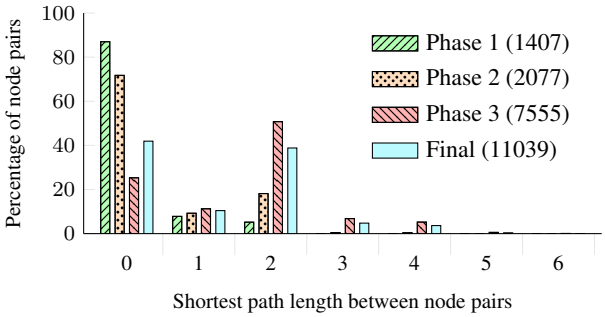

Figure 11: Enron ($\alpha_E = 0.43, \alpha_V = 1$)

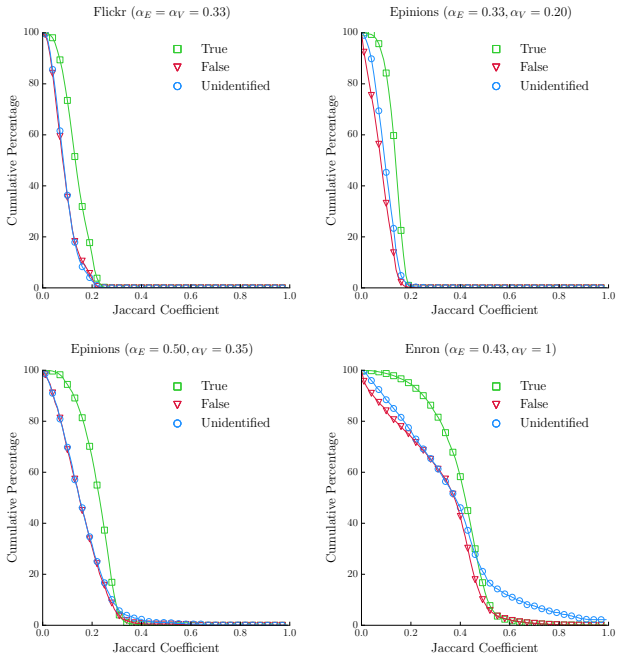

Figure 12: Cumulative percentage vs Jaccard Coefficient

al. [22] (**BDK**) were the first to present active and passive structural attacks on anonymized social networks. The active attack proceeds with the adversary inserting a small number of sybil nodes in the graph before it is released. The sybil nodes create links with a set of users whose privacy the adversary wishes to violate, and a pattern of connections is also formed among the sybil nodes to make it conspicuous. The passive attack is more subtle, the members of the graph do not form new links but try to find themselves in the published graph. They discover of links among users to which they are connected. While the active attack can be targeted it is likely to be detected [42–47]. The passive attack is hard to launch at scale and both attacks are sensitive to perturbation in the released data.

Srivatsa and Hicks [17] (**SH**) splice mobility traces with social network data to de-anonymize mobile users using common friends. They present a two step process; the first step identifies landmark nodes which are used to extend the mappings in the second step using node similarity heuristics. Node centrality is used to identify landmark nodes in both graphs; these nodes are mapped by trying all possible combinations and selecting the most likely mappings using a goodness-of-fit measure.

Narayanan and Shmatikov [18] (**NS**) present a two phase self-reinforcing and feedback-based attack which provides the adversary more auxiliary information as nodes are re-identified. The algorithm first re-identifies seed mappings which are then propagated to expand the mappings in the second phase. The attack requires enough seeds of high degree and with specific structural properties. The propagation step utilizes the seeds and topological information to discover new mappings, the process is iterative and large scale re-identification is possible under the right circumstances.

Korula and Lattanzi [23] (**KL**) use a high number of seeds (about 5-10%) and common neighbors to reconcile users across social networks. Their algorithm is not a hostile attack but is aimed at aiding users by suggesting friends. Their technique is based on absolute number of common friends and cannot attack low degree nodes.

Yartseva and Grossglauser [24] (**YG**) analyze the success of seed-based social graph de-anonymization algorithms. They analyze the dependence of large scale de-anonymization on the number of seeds and propose ways to estimate the critical seed set size. A seed based de-anonymization scheme similar to **KL**, based on common neighbors, is also proposed which is shown to depend upon a critical number of seeds, properties of the graph, the overlap with the auxiliary graph and common neighbor threshold for large scale percolation. The algorithm has a high error rate when vertex sets of auxiliary and sanitized graphs are not identical. Setting a threshold prevents it from attacking low degree nodes.

Ji *et al.* [25] quantify the de-anonymizability of social networks under seed knowledge. The authors investigate the feasibility of mounting seed based de-anonymization attacks on social graphs from the theoretical perspective. The results show that structural attacks are more potent than attacks based only on seed information.

Ji *et al.* [26] (**JLS+**) present two seed-based attacks that use structural similarity, common neighbors and seed induced distance between nodes to identify node mappings. The first attack needs the precise knowledge of overlap between the two graphs whereas the second attack estimates it. The two phase algorithm first selects seeds and then propagates them iteratively to the neighboring nodes. Each iteration uses the already mapped nodes for mapping expansion. The technique is resistant to moderate noise levels and achieves good accuracy.

Turning on to seedless attacks, Wondracek *et al.* [28] use web browser history stealing to demonstrate that social network group membership is sufficient to de-anonymize users. Ji *et al.* [27] (**JLSB**) study the de-anonymizability of social networks using structural information. They highlight the conditions under which perfect and imperfect de-anonymization can be achieved. Although, most social networks can be largely de-anonymized it is shown to be computationally infeasible to search for the optimal attack. A computationally feasible optimization based de-anonymization attack which is a relaxed version of the optimum scheme is proposed as a compromise. The attack performs well, though it is run on graphs with high overlap.

Perdarsani *et al.* [29] (**PFG**) propose a Bayesian framework to match common nodes across social networks without seeds. The algorithm proceeds in rounds starting with mapping high degree nodes using degree fingerprints. As most likely nodes are mapped they generate additional features based on distance to the mapped nodes which are used to map increasing number of nodes in subsequent rounds. The algorithm presented is limited in scope as the process relies on computing the probability of node pairs being identical based on their features. This requires knowledge of the anonymization scheme to create a probabilistic model and even after this knowledge it limits the features used to represent nodes as the model complexity increases with that of the features. Our algorithm is free from such limitations as the model is learned by the classifier automatically in the training phase.

Perdarsani and Grossglauser [30] study the conditions under which the structural correlation of two social graphs sampled with a noise from a larger graph becomes possible. The authors investigate the dependence of graph anonymity on its parameters irrespective of the graph de-anonymization algorithm applied. It is shown that the mean node degree needs to grow only slightly faster than $\log n$ for nodes of a random graph with n nodes to be identifiable. Although, the existence of an algorithm which achieves perfect matching has been demonstrated, it remains a challenge to discover it.

Sharad and Danezis [19] demonstrate that de-anonymization of social networks can be automated. Our work goes much beyond the scope of the prior work by substantially enhancing the previous techniques to recover end-to-end node mappings.

5.1 Comparison with the State of the Art

In this section we measure how well our attack fares as compared to other prominent ones. Ji *et al.* [37] conduct a survey of graph anonymization and de-anonymization strategies in which they evaluate the de-anonymization attacks using the Enron and Facebook datasets under $\mathcal{A}_4$'s threat model; we compare **3PSL** with the reported evaluation. We consider all the passive seed-based and seedless attacks for comparison, active attacks are excluded as they do not scale well and require considerable control on the part of the adversary. We exclude the passive attack of **BDK** as it does not work on perturbed graphs. All seed based attacks are provided 50 seed mappings [37]. The distance vector based attack of **SH** is used for comparison as all other attacks do not scale even with pre-identified seed mappings. Our comparison uses the graph overlap estimating attack of **JLS+**. Ji *et al.* [37] set $m_{out} = m_{tot}$ in their experiments[2], hence accuracy is equal to coverage (see, Section 4.3).

We compare the performance of attacks based on Enron and Facebook (nodes = 63 731, edges = 817 090) datasets same as those used by Ji *et al.* [37]. A shoulder to shoulder comparison is presented by measuring the coverage based on all node mappings, not just those attacked. We do not attack nodes of degree below three for Enron, doing so would increase the coverage but it would come at the cost of accuracy. Figure 4 shows the degradation of Enron's coverage computed for all nodes as degree is increased. Facebook is more dense than Enron and has a higher average node degree; as seen

[2]Confirmed via communication with the authors.

Table 9: Comparison of coverage and accuracy percentage with other attacks (accuracy = coverage for all attacks except **3PSL**); $^{\varnothing}$ denotes a seedless attack. A higher percentage is better.

	Enron		Facebook	
	Coverage	Accuracy	Coverage	Accuracy
KL	15.96	15.96	5.99	5.99
JLS+	13.05	13.05	15.68	15.68
SH	12.77	12.77	15.63	15.63
JLSB$^{\varnothing}$	11.91	11.91	14.73	14.73
YG	3.10	3.10	28.32	28.32
PFG$^{\varnothing}$	7.39	7.39	10.87	10.87
NS	0.37	0.37	0.18	0.18
3PSL$^{\varnothing}$	12.61	41.91	>40	>65

from Sections 4 and 5, this increases adversary's success [27, 37]. We estimate a lower bound for the overall accuracy and coverage for Facebook by using the values for Enron.

Table 9 presents a thorough comparison of all the attacks. We measure **3PSL**'s accuracy and coverage as node degree is varied thus facilitating a comparison. We see that despite attacking only nodes of degree greater than two and no seed knowledge **3PSL**'s overall performance is better in all scenarios by a large margin. **KL** is the best attack for Enron, even with seeds it only achieves a coverage and accuracy of 15.96% marginally better than our coverage of 12.61%, we achieve an accuracy of 41.91% which is over 2.5 times than that of **KL**. The best seedless attack for Enron fares even worse with an coverage and accuracy of 11.91%; comparatively our algorithm has a higher coverage with almost four times the accuracy. The best attack for Facebook is **YG** which uses seeds to achieve a coverage and accuracy of 28.32% our algorithm achieves a coverage of over 40% and more than twice the accuracy of over 65%. The best seedless attack for Facebook achieves a coverage and accuracy of 14.73%; comparatively our algorithm achieves more than twice the coverage with over four times the accuracy. Seed knowledge is critical to attack performance in sparse graphs (Enron) vis-a-vis a dense graph (Facebook). As adversary's access to structural information is limited, the dependence on seeds increases. Thus gap between seedless and seed-based attacks is larger for sparse graphs like Enron as compared to denser graphs like Facebook. Even in such an adverse scenario our algorithm shows better overall performance for Enron with only slightly lower coverage (compared to seed-based attacks) to achieve a drastic gain in accuracy, that too without seed knowledge. For Facebook which is much dense no other attack comes even close to the performance of our algorithm even with seed knowledge. The attack on Facebook is more successful as a larger fraction of node pairs fall under Phase 1 and 2 as compared to Enron which are easier to attack. In particular, only 32.79% of Enron nodes have degree over two whereas the percentage for Facebook rises to 65.18%. We can increase the coverage of our algorithm with a decrease in accuracy by attacking lower degree nodes. It is better to have a low coverage and high accuracy thus producing some useful mappings rather than high coverage and low accuracy which would be akin to random guessing. **3PSL**'s accuracy is appreciably higher than all other attacks including those that use seed knowledge.

6. DISCUSSION

As seen from Sections 4 and 5.1 **3PSL** shows a marked improvement over all graph de-anonymization algorithms. Not only does it perform better but it can also accommodate a variety of adversaries and does so while using less information than all the seed-based attacks. The attack is agnostic to the change of anonymization scheme and trains a model using data samples generated from anonymized graphs. The success of our attack emanates from the fact that we not only consider structural similarity but also similarity based on common friends to recover mappings. These metrics are orthogonal and thus complement each other perfectly. Our approach is contrasting to schemes relying only on common neighbors based similarity metrics [23, 24]; thus achieving an improved performance while attacking graphs with low node overlap.

Seed quality. Seed-based attacks assume knowledge of seeds to prime the de-anonymization process. This may be considered reasonable in certain scenarios, but possession of error free seed mappings is a very strong requirement. As seen in Section 5, algorithms impose additional requirements on the seed set such as their size, structure, degree, centrality and neighborhood. These requirements along with the size and structure of the graph under attack heavily influence the quality of the seeds and as a result do not generalize well [48]. Such graph de-anonymization algorithms are finicky as their success is very sensitive to the size and quality of the seed set, and below a critical size the attack fails to converge [24, 49, 50]. Table 9 shows that the performance of all seed-based attacks (**KL**, **NS**) does not improve when attacking Facebook as compared to Enron; seed selection has a huge impact on the results. Seed-based attacks are even less suitable when the node and edge set overlap of auxiliary and sanitized graphs is low. Exploiting seeds could lead to powerful de-anonymization attacks, though reliance on the availability of seeds is a concerning limitation. Ji *et al.* [25] demonstrate that structural attacks can be more potent than seed based attacks which is corroborated by our results thus tilting the scales further in favor of relinquishing dependence on seed-based attacks.

Parameter choice. Classifying node pairs is more expensive than filtering them hence choice of t_1 influences the overall efficiency of the algorithm as the runtime is linear in the number of node pairs classified. It is also important to partition node sets in such a way that trained decision forests can be focused to maximize accuracy due to similar properties of the data points. Choosing t_1 too low makes the data points too varied where as setting t_1 too high yields too few data points to effectively train the model; for this reason sparser graphs have a lower t_1. Flickr has a lower t_1 despite being dense because we could select node pairs based on identical group membership, this allows us to get more high quality mappings for Phase 1. Identifying node mappings in a phase helps reduce the number of node pairs to be tested in subsequent phases hence it is desirable to identify a large number of nodes in Phase 1, t_2 and t_3 are identical for all datasets except for Enron we chose a lower t_3 to increase coverage and enable comparison. We studied the effect of threshold selection by varying t_1 in the range 30 ± 9 in steps of 3 and t_2 in the range 9 ± 3 in steps of 2; as discussed varying t_3 influences the number of node pairs under attack hence we do not change it (see, Section 3.3). Such variations impacts the run-time of **3PSL** but has little effect on the overall results.

We set vector length and bin size (n, b) to be (7,50) and (30,35) for directed and undirected graphs respectively (see, Section 3.2.1). The value of (n, b) is chosen such that it can accommodate higher degrees and their variation, the choice does not have a huge impact on the results [19]. The cosine similarity threshold of 0.2 which filters out most spurious matches; the remaining matches are *cleaned* using the classifier. We experimented by setting the value to 0.3,0.4,0.5 but did not see appreciable change in the results; setting it too low defeats the purpose. Non-identical node pairs

with sufficiently high degree tend to have very low cosine similarity. Testing accuracy increases monotonically with the forest size [51–53]; Criminisi *et al.* [51] obtain strong results with forest size of 400 trees, which is what we use. Phase 1 mappings stabilize after about 10 iterations which can be increased further to ensure stability as each iteration is cheap at this stage. Subsequent phases do not require many iterations due to sufficient mappings being found in Phase 1.

Feature selection. We experimented by using features such as centrality, edge weights and group membership in addition to those proposed. Complicated features do not provide significant improvement over those used. Using the perfect knowledge of groups barely impacts the ROC curve; this is due to the fact that identical group membership is already captured to a large extent in the 2-hop neighborhood degree distribution. Exploiting group membership to improve attacks (**NKA**) is useful for schemes that only use local features but it does little in our case. Moreover, the knowledge of group membership is not likely to be precise in reality which further dampens their effect. Even if using complicated features were to provide a significant improvement, it would not matter a lot in the grand scheme of things. We use the success of classifier to select node pairs that are more likely to be true, hence beyond a certain point the classification quality does not make a major difference. It is more important to improve the filtering process (see, Section 3.3) as it makes the algorithm more efficient by improving the ratio of true vs false node mappings and decreasing the number of mappings to be classified. We also experimented with features beyond two hops but this increased false positives, as using large subgraphs to represent a node produces overlaps with many other nodes.

Robustness of 3PSL

Due to similar properties of social networks learning acquired from training statistical models are transferable across datasets [36]. This is one the key reasons behind robustness of **3PSL** which emanates from the success of the machine learning task (see, Section 3.2) at its core. The learning algorithm obtains a high TP at low FP and thus a ROC curve with a high AUC even when trained on Epinions (nodes = 75 879, edges = 508 837) to attack Pokec[1] (nodes = 1 632 803, edges = 30 622 564) and vice-versa [19]. Thus the learning algorithm transcends the quirks of the datasets and learns the de-anonymization function which allows it to attack a dataset when trained on another. The closer the source and target graph distribution the better the performance which is the primary reason behind our re-purposing of the anonymized datasets for training (see, Section 3.2.2) and simulates a practical adversary; but even cross-training using social networks of completely different nature produces very strong results.

Hence, even if auxiliary and sanitized graphs have different densities, sizes, average node degree etc., the algorithm can still attack them. The training is agnostic to the size of graphs as discussed above. If the graphs are too small (less than a few thousand nodes) or too sparse then this will impact training and the attack success, however, this would come at the cost of utility. Given the sum of total number of nodes in both graphs is constant the task is hardest for the adversary when the graphs are of equal size because this maximizes the number of node pairs to be tested. The learning model tries to distinguish between identical and non-identical node pairs. Our experiments show that such pairs are *very* different. Hence, even when G_{aux} and G_{san} are generated differently the extreme difference between the node pairs would persist unless a huge amount of perturbation is introduced. Additionally, random edge deletion is one of the most damaging ways to perturb graphs [20]. Generating auxiliary and sanitized graphs differently might make the attack even more potent if the graph perturbation algorithm preserves more information than random edge deletion.

Furthermore, as demonstrated by Sharad [20] the learning task which forms the core of **3PSL**, can handle a variety of perturbation-based social graph de-anonymization schemes – ranging from random edge deletion (see, Section 3.1), randomly adding and deleting edges [13, 54, 55], randomly switching edges [40, 54–56], randomly deleting edges and inserting non-edges [57], making the graph k-degree anonymous [58] and making the graph 1-hop k-anonymous [59]. No other attacks are known to work let alone be as effective for such a diverse set of threat models, social graph datasets and graph perturbation schemes. The improvements are a result of optimal utilization of an adversary's resources by using machine learning models instead of sub-optimally and arbitrarily hand-picking graph heuristics and algorithm attack parameters.

7. CONCLUSION

Adversarial machine learning and de-anonymizing behavioral patterns are two sides of the same coin and they are converging fast. In the presence of big data, attacks based on heuristics will gradually be replaced by learning models because of their adaptability, automation and superior performance. Not only are the automated models better but the learning is transferable across datasets [19, 36], this provides a significant improvement over the traditional techniques. It is not surprising that learning models can surpass human intuition although the margin of improvement is startling. Our work shows how to approach the problem of social graph de-anonymization in a systematic manner. We present an algorithm that outperforms all the other graph de-anonymization algorithms proposed so far while using much stringent threat models without seed knowledge. The algorithm is not dependent upon heuristics for its success and uses the simple classification task of categorizing node pairs as identical or non-identical across graphs. It uses modular features that can adapt to a variety of threat models. The machine learning model does not require knowledge of the anonymization scheme for training. De-anonymization is simple and efficient as confirmed by evaluation based on three real-world social graph datasets under four threat models. A thorough comparison with seven modern de-anonymization attacks using two datasets is also presented. Our algorithm achieves a coverage of 12.61% and accuracy of 41.91% for Enron dataset, the next best algorithm is seed-based and achieves a coverage and accuracy of 15.96% which is substantially lower. The difference is even more stark for Facebook where our algorithm achieves a coverage of over 40% and an accuracy of over 65%, the next best algorithm is seed-based and can only achieve a coverage and accuracy of 28.32%. Our attack shows a marked improvement over all other attacks. Optimizing parameters by training is better in adverse scenarios as human error is costlier in limited information.

The attack presented needs to evaluate all node pairs between auxiliary and sanitized graphs in order to discover node mappings, this makes the attack slightly expensive but it is still very practical. Machine learning approaches tend to suffer when the training data is insufficient e.g. for sparse graphs with low average node degree; on the contrary such graphs are of limited use and hard to attack in general. Future work could try to improve upon these limitations.

Acknowledgments. We thank George Danezis, Laurent Simon and Ross Anderson for their comments on the draft manuscripts. This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/J500665/1]; and Microsoft Research through its PhD Scholarship Programme.

References

[1] Orange, "D4D Challenge." http://www.d4d.orange.com, 2012.

[2] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pp. 111–125, 2008.

[3] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pp. 901–909, VLDB En-dowment, 2005.

[4] C. Dwork, "Differential privacy," in *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pp. 1–12, 2006.

[5] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *ACM Trans. Database Syst.*, vol. 39, pp. 22:1–22:33, Oct. 2014.

[6] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing graphs using differentially private graph models," in *Pro-ceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, (New York, NY, USA), pp. 81–98, ACM, 2011.

[7] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, (Washington, DC, USA), pp. 169–178, IEEE Computer Society, 2009.

[8] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *Proceedings of the 1st ACM SIGKDD International Conference on Privacy, Security, and Trust in KDD*, PinKDD'07, (Berlin, Heidelberg), pp. 153–171, Springer-Verlag, 2008.

[9] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang, "Anonymizing bipartite graph data using safe groupings," *Proc. VLDB Endow.*, vol. 1, pp. 833–844, Aug. 2008.

[10] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, "Resisting structural re-identification in anonymized social networks," *Proc. VLDB Endow.*, vol. 1, pp. 102–114, Aug. 2008.

[11] A. Campan and T. M. Truta, "Data and structural k-anonymity in social networks," in *Privacy, Security, and Trust in KDD, Second ACM SIGKDD International Workshop, PinKDD 2008, Las Vegas, NV, USA, August 24, 2008, Revised Selected Papers*, pp. 33–54, 2008.

[12] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava, "Class-based graph anonymization for social network data," *Proc. VLDB Endow.*, vol. 2, pp. 766–777, Aug. 2009.

[13] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, "Anonymizing social networks," *Computer Science Department Faculty Publication Series*, p. 180, 2007.

[14] P. Mittal, C. Papamanthou, and D. X. Song, "Preserving link privacy in social network based systems," in *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[15] B. Thompson and D. Yao, "The union-split algorithm and cluster-based anonymization of social networks," in *Proceed-ings of the 4th International Symposium on Information, Com-puter, and Communications Security*, ASIACCS '09, (New York, NY, USA), pp. 218–227, ACM, 2009.

[16] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn, "Community-enhanced de-anonymization of online social networks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 537–548, ACM, 2014.

[17] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social network as a side-channel," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 628–637, ACM, 2012.

[18] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, (Washington, DC, USA), pp. 173–187, IEEE Computer Society, 2009.

[19] K. Sharad and G. Danezis, "An automated social graph de-anonymization technique," in *Proceedings of the 13th Work-shop on Privacy in the Electronic Society*, WPES '14, (New York, NY, USA), pp. 47–58, ACM, 2014.

[20] K. Sharad, "True friends let you down: Benchmarking social graph anonymization schemes," tech. rep., Feb 2016.

[21] A. Narayanan, E. Shi, and B. I. P. Rubinstein, "Link prediction by de-anonymization: How we won the kaggle social network challenge," in *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*, pp. 1825–1834, 2011.

[22] L. Backstrom, C. Dwork, and J. M. Kleinberg, "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pp. 181–190, 2007.

[23] N. Korula and S. Lattanzi, "An efficient reconciliation algorithm for social networks," *Proc. VLDB Endow.*, vol. 7, pp. 377–388, Jan. 2014.

[24] L. Yartseva and M. Grossglauser, "On the performance of percolation graph matching," in *Proceedings of the First ACM Conference on Online Social Networks*, COSN '13, (New York, NY, USA), pp. 119–130, ACM, 2013.

[25] S. Ji, W. Li, N. Z. Gong, P. Mittal, and R. A. Beyah, "On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge," in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*, 2015.

[26] S. Ji, W. Li, M. Srivatsa, J. S. He, and R. A. Beyah, "Structure based data de-anonymization of social networks and mobility traces," in *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, pp. 237–254, 2014.

[27] S. Ji, W. Li, M. Srivatsa, and R. Beyah, "Structural data de-anonymization: Quantification, practice, and implications," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 1040–1053, ACM, 2014.

[28] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, "A practical attack to de-anonymize social network users," in *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pp. 223–238, 2010.

[29] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser, "A bayesian method for matching two similar graphs without seeds," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, pp. 1598–1607, 2013.

[30] P. Pedarsani and M. Grossglauser, "On the privacy of anonymized networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 1235–1243, ACM, 2011.

[31] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pp. 289–298, 2008.

[32] X. Wu, X. Ying, K. Liu, and L. Chen, "A survey of privacy-preservation of graphs and social networks," in *Managing and Mining Graph Data* (C. C. Aggarwal and H. Wang, eds.), vol. 40 of *Advances in Database Systems*, pp. 421–453, Springer US, 2010.

[33] E. Zheleva and L. Getoor, "Privacy in social networks: A survey," in *Social Network Data Analytics* (C. C. Aggarwal, ed.), pp. 277–306, Springer US, 2011.

[34] B. Zhou, J. Pei, and W. Luk, "A brief survey on anonymization techniques for privacy preserving publishing of social network data," *SIGKDD Explor. Newsl.*, vol. 10, pp. 12–22, Dec. 2008.

[35] L. Babai and E. M. Luks, "Canonical labeling of graphs," in *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, (New York, NY, USA), pp. 171–183, ACM, 1983.

[36] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: Graph mining using recursive structural features," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 663–671, ACM, 2011.

[37] 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 *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 303–318, USENIX Association, Aug. 2015.

[38] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, (New York, NY, USA), pp. 29–42, ACM, 2007.

[39] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, (New York, NY, USA), pp. 556–559, ACM, 2003.

[40] X. Ying and X. Wu, "On link privacy in randomizing social networks," in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, (Berlin, Heidelberg), pp. 28–39, Springer-Verlag, 2009.

[41] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009.

[42] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, "Sok: The evolution of sybil defense via social networks," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pp. 382–396, 2013.

[43] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*, 2009.

[44] M. Egele, G. Stringhini, C. Krügel, and G. Vigna, "COMPA: detecting compromised accounts on social networks," in *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[45] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 363–374, Aug. 2010.

[46] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," *IEEE/ACM Trans. Netw.*, vol. 18, pp. 885–898, June 2010.

[47] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 267–278, Aug. 2006.

[48] G. G. Gulyás and S. Imre, "Measuring importance of seeding for structural de-anonymization attacks in social networks," in *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom 2014 Workshops, Budapest, Hungary, March 24-28, 2014*, pp. 610–615, 2014.

[49] K. Bringmann, T. Friedrich, and A. Krohmer, "De-anonymization of heterogeneous random graphs in quasilinear time," in *Algorithms - ESA 2014 - 22th Annual European Sym-posium, Wroclaw, Poland, September 8-10, 2014. Proceed-ings*, pp. 197–208, 2014.

[50] E. Kazemi, S. H. Hassani, and M. Grossglauser, "Growing a graph matching from a handful of seeds," *PVLDB*, vol. 8, no. 10, pp. 1010–1021, 2015.

[51] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2-3, pp. 81–227, 2012.

[52] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchor-age, Alaska, USA*, 2008.

[53] P. Yin, A. Criminisi, J. M. Winn, and I. A. Essa, "Tree-based classifiers for bilayer video segmentation," in *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, 2007.

[54] F. Bonchi, A. Gionis, and T. Tassa, "Identity obfuscation in graphs through the information theoretic lens," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, (Washington, DC, USA), pp. 924–935, IEEE Computer Society, 2011.

[55] X. Ying and X. Wu, "Randomizing social networks: a spectrum preserving approach," in *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pp. 739–750, 2008.

[56] X. Ying and X. Wu, "Graph generation with prescribed feature constraints," in *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pp. 966–977, 2009.

[57] M. Xue, P. Karras, R. Chedy, P. Kalnis, and H. K. Pung, "Delineating social network data anonymization via random edge perturbation," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, (New York, NY, USA), pp. 475–484, ACM, 2012.

[58] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *Proceedings of the 2008 ACM SIGMOD Inter-national Conference on Management of Data*, SIGMOD '08,(New York, NY, USA), pp. 93–106, ACM, 2008.

[59] B. Zhou and J. Pei, "The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks," *Knowledge and Information Systems*, vol. 28, no. 1, pp. 47–77, 2011.