An Efficient and Robust Social Network De-anonymization Attack

Gábor György Gulyás, Benedek Simon, Sándor Imre

An Efficient and Robust Social Network De-anonymization Attack

Gábor György Gulyás*
INRIA, France
gabor.gulyas@inria.fr

Benedek Simon
BME, Hungary
fleezeum@gmail.com

Sándor Imre
BME, Hungary
imre@hit.bme.hu

ABSTRACT

Releasing connection data from social networking services can pose a significant threat to user privacy. In our work, we consider structural social network de-anonymization attacks, which are used when a malicious party uses connections in a public or other identified network to re-identify users in an anonymized social network release that he obtained previously.

In this paper we design and evaluate a novel social de-anonymization attack. In particular, we argue that the similarity function used to re-identify nodes is a key component of such attacks, and we design a novel measure tailored for social networks. We incorporate this measure in an attack called Bumblebee. We evaluate Bumblebee in depth, and show that it significantly outperforms the state-of-the-art, for example it has higher re-identification rates with high precision, robustness against noise, and also has better error control.

Keywords

social network; privacy; anonymity; de-anonymization

1. INTRODUCTION

The rise of social networking services has paved the way for businesses monetizing social data; however, abuses and malintents emerged as quickly as legal opportunities. In parallel, the deep integration of social networking services into everyday life provides a never seen possibility for governments to extend their spying capabilities [4]. This asymmetry of powers between the users versus business parties and governments continuously urges the investigation of privacy issues beyond positive uses of social networks.

In this paper we consider the problem how the graph structure of a network alone can be abused to match users in different social networks, which often leads to the violation of user privacy. More specifically, we focus on large-scale re-identification attacks when two large networks are aligned together in order to de-anonymize one of them. As a result, each node in the anonymized network is either connected to a known identity in another network or not, meaning that the identity of the node is revealed (or it is de-anonymized).

There is a myriad of motivations for such attacks. For example a business party can be financially motivated in the de-anonymization of freshly bought anonymized datasets in order to update its previous records with the new data. Meanwhile, gluing several communication networks together can be an attractive target for government agencies (e.g., email and telephone communications). Beside several sucessful attacks on social networks [5,13–15,19–21,27], it has also been shown that location data can be de-anonymized with social networks as a background knowledge [14, 25].

The algorithm proposed by Narayanan and Shmatikov in 2009 (to which we refer as Nar) was the first to achieve large-scale re-identification in social networks [19]. Their work applied comparison of nodes based on previously discovered matching of neighboring nodes, meaning that instead of comparing each node in background knowledge to all in the anonymized graph (i.e., global comparison), their algorithm compared each node only to a smaller set of others that had been close enough to existing mappings (local comparison). This reduced computational complexity of the problem significantly, thus allowing large-scale re-identification in networks consisting of even more than a hundred thousand nodes. The authors in their main experiment re-identified 30.8% of co-existing nodes between a Twitter (224k nodes) and a Flickr (3.3m nodes) crawl with a relatively low error rate of 12.1%.

Let us now illustrate how the Nar attack works on a simple example. An adversary obtains datasets as depicted in Fig. 1, a dataset of known identities that he uses as a background knowledge (left), and an anonymized/sanitized dataset (right). The attacker's goal is to to learn political preferences by structural de-anonymization. Initially, the attacker re-identifies (or maps) some nodes that stand out globally from the dataset (the seeding phase). In our example, two top degree nodes are re-identified: `Dave` $\leftrightarrow$ `D` and `Fred` $\leftrightarrow$ `F` – as their degrees are outstandingly high and globally match each.

Then the attack continues by inspecting nodes related to the ones already re-identified (the propagation phase). The attacker picks a yet unidentified node in the background knowledge network. Let he first pick `Harry`, who has two neighbors already mapped: `Dave`, `Fred`. Nodes that could correspond to `Harry` in the anonymized dataset should also be neighbors of `D` and `F`, which are nodes `A`, `B`, `C`, `E`, `G`, `H`, and `I`. In the next step, similarity of these nodes are measured to `Harry`, which is calculated by dividing the number of their common mapped neighbors with their degree. This results a similarity score of 1 for nodes `B`, `I`; $1/\sqrt{2}$ for nodes `A`, `C`, `G`; $2/\sqrt{3}$ for node `E`; and $2/\sqrt{2}$ for node `H`. As node `H` alone has the highest score, the attacker selects `Harry` $\rightarrow$ `H`. However, for precaution, the attacker uses the same technique to check which node would be the best match (in the public dataset)

*Corresponding author.

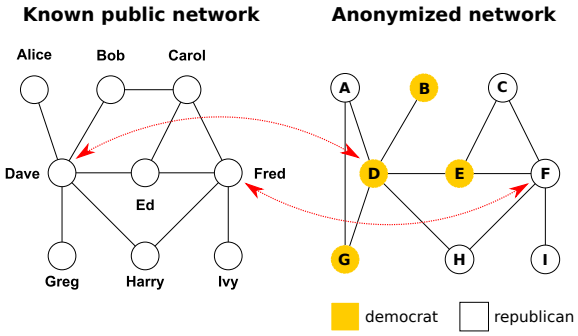


Figure 1: Datasets for the example of de-anonymization with an initial re-identification mapping between some nodes.

for `H`. If he gets `Harry` ← `H` in this case as well, the new mapping `Harry` ↔ `H` is registered.

While the similarity metric used by Nar provides a good balance between correct and incorrect re-identification matches, it is biased towards low degree nodes. For instance, if the attacker picks `Ed` first, he cannot re-identify him correctly: node `H` would have a higher score than node `E` (with $2/\sqrt{3}$). Therefore we argue that selecting the right metric for comparison is a critical part of the algorithm, as it determines the performance and final limits of the attack. We propose an alternative metric, for which we show by evaluation that it significantly improves the state-of-the-art. Our main contributions are the following:

- We create a novel structural re-identification algorithm called *Bumblebee*, which is based on a new similarity metric specifically designed for the current re-identification task. We show that improving this critical point enhances several properties of the attack significantly. Parameters of Bumblebee allow a more fine-grained control than the Nar algorithm due to the larger score diversity.
- We evaluate the performance, robustness of Bumblebee and its sensitivity of initialization. We show that Bumblebee can reach significantly higher correct de-anonymization rates while having the same or lower error rate as of Nar. In our experiments, we observe an increase of 28%-50% of correct re-identifications while error remains the same. Inversely, error rates can also be decreased close to zero. We measured error rates below 0.2% when having similar proportions of correct re-identification rates. We also show that Bumblebee is significantly more robust to noise than Nar, as it achieves large-scale correct re-identification in cases where other attacks could not rise above 1%. For seeding, we found that only $1-2$ seed nodes are enough for successful attacks, significantly smaller compared to previous works.
- We compare the performance of Bumblebee to other modern de-anonymization attacks, and show that its performance is the most outstanding compared to the state-of-the-art, even against anonymized datasets as well.
- We publish the source code of Bumblebee along our framework called Structural Anonymity Lab (`SALab`). Although we used earlier versions of it in our previous works, we now release `SALab` as an open source framework that supports the experimental analysis of re-identification attacks in social networks. It guarantees the reproducibility of our results.

The paper is organized as follows. In Section 2, we discuss the related work, and in Section 3, we provide the methodology of our evaluation. Section 4 provides the details of the Bumblebee algorithm, and in Section 5, we evaluate our algorithm and compare it to other ones as well. Finally, in Section 6, we conclude our work.

2. RELATED WORK

Several attacks have followed Narayanan and Shmatikov [19], and now we discuss the most relevant ones [5,13–15,18, 20,21,25,27]. These are heuristic approaches for solving the de-anonymization problem as finding the optimal solution is not computationally feasible [13]. The attacks may also differ in many details, albeit they share some common properties. Generally, they consist of two sequentially executed phases: one for initialization (seed phase with global node comparison), and another that extends re-identification in an iterative manner (propagation phase with local node comparison).

This is not a strict rule, some attacks run their propagation phase for seed generation [13,21], we call these self-starting attacks. Other attacks need seeds to start, which we call seed-based attacks [5,14,15,18–20,25,27]. Another important property of attacks is how they compare nodes. Some attacks use simple node properties for measuring node similarity [5,15,18,19,27], such as Nar uses cosine similarity, while others require deeper structural properties [13,14, 21,25], like distance vector comparison where shortest path distances are considered from already mapped nodes [25].

Narayanan et al. published also the first attack following [19], that consisted of a simplified version of the original attack [18]. Srivatsa and Hicks showed on small datasets that location traces can be re-identified by matching them to social networks [25], and other algorithms were proposed to this problem by Ji et al. [14], verified on the same datasets. Pham et al. proposed an algorithm that converts spatiotemporal data into a social network on large datasets [22]. Building upon their work, Ji et al. confirmed that location data can be efficiently re-identified with social networks [11,13]. Despite their original use, algorithms of [13,14,25] can also be used on attacking regular social networks.

The algorithm designed by Nilizadeh et al. uses other attacks as a base algorithm, and it exploits the cluster-oriented structure of the networks: runs the base re-identification algorithm first on the cluster structure, then inside them [20]. In their evaluation they used the Nar algorithm. Pedarsani et al. proposed the first self-starting attack that required no seeding [21]. The works of Yartseva and Grossglauser [27], and the paper of Korula and Lattenzi [15] contain simplified de-anonymization attacks in order to enable formal analysis of the algorithms. In our previous work we proposed Grasshopper [5] (also referred to as Grh), a robust attack algorithm that works with very small error rates (typically less then 1%).

The first comparative evaluation of several structural de-anonymization attacks were first provided in [12]. Seven algorithms in [13–15,19,21,25,27] were selected based on

generality, scalability and practicality, were compared regarding robustness of background knowledge and against anonymization. Ji et al found that some algorithms were the most prominent only conditionally; they concluded that none of the evaluated algorithms could be considered as generally better than others. To the best of our knowledge, we neither know about previously published attacks that have been proven to be better in general than the Nar algorithm. Thus, we considered the Nar algorithm as the baseline attack in our work, and we included Grasshopper also, as this attack has appealing properties, like fairly high re-identification rates in general with error rates almost at zero [5]. However, we also provide comparison based on our own measurements against results published in [12].

To the extent of our knowledge, the only structural re-identification framework that is similar to `SALab`, is SecGraph [12]. The benefit of SecGraph is that it implements several re-identification, anonymization and utility algorithms. Unfortunately, it does not implement several important functions that are necessary for large-scale experimentation, such as synthetic dataset perturbation (e.g., as in [19]), seed generation algorithms, and detailed evaluation of results (e.g., saving every minute detail while attacks are executed). `SALab` had these functions and attacks implemented already, as previous versions of it has been used in multiple works: for the analysis of the importance of seeding [9], for measuring anonymity [7,10], for evaluating the Grasshopper attack [5], and for analyzing numerous settings for adopting identity separation for protecting privacy [6,8,10]. Therefore we decided to choose `SALab` as our main tool.

3. NOTATION AND METHODOLOGY

We use the following notations in our work. Given an anonymized graph $G_{tar} = (V_{tar}, E_{tar})$ (target graph) to be de-anonymized by using an auxiliary data source $G_{src} = (V_{src}, E_{src})$ (where node identities are known), let $\tilde{V}_{src} \subseteq V_{src}, \tilde{V}_{tar} \subseteq V_{tar}$ denote the set of nodes mutually existing in both. Ground truth is represented by the mapping $\mu_G : \tilde{V}_{src} \rightarrow \tilde{V}_{tar}$ denoting the relationship between coexisting nodes. Running a deterministic re-identification attack on (G_{src}, G_{tar}) initialized by a seed set $\mu_0 : V^{\mu_0}_{src} \rightarrow V^{\mu_0}_{tar}$ results in a re-identification mapping denoted as $\mu : V^{\mu}_{src} \rightarrow V^{\mu}_{tar}$. Nodes are denoted as v_i, its set of immediate neighbors are denoted as $V_i \subseteq V$, and its neighbors of neighbors are denoted as $V_i^2 \subseteq V$.

In our work, we use simple measures for assessing the extent of what the attacker could learn from μ. The *recall rate* reflects the ratio of correct re-identifications: it is the number of correct re-identifications divided by the number of mutually existing nodes ($\tilde{V}$). The *error rate* is the proportion of incorrectly re-identified nodes compared to $\tilde{V}$; however, as there might be erroneous mapping outside $\tilde{V}$, this leaves the possibility of the error rate to emerge above 100%. We also use *precision*, which is the proportion of recall divided by the mapping size ($|\mu|$).

Data preparation and simulation parameters are key parts of methodology. We used multiple, large datasets with different characteristics in order to avoid related biases. We obtained two datasets from the SNAP collection [3], namely the Slashdot network (82k nodes, 504k edges) and the Epinions network (75k nodes, 405k edges). The third dataset is a subgraph exported from the LiveJournal network crawled in 2010 by us (66k nodes, 619k edges). Datasets were obtained from real networks which assures that our measurements were being realistic.

We used the perturbation strategy proposed by Narayanan and Shmatikov [19] to generate synthetic graph pairs for our experiments. Their method only uses deletion, thus the original graph acts as a ground truth between G_{src}, G_{tar}. Their algorithm makes two copies of the initial graph as G_{src}, G_{tar}, then performs independent deletions of nodes to achieve the desired fraction of overlap (denoted as α_v), and then deletes edges independently to set the size of overlapping edges (to α_e). Overlap parameters α_v, α_e are measured by the Jaccard similarity[1]. These parameters control the level of noise in the data, therefore they are also a representation of the strength of anonymization and the quality of the adversarial background knowledge. Unless noted otherwise, we used the setting of $\alpha_v = 0.5$, $\alpha_e = 0.75$, where a significant level of uncertainty is present in the data and thus harder to attack.

The default simulation settings were set as follows. In each experiment we created two random perturbations, and run simulations once, as algorithms provided deterministic results when we used the same seed set (e.g., see `top` below). When seeding was random, we run the attacks multiple times. Nar and Grh both have an important parameter Θ that controls the ratio of correct and incorrect matches (greediness): the lower Θ is the less accurate mappings the algorithm will accept. As we measured fairly low error rates even for small values of Θ, we have chosen to work with $\Theta = 0.01$ for both algorithms.

The seeding method and size have been showed to have a strong effect on the overall outcome of de-anonymization [9]. In [9] we provided details for various methods, and highlighted seeding methods that were top performers on large networks, regardless of the network structure, e.g., nodes with the highest degree and betweenness centrality scores. Considering these results, we applied two seeding methods. Our prime choice was selecting an initial mapping between top degree nodes (denoted as `top`), as this method proved to be the most effective in previous measurements in terms of run-time and smallest seed set size [9]. It can also be used in real attacks, as a weighted graph matching technique has been proposed for `top` seeding in [18]. Seed sizes were typically $50 - 200$ for `top`. In some cases, we also considered random seed selection with high degree nodes, where nodes are selected from the top 1% by degree (denoted as `random.01`).

[1] $\text{JaccSim}(U, V) = \frac{|U \cap V|}{|U \cup V|}$.

4. THE BUMBLEBEE ATTACK

4.1 The Motivation

Biases of the similarity measure of Nar and Grh, such as we discussed in the introduction, motivated the invention of the Bumblebee attack (Blb). We now illustrate these biases and the new scoring function we propose on the source node $v_i \in V_{src}$ and possible mapping candidates denoted as $v_j \in V_{tar}$. The similarity measure in Nar is a simplification of the cosine similarity [19] as

$$\mathrm{NarSim}(v_i, v_j) = \frac{|V_i \cap V_j|}{\sqrt{|V_j|}}, \tag{1}$$

where only a division with $\sqrt{|V_i|}$ is omitted, as this would be constant while comparing v_i to any other nodes (n.b. $|V_i \cap V_j|$ is the number of common mapped neighbors). Unfortunately (1) shows biases towards low degree nodes when the degree of v_i, v_j differ significantly.

We demonstrate this on the example as shown in Fig. 2a. Here, we assume a situation where each algorithm has to make a decision, having two nodes, namely v_A, v_B, to find the correct mapping for. We assume a significant difference in node degrees, having $\deg(v_A) = \deg(v_{A'}) = 100$, and $\deg(v_B) = \deg(v_{B'}) = 2$, and also have 5 mappings already existing in μ of which 2 overlap. Given this situation, we calculated similarities and the resulting decisions, see it in Fig. 2b. As NarSim penalizes high degree nodes, the re-identification algorithm cannot re-identify v_A.

The Grasshopper algorithm uses a similarity measure that weights mappings [5]:

$$\begin{aligned}\mathrm{GrhSim}(v_i, v_j) &= \sum_{\substack{\forall (v', v'') \in \{V_i \cap V_j\} \\ \text{where } \mu(v') = v''}} \omega[v'] \\ &= \sum_{\substack{\forall (v', v'') \in \{V_i \cap V_j\} \\ \text{where } \mu(v') = v''}} \left(\frac{|V' \cap V''|}{\sqrt{|V'| \cdot |V''|}} + 1 \right)\end{aligned} \tag{2}$$

where ω is the weight of each mapping in μ. Both ω, μ are updated at the end of every iteration. This measure is biased towards high degree nodes as the sum increases faster with a higher number of connections rather than having a smaller number of high degree adjacent nodes. Thus, GrhSim cannot discover node v_B as we demonstrate this with the similarity matrix shown in Fig. 2c (all weights are equal: $\forall \omega_i = 1$).

This leads to the need of a measure which yields higher values when degree values are similar. We propose the following similarity measure to remedy the situation:

$$\mathrm{BlbSim}(v_i, v_j) = \left| V_i \cap V_j \right| \cdot \left(\min \left(\frac{|V_i|}{|V_j|}, \frac{|V_j|}{|V_i|} \right) \right)^{\delta}, \tag{3}$$

where δ is a parameter that can be chosen freely as $\delta \in [0, 1]$. This similarity measure is more balanced than the previous ones when node degrees differ, as it is symmetric and yields higher score when node degrees are closer to each other. BlbSim can correctly re-identify both v_A and v_B as in Fig. 2d.

4.2 Evaluation of BlbSim

Let us now evaluate if BlbSim is in fact better than NarSim, and if so, under what conditions. When two nodes are compared with the discussed similarity metrics, we use the already discovered mappings in μ to glue the source and target networks together. Here, by using the mappings as additional edges, $v_i \in V_{src}$ and all possible candidates $v_j \in V_{tar}$ are reckoned as neighbors of neighbors in the same graph. In this setting, we assume that if nodes v_i and v_j correspond to each other ($\exists \mu_G(v_i) = v_j$) then v_i, v_j should be structurally similar (to some extent). An appropriate similarity measure $S(\cdot)$ should yield the highest score for $S(v_i, v_j)$ than for all other possible candidates. We can also describe this with the following correct re-identification decision criteria:

$$\mathrm{S}(v_i, v_j) > \mathrm{S}(v_i, v_j') + \Theta, \tag{4}$$

where $v_i \in V_{src}, v_j, v_j' \in V_{tar}, v_j \neq v_j'$. The ratio between correct and false decisions depend on the value chosen for $\Theta \in [0, \infty)$.

Due to the way existing mappings are used, this is similar to the problem when we compare two nodes in the same graph $v_A, v_B \in V$. As we expected correct decisions in case of re-identification (eq. (4)), we expect the similarity measure $S(\cdot)$ to give higher score for v_A to itself, then of any other node $v_B \in V \setminus \{v_A\}$, which is written as

$$\mathrm{S}(v_A, v_A) > \mathrm{S}(v_A, v_B) + \Theta \tag{5}$$

Let us call this the *graph node structural comparison problem.*

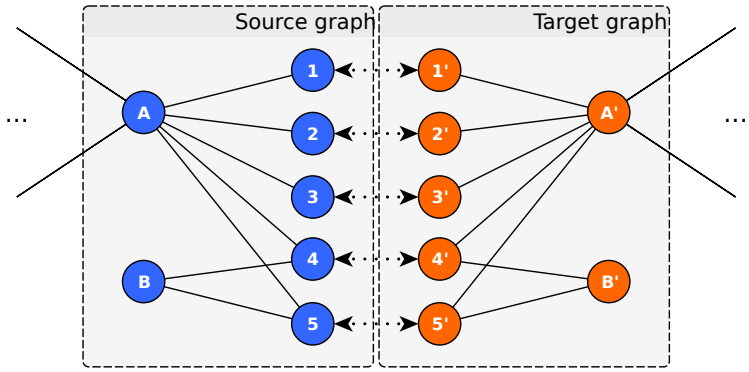
THEOREM 1. *Given the graph node structural comparison problem, having $\Theta = 0$ and $\delta \geq 1/2$, BlbSim$(\cdot)$ leads to a higher number of correct decisions than NarSim$(\cdot)$.*

We provide the proof for this theorem in the Appendix. While we restricted ourselves to some parameter settings in Theorem 1, later we show with experiments that Θ and δ parameters can be used together to achieve settings where the attack will exhibit interesting properties, e.g., providing high recalls or having extremely low error rates. Furthermore, as the scoring schemes output a different distribution, for BlbSim parameter Θ gives a significantly richer control of greediness (cf. in Fig. 6b).

4.3 Attack Description

We now discuss how the complete attack works based on BlbSim, for which we provide the pseudocode as in Algo. 1 and 2. After obtaining the seed set (μ_0), the attack starts by iteratively calling PROPAGATE, and it stops when no new mappings can be registered (when $\Delta = 0$). In each propagation step all nodes in V_{src} are checked for new mappings. Existing mappings are visited again, as this allows correction after discovering more mappings. Then we calculate the BlbSim between v_{src} and all possible candidates in SCORE by using δ (Algo. 1 line 3). SCORE selects candidates by iterating unmapped neighbors of neighbors through μ (Algo. 2 lines 2-7).

If there is an outstanding candidate v_c (Algo. 1 line 4), we do a reverse checking to decrease false positives. We test if we swap the graphs and invert the mapping, would v_{src} be the best candidate for v_c (Algo. 1 lines 8-12). If it is, we



(a) Here, $\deg(v_A) = \deg(v_{A'}) = 100$, and $\deg(v_B) = \deg(v_{B'}) = 2$ and 5 mappings already exist in μ.

NarSim	A'	B'	?
A	$\frac{5}{\sqrt{100}}$	$\frac{2}{\sqrt{2}}$	B'
B	$\frac{2}{\sqrt{100}}$	$\frac{2}{\sqrt{2}}$	B'

(b) NarSim decisions

GrhSim	A'	B'	?
A	5	2	A'
B	2	2	N/A

(c) GrhSim decisions with $\forall\omega_i = 1$ ($\omega_i \in \omega$)

BlbSim	A'	B'	?
A	5	0.89	A'
B	0.89	2	B'

(d) BlbSim decisions with $\delta = 0.5$

Figure 2: Example for demonstrating biases in the similarity measures.

Algorithm 1: PROPAGATE

Data: G_{src}, G_{tar}, μ
Result: μ, Δ

```
1  Δ ← 0;
2  for v_src ∈ V_src do
3  |  S ← SCORE(G_src, G_tar, v_src, μ);
4  |  if ECC(S.VALUES()) < Θ then
5  |  |  CONTINUE;
6  |  end
7  |  v_c ← RANDOM(MAX(S));
8  |  S_r ← SCORE(G_tar, G_src, v_c, μ^-1);
9  |  if ECC(S_r.VALUES()) < Θ then
10 |  |  CONTINUE;
11 |  end
12 |  v_rc ← RANDOM(MAX(S_r));
13 |  if v_src = v_rc then
14 |  |  μ[v_src] ← v_c;
15 |  |  Δ ← Δ + 1;
16 |  end
17 end
```

Algorithm 2: SCORE

Data: $G_{src}, G_{tar}, v_{src}, \mu$
Result: S

```
1  S ← DICT(∀v ∈ V_tar : v → 0);
2  for v_i ∈ G_src.NBRS(v_src) do
3  |  if ∄μ(v_i) then
4  |  |  CONTINUE;
5  |  end
6  |  for v_j ∈ G_tar.NBRS(μ(v_i)) do
7  |  |  if ∃μ^-1(v_j) then
8  |  |  |  CONTINUE;
9  |  |  end
10 |  |  r_1 = DEG(v_src)/DEG(v_j);
11 |  |  r_2 = DEG(v_j)/DEG(v_src);
12 |  |  S[v_j] ← S[v_j] + (MIN(r_1, r_2))^δ;
13 |  end
14 end
```

register the new mapping and indicate convergence (Algo. 1 lines 14, 15). We say that a node with maximum score in S is outstanding if its similarity score is higher than all others at least by Θ:

$$\text{ECC}(S) = \frac{\max(S) - \max(S \setminus \{\max(S)\})}{\sigma(S)} \geq \Theta. \quad (6)$$

An off-the-shelf working version of this attack is available in `SALab`. Bumblebee inherited some parts of Nar which we found to be natural and good design choices, such as candidate selection through mappings and reverse checking to eliminate false positives. However, we use a greedy convergence criteria triggered by even the smallest changes (Algo. 1 line 15), and we redesigned the scoring function (SCORE) for eliminating biases. Albeit we use the same eccentricity function as Nar, due to the wider scoring diversity of BlbSim, this plays a more important role in Blb, and allows a more fine-grained control of the algorithm (cf. in Fig. 6b).

5. EVALUATION

In this section we evaluate the Bumblebee algorithm, and compare it to other attacks. Although we use multiple settings of Bumlbebee, parameters Θ and δ do remain free to choose. Best settings may depend on several factors, such as the size and structure of the dataset under analysis. We use the notation as Blb(δ) (when Θ is fixed) or as Blb(Θ, δ).

As we discussed before, we use `SALab` which enables analyzing various aspects of large-scale structural attacks. We also make it available at https://github.com/gaborgulyas/salab; we hope it will contribute to further research helping to have a better understanding of structural anonymity.

Our simulations were run on a commodity computer with an Intel Xeon 3.6 GHz processor and 8 GB RAM.

5.1 Experimental Analysis of δ and Θ

First we evaluate how parameters δ and Θ influence the overall outcome. We used `top` seeding with 200 nodes and $\Theta = 0.01$, meaning a greedy setting in case of Nar [19] and Grh [5] (we provide analysis of Θ for Bumblebee later), safe enough for all algorithms to provide high recall rates [5,

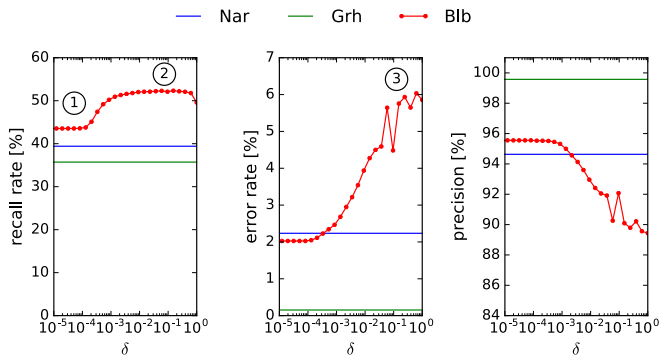


Figure 3: Effect of the δ parameter on results (Slashdot dataset).

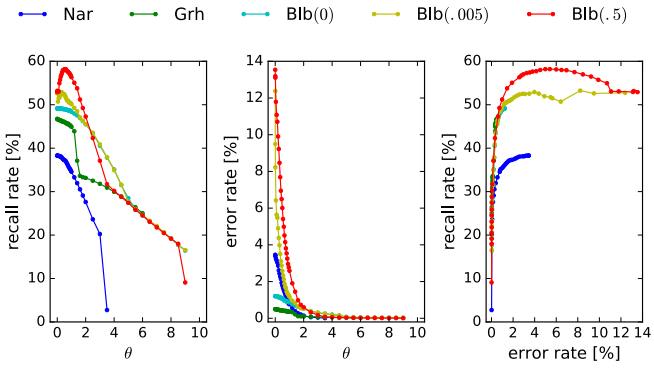


Figure 4: Effect of the Θ parameter for different algorithms in the LiveJournal dataset.

9]. Results for a wide range of δ values in the Slashdot dataset are visible in Fig. 3. Due to space limitations, we select results of one network for demonstration; however, until noted otherwise, we were getting similar results in the other networks as well.

Interestingly, we observed transition phases of the recall and error rates as a function of δ, which divided results into three interesting parameter subclasses for Blb. We used these classes for analyzing parameter Θ. In the first class (denoted with (1)) we observed lower recall rates and low error. We use such a setting as $\delta = 0$ and denote it as Blb(0). This conservative setting makes decisions only based on the number of common mappings without considering any other information. Another classes are where the recall rates plateau after increasing significantly ((2)). Here we selected two settings, one when recall peaked but error is relatively low, and the other when the error is also peaked ((3)). We observed that the recall rate usually at its highest at $\delta = 0.005$ while error rate is still lower, thus we selected Blb(.005), and finally we selected Blb(.5) also.

We provide comparison on the effect of the Θ parameter for the LiveJournal dataset in Fig. 4. What clearly stands out from these results, that Blb could achieve the highest recall rates, while it also had higher errors for the same Θ settings compared to Nar and Grh. Furthermore, the conservative setting of Blb(0) managed to achieve higher recall than Grh with similarly low error, and in most cases higher recall as Nar, but with a lower error rate. Another interesting property of Bumblebee is that it achieved high recall rates as Nar with extremely low error, e.g., Blb($\Theta = 4.0, \delta = 0.5$) recall = 29.91%, error = 0.15%; in other cases it achieved a significantly better trade-off between recall-error rates, as the error/recall results suggest. Comparing the recall-error results additionally shows that results provided by Grh, Blb(0) are subclasses of the results of Blb(.5). The only difference is they provide the same (recall, error) results at different values of Θ. Finally, Blb(.005) is clearly worse than Blb(.5).

From these results, we conclude that both parameters of Bumblebee enable the control of greediness in the algorithm by taking values as $\Theta \in [0, \ldots, \infty)$, $\delta \in (0, ..., 1]$. Conservative (low error) settings are typically with high Θ and low δ, greedy settings are with the opposite settings. In addition, parameters can also be used compensate each other, like as Blb(1, .5) provided high recall with low error in Fig. 4. We used these measurements to set parameters for further experiments.

5.2 Analysis of Seeding Sensitivity

The number of required seeds is an important aspect of seed-based attacks, as these need to be provided before starting propagation, or in another sense, before starting the attack. Thus, the smaller the seed size is, the better. We consider the minimum required number of seeds for which the attack is stable: observed recall and error rates only vary minimally and the recall rate reaches its maximum [9]. We provide an example on the minimum number of seed nodes with `top` in Fig. 5 on Epinions dataset. Beside getting better recall and error rates than others, some Bumblebee variants managed to start with only a single seed node.

In other networks 1-2 seed nodes were enough to achieve large-scale re-identification with a higher error rate, which dropped when the number of seeds reached 5-6. We observed similar results when relaxing the seed criteria to `random.01`, multiple successful seeding attempts with a single random seed for Bumblebee. This means top 1% of high degree nodes, random selection from approx. a set of 200-300 nodes.

To the best of our knowledge, these results are superior to existing previous works, which reported minimum number of seeds at 16 seed nodes [20], or $8 - 14$ seed nodes [5]. In addition, relaxed seed criteria means easier attacks: one randomly selected node from the top 1% as a seed is an easily doable task for an attacker; even with some retries or collecting a handful of seeds in order to achieve large-scale propagation.

5.3 Robustness

The quality of the background knowledge, or the overlap between G_{src} and G_{tar}, has a strong influence on the limits of re-identification. For example, the Grh algorithm has typically higher recall rates than Nar when the overlap is smaller [5]. In order to investigate such biases, we measured the performances of the algorithms with different perturbation settings. We note that almost all overlaps in these measurements are significantly lower than in experiments of Section 5.4.

We provide results in the Epinions network in Fig. 6a. A greedy setting of Bumblebee, Blb(.5, .5) had the highest recall rates in all cases, with an acceptable level of error except in two settings. A more conservative setting, Blb(.01, 0)

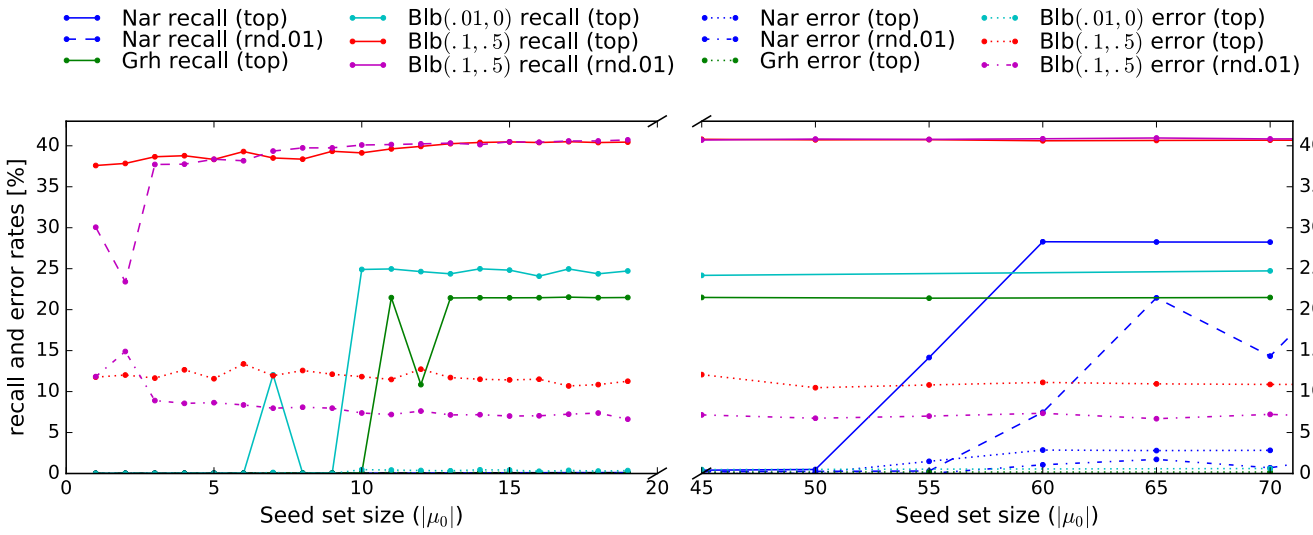
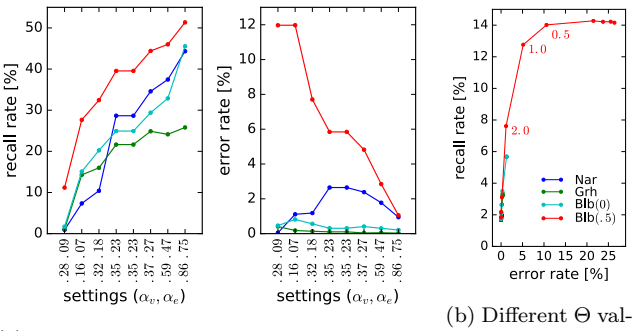


Figure 5: Seed sizes of `top` and `random.01` required for large-scale re-identification (Epinions dataset).



(a) Performances for different perturbation settings. (overlap values are as actually measured, not as set for perturbation)

(b) Different Θ values used on $\alpha_v = 0.28, \alpha_e = 0.09$ setting.

Figure 6: Robustness comparison of different algorithms (Epinions dataset); recall and error rates for different perturbation settings.

provided recall rates close to Nar (and higher to Grh), but with negligible error.

In experiments with small overlap, error rates of Blb(.5, .5) were higher in case of all three networks and while staying low for Nar (as of no propagation); for Blb it reached 12% in case of Epinions. These cases were with perturbation settings $\alpha_v = 0.25, \alpha_e = 0.75$ and $\alpha_v = 0.5, \alpha_e = 0.25$. However, these are the perturbation parameters, not actual values: deleting nodes delete some edges and removing edges sometime totally disconnect nodes. Therefore we displayed the actual overlaps on the x-axis in Fig. 6a, which are measured with the Jaccard similarity [19]. In these special cases the background knowledge and anonymized networks shared typically $\leq 10\%$ of overlapping edges in all three networks.

In case of $\alpha_v = 0.28, \alpha_e = 0.09$ only Blb(.5, .5) could achieve large-scale propagation, but with high error. This can be decreased by setting parameters δ or Θ. With the latter, we could achieve recall = 12.7% with an error only at 5.1% ($\Theta = 1.0$), or recall = 7.6%, error = 1.1% ($\Theta = 2.0$); see more details in Fig. 6b.

We furthermore note that Fig. 6b also provides evidence that parameter Θ allows significantly more fine-grained control in Bumblebee than in the Nar algorithm. Fig. 6b is also an example how one can search the parameter space for the desired trade-off between recall and error by fixing one of the parameters.

5.4 Comparison with Other Attack Algorithms

We can now compare the performance of the Bumblebee algorithm with other modern attacks. As [12] provides a recent selection and comparison of efficient structural social network de-anonymization attacks, we compare Bumblebee to these attacks under the same settings. The source code of the attacks are released with the SecGraph tool [2]. Therefore, we obtained the same datasets, the Enron [3] (36k nodes, 183k edges) and Facebook [1] networks (63k nodes, 817k edges). With the sampling based perturbation method (implemented in `SALab`) we recreated the datasets G_{src}, G_{tar} as of [12]: we only needed to randomly sample edges with probability s, while nodes were kept intact as much as possible. We note that this perturbation technique does notably smaller damage to the data compared to how we created synthetic datasets; for example, the lowest setting for Enron with $s = 0.6$ corresponds to $\alpha_v = 0.77, \alpha_e = 0.42$, or Enron $s = 0.95$ corresponds to $\alpha_v = 0.97, \alpha_e = 0.90$ (cf. x-axis in Fig. 6a).

According to the results of Table 6 in [12], we selected top performing attacks for comparison. For brevity, we simply provide the list of these attacks without details: *percolation graph matching* (later referred as YG) [27], *distance vector matching* (DV) [25], *reconciliation attack* (KL) [15]. We used 50 `top` seeding (as in [12]), and run the Nar and Blb(.1, .5) attacks from `SALab`. We run the three other attacks from the SecGraph implementation on the generated datasets. As both frameworks output the final the mapping μ, we could safely use this for our comparison.

We obtained results as in Fig. 7. For Nar and YG we obtained almost identical results as in [12], while recall for DV

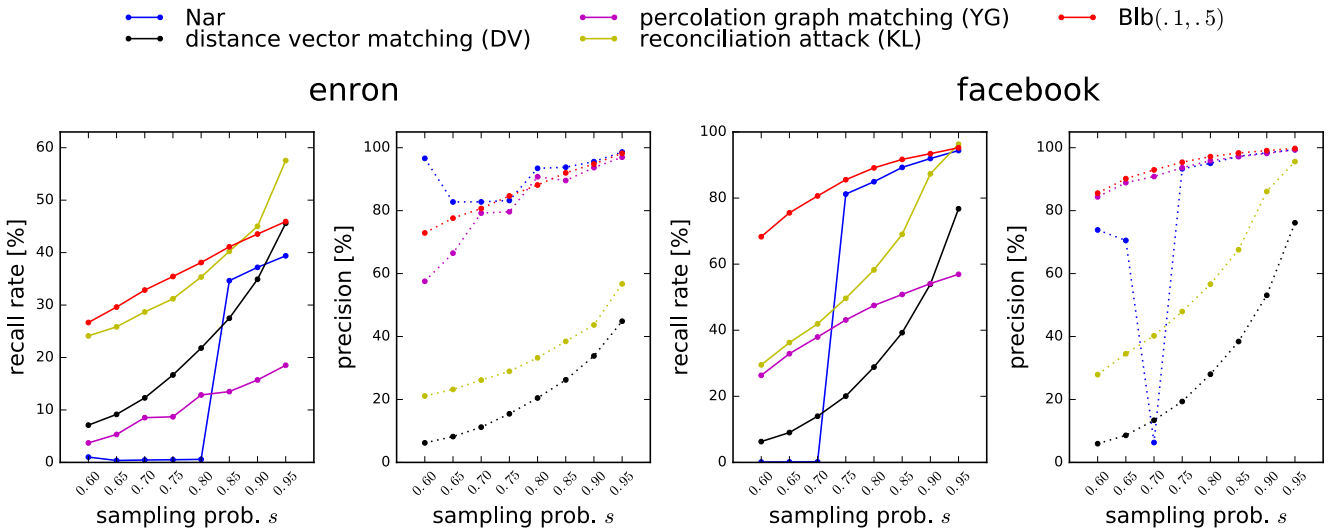


Figure 7: Comparison of efficient de-anonymization algorithms. We used the same datasets, settings and code as in [12], based on which we selected outstanding attacks in [15, 25, 27] to provide a comparison with Bumblebee.

were consistently smaller, and for KL significantly higher. The latter differences can be due to configurational settings however, we tried several configurations to achieve best results for all attacks. Here, we observed that Bumblebee provided a balanced, high performance in all settings, outperforming all other attacks.

In [12] only recall rates are concerned for describing the performance of de-anonymization attacks, while we found that error rates divide attacks into two strongly separated classes; we found that Nar, YG, Blb were high and DV, KL were low precision attacks. This finding puts high recalls into another perspective, as related error rates between 40-100% are unacceptably high. Therefore, while the KL attack had higher recall rates than Blb for Enron $s = 0.95$, with the 43.9% error rate KL had a very poor precision, unlikely to be acceptable for an attacker.

We also evaluated robustness of attacks against anonymization schemes as a comparison of Bumblebee for results in Table 8 in [12] (while keeping scheme parameters identical). Again, for brevity, we only enlist the schemes we included: *random edge switching* (Switch) [28], *k-degree anonymization* (k-DA) [16], and a method providing *differential privacy* (DP) [23]. We left out a clustering approach due to unreasonably long runtimes [26], and a random walk based graph regeneration method [17] – against the latter we could not achieve recall rates higher than 1% with any of the algorithms.

Here, we first sampled the background knowledge network from the original datasets with probability s, and obtained the anonymized network by running the anonymization algorithms also on the original datasets (no sampling). We observed robustness of evaluated attacks as in Table 1. In case of anonymization, Blb proved to be generally the most robust attack, as it always achieved successful attacks with the high recall rates compared to others with high precision. It had error rates around 1-2%, exceptionally 5-6% only for Enron, DP $\epsilon = 50$. The latter was exceptional as this was the only case when another attack provided higher recall than Bumblebee; however, KL provided very poor precision with error rates between 61.7%-70.2%.

These results show that Bumblebee is generally a balanced, robust, high performing attack with low error rates, outstanding of the state-of-the-art. Beside, while Bumblebee had runtimes typically between 5-10mins in each case, not all others were that fast, for example, KL had runtimes between 12-24h.

5.5 Complexity and Runtime

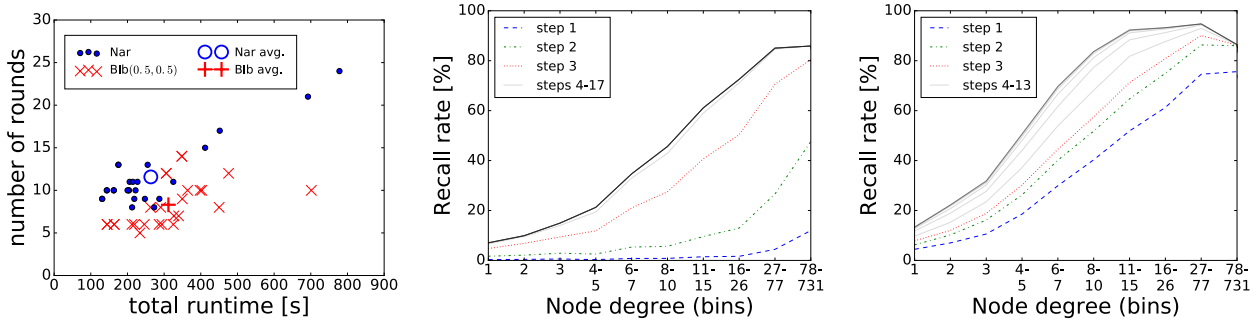
Runtimes of Bumblebee depend mainly on the complexity of propagation iterations and speed of convergence. The complexity of the Bumblebee propagation phase is $O(d_{src} \cdot d_{tar} \cdot (|V_{src}| + 1))$, where d_{src}, d_{tar} denotes maximum degree in each graph. The algorithm iterates over $\forall v \in V_{src}$. We assume having mapping $\tilde{\mu}$ when inspecting v, and the algorithm selects the neighbors-of-neighbors of v. More precisely, neighbors are $\forall v' \in V \subseteq V_{src}$ having $|V| \leq d_{src}$, and their neighbors are the neighbors of the mapped counterparts of v, the neighbors of $v'' = \tilde{\mu}(v')$. Here $V'' \subseteq V_{tar}$ is upper bounded as $|V''| \leq d_{tar}$. This leads us to $O(|V_{src}| \cdot d_{src} \cdot d_{tar})$. We need to add the reverse checking complexity ($O(d_{tar} \cdot d_{src})$) which leads us to the given bound.

However, this does not necessarily captures the total runtime, as it significantly depends on the speed on the convergence, i.e., how fast the algorithm converges to its final result in terms of the number of new re-identifications in each propagation round [5, 6]. In addition, run-times need to be handled with caution and these should be only compared when Nar and Blb achieve high re-identification rates; otherwise, significant differences could occur. For instance, the number of propagation rounds could be less for the algorithm that has lower recall as it finishes before.

In order to avoid such biases, we compared runtimes and the number of propagation rounds where both algorithm achieved very high recall rates both (see in Fig. 8a). We found that Blb(.1, .5) converges faster in a lower number of rounds compared to Nar, as its convergence profile is quite

Table 1: Robustness of attacks against different anonymization schemes from [16, 23, 28]. Values are recall rates [%] (denoted as Rc) and precision [%] (Pr) of each algorithm under the given settings. Settings and parameters are intentionally replicating results of [12] in order to enhance comparability of results.

		Enron												Facebook											
	s	Switch(k)				k-DA (k)				DP (ϵ)				Switch (k)				k-DA (k)				DP (ϵ)			
		5		10		5		50		300		50		5		10		5		50		300		50	
		Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr
Nar	.85	34.9	94.3	0.5	89.9	39.1	98.2	1.0	94.6	38.2	98.0	0.5	97.3	89.1	96.6	82.8	93.6	94.0	98.9	93.8	98.8	93.5	98.6	87.1	95.8
	.90	36.1	95.6	0.7	91.1	40.6	98.9	37.7	97.7	39.1	98.8	0.5	97.2	90.0	97.2	84.4	94.7	94.7	99.2	94.6	99.2	94.5	99.1	89.7	97.5
	.95	37.0	96.5	30.3	94.8	41.0	99.2	38.6	98.7	40.2	99.4	0.6	98.9	91.2	97.9	85.6	95.6	95.5	99.7	95.4	99.7	95.2	99.6	91.0	98.5
YG	.85	13.6	90.0	11.2	86.3	16.8	94.7	1.3	24.4	15.5	92.0	9.7	76.2	51.9	97.8	47.5	96.4	55.7	99.0	48.8	92.1	54.1	98.3	45.7	89.8
	.90	15.0	93.4	12.8	91.1	15.2	90.8	1.7	29.1	16.9	95.2	10.2	77.5	53.4	98.3	49.3	96.9	56.6	99.1	52.3	94.7	55.2	98.4	48.7	92.7
	.95	16.4	94.9	13.8	92.1	17.4	95.0	4.8	55.1	18.2	96.7	9.7	75.8	54.7	98.8	51.2	98.2	57.9	99.4	54.3	95.9	56.1	98.3	48.4	91.5
DV	.85	13.2	13.2	8.2	8.2	13.2	13.2	4.0	4.0	13.8	13.7	0.3	0.3	17.7	17.7	9.3	9.3	25.5	25.5	5.6	5.6	18.3	18.3	8.7	8.7
	.90	15.4	15.4	9.4	9.4	15.2	15.2	4.8	4.8	14.9	14.9	0.3	0.3	22.3	22.2	11.7	11.7	31.7	31.7	7.1	7.1	23.6	23.6	11.1	11.1
	.95	19.2	19.2	11.8	11.8	19.6	19.6	5.8	5.8	17.8	17.7	0.3	0.3	32.1	32.1	16.9	16.9	47.6	47.6	12.2	12.1	39.4	39.4	12.4	12.4
KL	.85	28.7	28.7	26.6	26.6	29.9	29.9	20.5	20.5	30.1	30.1	29.8	29.8	66.0	66.0	63.0	63.0	67.1	67.1	66.4	66.4	67.0	67.0	73.2	73.2
	.90	31.2	31.2	28.6	28.6	32.9	32.9	23.2	23.2	32.2	32.2	34.0	34.0	75.3	75.3	71.2	71.2	84.9	84.9	84.2	84.2	84.0	84.0	81.4	81.4
	.95	37.0	37.0	35.0	35.0	38.2	38.2	32.4	32.4	36.5	36.5	38.3	38.3	94.4	94.4	91.6	91.6	97.2	97.2	96.4	96.4	97.0	97.0	79.8	79.8
Blb	.85	42.5	93.5	38.9	90.8	45.2	97.3	42.8	94.5	43.8	96.1	27.8	80.3	92.1	98.5	88.2	97.3	94.8	99.2	94.4	98.9	94.5	99.1	89.2	97.0
	.90	44.2	95.7	39.9	93.2	46.4	98.3	44.8	96.7	45.5	97.5	29.0	83.3	93.2	99.1	89.9	98.4	95.4	99.6	95.3	99.5	95.3	99.6	91.2	98.2
	.95	45.1	97.5	41.2	96.3	47.3	99.3	46.1	98.5	46.4	98.8	32.0	84.7	93.8	99.6	91.0	99.2	96.0	99.9	95.9	99.9	95.8	99.8	91.9	98.5



(a) Number of propagation rounds and related runtimes.

(b) Re-identification convergence of Nar over degree classes.

(c) Re-identification convergence of Blb(0.1, 0.5) over degree classes.

Figure 8: Blb converges faster in a lower number of rounds compared to Nar.

different (cf. Fig. 8b and 8c). Blb re-identifies the majority of nodes in the first propagation round and then converges for some rounds, while Nar has a slower convergence. This is also shown by the average number of propagation rounds, which was 11.5 for Nar and 8.3 for Blb, while each round turned to be longer for Blb. The average final runtimes were $t_{Blb} = 311$ secs and $t_{Nar} = 263$ secs, which are notable, but not dramatic differences.

6. CONCLUSION

In this paper, we considered structural social network de-anonymization attacks, where we argued that node similarity metric plays a critical part. In fact, in order to improve that critical step, we proposed a new metric designed specifically for social networks, which we incorporated our new attack called Bumblebee.

We evaluated the new attack in different real-life networks and under multiple settings. We showed that Bumblebee outperforms the baseline attack in multiple aspects, as it is robust to noise and when attacker background knowledge is weak. We have also shown improvement in the minimum required size of seed sets, achieved high recall rates. Bumblebee allows a fine-grained control over the trade-off between yield and error; which either allows fixing error rates for reaching higher recall than other attacks, or decreasing the level of error.

We have also compared Bumblebee to the other state-of-the-art attacks, and found that Bumblebee was highly efficient under all circumstances, even against different anonymization schemes. Reaching high recall with high accuracy in all cases, with balanced performance against noise and anonymization attempts make Bumblebee the most outstanding attack within the state-of-the-art.

Acknowledgements

The authors would like to thank Amrit Kumar, Gergely Ács and Cedric Lauradoux for reviewing draft versions of this paper and for their valuable comments and discussions. Our work is based on the idea of Benedek Simon [24]. This work was carried out during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme.

7. REFERENCES

[1] The koblenz network collection. http://konect.uni-koblenz.de. Accessed: 2016-04-20.

[2] Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. http://www2.ece.gatech.edu/cap/secgraph/dataset.html. Accessed: 2016-04-20.

[3] Stanford network analysis platform (snap). http://snap.stanford.edu/. Accessed: 2014-04-22.

[4] What nsa's prism means for social media users. http://www.techrepublic.com/blog/tech-decision-maker/what-nsas-prism-means-for-social-media-users/. Accessed: 2014-05-26.

[5] S. Benedek, G. G. Gulyás, and S. Imre. Analysis of grasshopper, a novel social network de-anonymization algorithm. *Periodica Polytechnica Electrical Engineering and Computer Science*, 58(4):161–173, 12 2014.

[6] G. G. Gulyás. *Protecting Privacy Against Structural De-anonymization Attacks in Social Networks*. PhD thesis, CrySyS Lab, Budapest University of Technology and Economics, May 2015.

[7] G. G. Gulyás and S. Imre. Measuring local topological anonymity in social networks. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pages 563–570, 2012.

[8] G. G. Gulyás and S. Imre. Hiding information in social networks from de-anonymization attacks by using identity separation. In B. Decker, J. Dittmann, C. Kraetzer, and C. Vielhauer, editors, *Communications and Multimedia Security*, volume 8099 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin Heidelberg, 2013.

[9] G. G. Gulyás and S. Imre. Measuring importance of seeding for structural de-anonymization attacks in social networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 0–6, 2014.

[10] G. G. Gulyás and S. Imre. Using identity separation against de-anonymization of social networks. *Journal of Transactions on Data Privacy*, 8(2):113–140, 8 2015.

[11] S. Ji, W. Li, J. He, M. Srivatsa, and R. Beyah. Poster: Optimization based data de-anonymization, 2014. Poster presented at the 35th IEEE Symposium on Security and Privacy, May 18–21, San Jose, USA.

[12] 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 *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 303–318, Berkeley, CA, USA, 2015. USENIX Association.

[13] 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, pages 1040–1053, New York, NY, USA, 2014. ACM.

[14] S. Ji, W. Li, M. Srivatsa, J. S. He, and R. Beyah. *Information Security: 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, chapter Structure Based Data De-Anonymization of Social Networks and Mobility Traces, pages 237–254. Springer International Publishing, Cham, 2014.

[15] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *Proc. VLDB Endow.*, 7(5):377–388, Jan. 2014.

[16] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 93–106, New York, NY, USA, 2008. ACM.

[17] P. Mittal, C. Papamanthou, and D. Song. Preserving link privacy in social network based systems. 2012.

[18] 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*, pages 1825–1834, 2011.

[19] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187, 2009.

[20] 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, pages 537–548, New York, NY, USA, 2014. ACM.

[21] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser. A bayesian method for matching two similar graphs without seeds. In *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, pages 1598–1607, Oct 2013.

[22] H. Pham, C. Shahabi, and Y. Liu. Ebm: an entropy-based model to infer social strength from spatiotemporal data. In *Proceedings of the 2013 international conference on Management of data*, pages 265–276. ACM, 2013.

[23] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 81–98, New York, NY, USA, 2011. ACM.

[24] B. Simon. Analysis and development of structural de-anonymization algorithms (in hungarian). Master's thesis, Budapest University of Technology and Economics, Budapest, Hungary, 2015.

[25] 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, pages 628–637, New York, NY, USA, 2012. ACM.

[26] B. Thompson and D. Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 218–227, New York, NY, USA, 2009. ACM.

[27] 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, pages 119–130, New York, NY, USA, 2013. ACM.

[28] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, volume 8, pages 739–750. SIAM, 2008.

APPENDIX

A. PROOF FOR THEOREM 1

PROOF. We expect that equation (5) will hold for NarSim. Substituting equation (1) into (5):

$$\frac{|V_A|}{\sqrt{|V_A|}} > \frac{|V_A \cap V_B|}{\sqrt{|V_B|}} \tag{7}$$

Similarly with (3), we get the following for BlbSim:

$$|V_A| > |V_A \cap V_B| \cdot \left(\min \left(\frac{|V_A|}{|V_B|}, \frac{|V_B|}{|V_A|} \right) \right)^{\delta} \tag{8}$$

We can rearrange these equations to the following form:

$$\frac{|V_A|}{|V_A \cap V_B|} > \frac{\sqrt{|V_A|}}{\sqrt{|V_B|}} \text{ and } \frac{|V_A|}{|V_A \cap V_B|} > \left(\min \left(\frac{|V_A|}{|V_B|}, \frac{|V_B|}{|V_A|} \right) \right)^{\delta}. \tag{9}$$

If NarSim scores give an upper-bound to BlbSim, then BlbSim will meet the criteria in (5) for more cases. This means that BlbSim would make more correct similarity comparisons than NarSim.

We can write the upper-bound criteria as

$$\left(\min \left(\frac{|V_A|}{|V_B|}, \frac{|V_B|}{|V_A|} \right) \right)^{\delta} \leq \frac{\sqrt{|V_A|}}{\sqrt{|V_B|}}. \tag{10}$$

This finally boils down to the following cases.

1. $|V_A| = |V_B|$: Here, (10) reduces to an an equality that holds for all values of δ.
2. $|V_A| > |V_B|$: We will get the following inequality

$$\frac{(|V_B|)^{\delta}}{(|V_A|)^{\delta}} \leq \frac{\sqrt{|V_A|}}{\sqrt{|V_B|}}, \tag{11}$$

where the left hand side is < 1, while the right hand side is > 1, thus equation (10) holds for all values of δ.

3. $|V_A| < |V_B|$: Similarly, we will get:

$$\frac{(|V_A|)^{\delta}}{(|V_B|)^{\delta}} \leq \frac{\sqrt{|V_A|}}{\sqrt{|V_B|}}, \text{ rearranged as } \left(\frac{|V_A|}{|V_B|} \right)^{\delta - 1/2} \leq 1. \tag{12}$$

Here, for $\delta < 1/2$, due to the negative exponent the left hand side will be > 1, while for $\delta \geq 1/2$ the left hand side will be ≤ 1.

This means, that equation (10) holds for all three cases when $\delta \geq 1/2$, which gets us to the statement of the Theorem. □