Analysis of Grasshopper, a Novel Social Network De-anonymization Algorithm

Benedek SIMON[1], Gábor György GULYÁS[*1,2], and Sándor IMRE[2]

[1]Laboratory of Cryptography and Systems Security (CrySyS), Dept. of Networked Systems and Services, BME

[2]Mobile Communication and Quantum Technologies Laboratory (MCL), Dept. of Networked Systems and Services, BME

December 17, 2014

Abstract

Social networks have an important and possibly key role in our society today. In addition to the benefits, serious privacy concerns also emerge: there are algorithms called de-anonymization attacks that are capable of re-identifying large fractions of anonymously published networks. A strong class of these attacks solely use the network structure

[*]Contact author email: gulyas@crysys.hu

to achieve their goals. In this paper we propose a novel structural de-anonymization attack called Grasshopper. By measurements we compare Grasshopper to the state-of-the-art algorithm, and highlight its enhanced capabilities, such as having negligible error rates and accessing yield levels that was not possible before: in cases when there is greater noise in the background knowledge. We furthermore evaluate an anonymity measure for the Grasshopper algorithm which enables the approximate ranking of nodes according to their re-identification rates. Finally, we characterize the robustness of Grasshopper in tackling identity separation, a privacy-enhancing technique that facilitate hiding of structural information.

1 Introduction

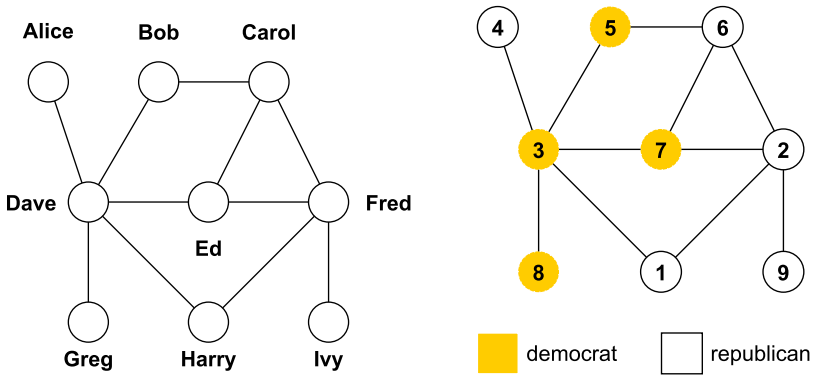
Most of social networking services provide interfaces for managing social relationships, while others focus on enabling the collaboration of their users. A useful feature of these services is that they are supported by an underlying (and occasionally only implicitly existing) graph structure. However, beside the values these services give to humanity, social media also serves as an optimal platform for all kinds of surveillance activities, as members can snoop upon each other, commercial parties can access vast amounts of private data, and as recent events confirm [4], government surveillance is also present as well. Therefore it is crucial to investigate privacy issues beyond the use of related settings.

In this paper we consider how the graph structure can be abused to violate user privacy. There are several ways to access anonymized datasets,

for example, someone can obtain such a dataset that was previously released for business or research purposes. While such a dataset should contain private attributes without explicit identifiers, a malicious third party can try to re-identify nodes by using their relationships. In case of success, the private information could be used (and monetized) with real identities. The basic idea for performing this type of attack is to use structural data from another social network to execute an iterative re-identification algorithm. Despite the difficult nature of the problem, several attacks have been published recently that are able to breach user privacy at large-scale even in networks having hundreds of thousands of nodes [21].

Let us now illustrate how these attacks work on a simple example. An adversary obtains datasets as depicted on Fig. 1a (background knowledge) and Fig. 1b (sanitized dataset), wishing to learn an otherwise inaccessible private attribute by structural de-annymization: who is a democrat or republican voter in the public network. Initially, the attacker re-identifies (or maps) $v_{Dave} \leftrightarrow v_3$ and $v_{Fred} \leftrightarrow v_2$ as they have globally the highest matching degree values in both networks. Then he continues with local re-identification by inspecting nodes related to the ones already re-identified. Therefore, he picks v_{Ed}, who is the highest degree common neighbor of (v_{Dave}, v_{Fred}), and then it is mapped as $v_{Ed} \leftrightarrow v_7$, as v_7 is the only node neighboring v_2, v_3 and have a degree of 3. This simple algorithm can continue iterating through unmapped nodes, resulting in discovering further possible mappings (e.g., $v_{Harry} \leftrightarrow v_1, v_{Carol} \leftrightarrow v_6$).

In this work, we propose a novel structural re-identification algorithm called Grasshopper. Beside providing the analysis of this attack, we also



(a) Public network (as auxiliary data) (b) Anonymized network

Figure 1: Datasets for the example of de-anonymization.

consider its robustness versus a privacy-enhancing method related to the identity partitioning technique [7,16,28,29], called identity separation. Identity separation allows a user to have multiple unlinkable profiles in the same network, which results in multiple unlinkable nodes in sanitized graphs also (i.e., as the service provider should also be unaware of the link between the identities). This could be imagined as a feature of having multiple registrations in parallel, where the ease of use is provided by software. Our simulation evaluation provides the model level analysis of identity separation in tackling the Grasshopper algorithm. Designing a system that supports identity separation in a private way is possible and feasible, and could benefit our work by incorporating useful strategies; however, it is a complex task, the detailed elaboration of such a system is beyond the scope of the current work.

Our main contributions in this paper are the following:

- We propose a novel structural re-identification algorithm called Grasshopper. We experimentally show that Grasshopper can achieve signifi-

cantly higher correct re-identification rates when the background knowledge of the attacker is noisy, what was not possible with the state-of-the-art attack Nar09 [21]. In addition, we show that Grasshopper has some significantly improved properties compared to Nar09: in our experiments we observed negligibly small error rates, and found that Grasshopper can be initialized with only a small fraction of nodes that was required for Nar09.

- We evaluate two anonymity measures for Grasshopper. Our findings show that the values produced by these measure have a strong correlation with re-identification rates (typically around $0.6 - 0.8$), and thus can be used to rank the nodes within the network to assess their level of anonymity.

- We characterize the robustness of the Grasshopper attack against identity separation. In particular, we evaluate two series of experiments on non-cooperative and cooperative identity separation. We show that non-cooperative strategies are practically ineffective in stopping the attack, but allow adopters of identity separation to minimize data leakage. We show that if cooperation can be organized, such settings can effectively preserve network privacy.

The paper is organized as follows. In Section 2, we discuss related work, and in Section 3, we provide the methodology of our evaluation. Section 4 provides the details of the Grasshopper algorithm, and in Section 5, we compare significant properties of our algorithm to the ones of the state-of-the-art attack, Nar09. In Section 6, we evaluate anonymity measures for Grasshop-

per. Identity separation, a potential tool for enhancing privacy against structural re-identification is evaluated against Grasshopper in Section 7. Finally, in Section 8, we conclude our work.

2 Related Work

2.1 Large-Scale De-anonymization Attacks

The algorithm proposed of Narayanan and Shmatikov in 2009 (Nar09) had a significant novelty compared to the literature discussed so far: it applied local comparison of nodes based on previously discovered matching of neighboring nodes [21]. The Nar09 algorithm aims to reveal the identities of nodes within a sanitized graph (the target graph) by using a social network obtained from an auxiliary source (the source graph). The authors in their main experiment re-identified 30.8% of nodes being mutually present in a Twitter and a Flickr crawl with a relatively low error rate of 12.1%.

Works following their approach also used a similar procedure; in most cases these consist of an initialization phase (or seed phase), which is then followed by a propagation phase. In general, the seeding identifies a small set of globally outstanding nodes, and then the propagation phase extends this set, for instance, by searching locally outstanding nodes that are connected to the set of already re-identified ones. These phases can also be named as global and local re-identification phases.

In their original experiment, the seeding is based on 4-cliques. The steps of the propagation phase are iterated on the neighbors of the nodes already re-

identified until new matchings can be discovered (i.e., it continuously extends the seed set). Identified nodes are also revisited. In each iteration, candidates are selected from target graph nodes, which share at least a common mapped neighbor with the source node being re-identified. Target candidates are then compared by scoring their similarity to the source node. If there is an outstanding candidate, the source and target graphs are exchanged, and a reverse checking is executed in order to verify the proposed mapping. If the result of reverse checking equals the source node, this is accepted as a valid mapping.

Narayanan et al. in 2011 presented another variant of their attack [20] specialized for the task of working on two snapshots of the same network, that could achieve a higher recall rate. Another proposal of Wei et al. [23] challenged Nar09; however, their attack is only evaluated against a light edge perturbation procedure, instead of the more realistic one proposed in [21]. The latter deletes both nodes and edges from both networks (resulting overlaps can be as low as 25%), while in [23] perturbation only adds edges to the target network (up to 3%) without any deletion. For a more comprehensive evaluation, their algorithms need to be compared with a perturbation method that includes deletion. In addition, experiments in [23] are performed on two small graphs consisting only of handful of nodes (graph vertex sizes are 125 and 600) – if it is feasible for the seed-and-grow algorithm, a comparison on larger datasets need to be done.

Pedarsani et al. proposed a novel type of attack that can work without any initial input such as seeds [22]. Their design incorporated seeding into the propagation phase, as the initial propagation step starts identifying top

degree nodes according to a given node fingerprint measure. However, their algorithm requires very high similarity between the source and target datasets (e.g., $\alpha_v = 1.0$ and $\alpha_e = 0.85$; for explanation, see Section 3.2), which is hard to meet in many cases. Additionally, their work was experimentally tested only on a single, small network with 2024 nodes and 25,603 edges.

Danezis and Sharad presented a generic deanonymization framework for the evaluation of anonymization schemes [25], which can be trained on a relatively small set of sanitized data. While their results can not be directly compared to global matching algorithms such as [21], their framework can be used for testing new sanitization schemes, such as identity separation (as future work).

It has been shown that even a relatively small amount of mobility data can easily identify users [19], and even short periods of surveillance enable identification [9]. However, it was first shown by Srivatsa and Hicks that location traces can also be re-identified with similar methods what was used for social networks [27]. In their work on small datasets (125 nodes and below), they succeeded in identifying circa. 80% of users by building anonymous networks of location traces, and using explicit social networks for de-anonymization.

The work of Pham et al. showed that the ability of algorithms using spatiotemporal data for making social network connections, can be extended to large datasets [24]. Building upon their work, Ji et al. showed that spatiotemporal data at the scale of hundred thousand entities can be easily re-identified [18]: first a social network is generated based on the inspection of co-occurrences in the spatio-temporal dataset, then it is re-identified by using a social network as auxiliary data.

As none of the attacks discussed here have been proven to be better in general (e.g., always have higher correct re-identification rates under the same circumstances) than the algorithm proposed by Narayanan and Shmatikov in [21] (e.g., due to lack of comparative experiments on large networks), we considered the Nar09 algorithm as the state-of-the-art attack at the time of evaluation Grasshopper.

2.2 Enhancing Privacy Contra De-anonymization

We consider user centered privacy protection mechanisms for preventing de-anonymization, ones that can be adopted in existing services (instead of graph sanitization applied by the service provider). For instance, Scramble is a good example for solutions being independent of the service provider and allowing a fine-grained access of social data [6]. Otherwise, one might consider using revised service models, such as distributed social networks like Safebook [8]; however, these services are more difficult to introduce.

Beato et al. proposed the friend-in-the-middle model, where proxy-like nodes serve as mediators to hide connections, enabling to repel the attack on a network level [5]. Their concept could be also implemented as an external tool that could be used in existing social networking services. The viability of the FiM model is presented (successfully) on two snapshots of the Slashdot network [3] (which we also used; for more details see Section 3). However, identity separation allows more than hiding connections, even hiding profile information beside relationships [7]. As this allows finer-grained management of information, with less cooperation – this can even enable the protection

of a single individual.

The concept of privacy-enhancing identity management was developed in details within the framework of the PRIME Project [17], including how identity partitioning and separation could be implemented in various contexts and services. The possible use of identity separation in social networks was introduced by us in [16], where we proposed a modified social network model with a non-flat structure. The works of van den Berg and Leenes in [28, 29] provided further details on identity partitioning, especially focusing on access control and division of information shared.

Previously, we have analytically showed that identity separation is an effective tool against clique based seeding mechanisms [11]. In subsequent works, we analyzed the protective strength of identity separation against the propagation phase of Nar09 [13, 15] with simulation on datasets obtained from three different social networks. In [13] we analyzed the non-cooperative setting, and we have shown that while almost half of the users are required to repel the attack (and retain network privacy), it is possible to effectively hide information from the attacker even for a few nodes if the proper settings are applied. In [15] we analyzed the cooperative setting organized accordingly to the importance of nodes, based on their anonymity values. The minimum number of required nodes to repel the attack dropped down to the fraction measured in the non-cooperative case.

In cases when a node is identified globally in a network, it is trivial (but likely ineffective) to measure its anonymity level, which usually proportional to the number of nodes with the same fingerprint, i.e., number of nodes being in the same anonymity set. However, in case of attacks like Nar09, nodes

are compared locally, and therefore anonymity sets can not be considered in the same sense. Furthermore, without knowing the background knowledge of the attacker, anonymity can be only estimated. To circumvent this problem we proposed local anonymity measures in [12]. These can be useful from a privacy-oriented point of view, as these can express the node's resistance level against local re-identification techniques, and these can also support data providers and attackers to make estimates of the possible success of attacks.

Our approach for measuring anonymity is called Local Topological Anonymity (LTA). We also proposed and evaluated multiple LTA variants for the Nar09 attack, selecting the most outstanding one denoted as LTA_A. In the evaluation of [12], we measured an average Pearson correlation [1] of -0.421 between LTA_A anonymity values and re-identification rates of nodes. As this evaluation was done for networks sized at most ten thousand nodes, we provided further evaluation on larger networks in [15]. We evaluated LTA_A in three networks having more than sixty thousands of nodes, and observed a Spearman rank correlation [2] typically around -0.65. Besides, we observed also similarly strong correlation values for node degree, denoted as LTA_{deg}.

Finally, we have shown that seeding parameters are an important aspect of the de-anonymization procedure, as they have a significant effect on the overall results [14]. Thus, it should be detailed both for comparing new attack schemes (e.g., [23]) and for evaluating protection mechanisms (e.g., [5, 13]). Therefore we analyzed our findings regarding this finding, too.

3 Notation and Method of Evaluation

3.1 Notation and Definitions

Given a sanitized graph G_{tar} (target graph) to be de-anonymized by using an auxiliary data source G_{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 mapping $\mu_G : \tilde{V}_{src} \to \tilde{V}_{tar}$ denoting relationship between coexisting nodes. Let us denote a vertex set V as V' after having identity separation adopted (by some or all of its nodes). We denote the set of nodes before adopting identity separation as $V_{ids} \subseteq V_{tar}$, and denote $\tilde{V}_{ids} \subseteq \tilde{V}_{tar}$ the subset coexisting nodes; thus $\tilde{V}'_{ids}$ contains multiple identities of nodes from $\tilde{V}_{ids}$. Let $\lambda_G : \tilde{V}_{src} \rightrightarrows \tilde{V}'_{ids}$ denote the ground truth mappings between coexisting nodes in G_{src} and the sets of their separated identities in G_{tar}. Running a deterministic re-identification attack on (G_{src} , G_{tar}) initialized by seed set $\mu_0 : V_{src} \to V'_{tar}$ results in a re-identification mapping denoted as $\mu : V_{src} \to V'_{tar}$.

Furthermore, we denote the corresponding nodes in different networks as $v_n^{src} \in V_{src}$ and $v_n^{tar} \in V_{tar}$. When *identity separation* of user $v_n^{tar} \in V_{ids}$ is committed, the user creates a total of y new partial identities which are denoted as $v_{n\backslash i} \in \tilde{V}'_{ids}$ ($i \in [1, \ldots, y]$), and then distribute edges between new identities. It is assumed that the attacker only captures the sanitized dataset after the user committed identity separation, and knows no information about the identity separation process itself.

We use two measures for assessing the extent of what the attacker could learn from μ. The *recall rate* reflects the extent of re-identification, describing

success from an attacker point of view. This itself can be used due to small error rates. As identity separation is a personal information hiding tool, the quantity of information the attacker gained access to should also be concerned, which is quantified by the *disclosure rate*. This describes an overall protection efficiency from a user point of view.

Now we can describe the mode of calculation of these rates. The *recall rate* is calculated by dividing the number of correct identifications with the number of mutually existing nodes (seeds are excluded from the results). The score of a node $v^{src} \in \tilde{V}_{src}$ regarding a given re-identification mapping μ can be expressed as:

$$s(v^{src}, \mu) = \begin{cases} 0 & \text{if } \nexists \mu(v^{src}) \\ 1 & \text{if } \mu(v^{src}) = \mu_G(v^{src}) \vee \mu(v^{src}) \in \lambda_G(v^{src}) \\ -1 & \text{if } \mu(v^{src}) \neq \mu_G(v^{src}) \wedge \mu(v^{src}) \notin \lambda_G(v^{src}) \end{cases} . \quad (1)$$

We can now quantify the *recall rate* of an attack resulting in mapping μ can be calculated as

$$R(\mu) = \sum_{\forall v^{src} \in \tilde{V}_{src}} \frac{s(v^{src}, \mu) \cdot \max(0, s(v^{src}, \mu))}{|\tilde{V}_{src}|}. \quad (2)$$

The *disclosure rate* can be calculated in a similar manner. As current identity separation models are bond to structural information, the measure reflects the average percent of edges that the attacker successfully revealed (this can be extended for further types of information in other experiments, e.g., sensitive profile attributes). The disclosed information can be quantified

for an individual node $v_n^{tar} \in \tilde{V}_{ids}$ as

$$d(v_n^{tar}, \mu) = \begin{cases} \frac{deg(v_{n\setminus i})}{deg(v_n^{tar})} & \text{if } \exists \mu(v_n^{src}) = v_{n\setminus i} \wedge v_{n\setminus i} \in \lambda_G(v_n^{src}) \\ 0 & \text{otherwise} \end{cases} . \quad (3)$$

By using this function we can now define the disclosure rate of the attacker over the nodes applying identity separation w.r.t. mapping μ as

$$D(\mu) = \sum_{\forall v_n^{tar} \in \tilde{V}_{ids}} \frac{d(v_n^{tar}, \mu)}{|\tilde{V}_{ids}|}. \quad (4)$$

The *re-identification rate of a node* v in a series of experiments ν is considered in some cases, which is calculated as

$$S(v) = \sum_{\forall \mu \in \nu} s(v, \mu), \quad (5)$$

where $s(v, \mu)$ can theoretically take arbitrary values in the series of ν.

3.2 Social Network Datasets and Perturbation

During our experiments we used multiple datasets with different characteristics in order to avoid related biases. In addition, we used large networks, as brute-force attacks can be mounted against smaller ones. We obtained two datasets from the SNAP collection [3], namely the Slashdot network crawled in 2009 (82,168 nodes, 504,230 edges) and the Epinions network crawled in 2002 (75,879 nodes, 405,740 edges). The third dataset is a subgraph exported from the LiveJournal network crawled in 2010 (at our dept.;

consisting of 66,752 nodes, 619,512 edges). All datasets were obtained from real networks in order to maintain our measurements being realistic.

In order to generate the test data, first we derived a background knowledge (G_{src}) and a target graph (G_{tar}), having desired overlap of nodes and edges, and then modeled identity separation on a subset of nodes in the target graph. For the first part, we used the perturbation strategy proposed by Narayanan and Shmatikov [21], as we found their method to be producing fairly realistic test data. Their algorithm takes the initial graph to derive G_{src}, G_{tar} with the desired fraction of overlapping nodes (α_v), and then edges are deleted independently from the copies to achieve edge overlap α_e. By knowing the original graph, the ground truth μ_G can be easily created at this point.

We found $\alpha_v = 0.5$, $\alpha_e = 0.75$ to be a good trade-off at which a significant level of uncertainty is present in the data (thus life-like), but the Nar09 attack is still capable of identifying a large ratio of the co-existing nodes. Fraction of correctly identified nodes are presented for various settings in all the test network in Table 1.

Due to the lack of real-world data, we used the probability based models we previously introduced in [11] for deriving test data from real-world datasets featuring identity separation (these models were also used in [13,15]). These models capture identity separation as splitting a node, and assigning previously existing edges to the new nodes. The number of new identities is modeled with a random variable Y (with no bounds on distribution), which we either set to a fixed value, or model it with a random variable having a power-law-like distribution. In our work it is assumed, that the identity

separation is done in secret, and can not be learned by the attacker from auxiliary sources.

For edge sorting, there are four models in [11] regarding whether it is allowed to delete (i.e., an edge becomes private) or to duplicate edges, from which we used three in our experiments. The basic model is simple and easy to work with, as it consists a simple redistribution of edges between the new identities (no edge deletion or duplication allowed). In order to represent privacy-oriented user behavior, we also used the best model, where no edge duplication is allowed, but edges can be deleted.

Identity separation is then modeled on the target graph by uniformly sampling a given percent of nodes with at least $deg(v) = 2$ (this ratio is maintained for the ground truth nodes), and then nodes are split and their edges are sorted according to the settings of the currently used model. This results in extending the ground truth mapping μ_G with λ_G by recording identity separation operations.

3.3 Calibrating Simulations

By comparing the directed and undirected versions of Nar09, we found little difference in results. Therefore, due to this reason and for sake of simplicity, in our experiments we used undirected networks. Additionally, in each experiment we created two random perturbations, and run simulations two times on both with a different seed set (unless different settings are noted). We observed only minor deviations in results, usually less than a percent.

Probably the most important parameter of Nar09 is Θ, controlling the

ratio of true positives (recall rate) and false positives (error rate). 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 $\Theta = 0.01$. In the majority of experiments the ground truth error rate (later referred as the error rate) stayed typically around a few percents. The overall error was around 5% without identity separation, and decreased significantly when identity separation was applied.

Another important property of the simulation the seeding method and size. In our previous work in [14] we provided details for various methods, and showed that the overall recall rate is influenced by several properties of seed nodes, such as the structural relation between them (e.g., cliquish structure or neighboring), and their global properties (e.g., node degree, betweenness centrality score). Results are also shown to be dependent on network size and structure. Our experiments highlighted seeding methods that were top performers on the large networks, regardless of network structure (e.g., nodes with highest degree and betweenness centrality scores).

Regarding these results, for simulating an attacker, we applied random seed selection with high degree nodes, where nodes are selected from the top 25% by degree (denoted as `random.25`). Seed set size was selected constantly for a thousand nodes, as this proved to be robust in all networks [14]. For the simulation of stronger attackers top degree nodes were selected as seeds (denoted as `top`), as this methods proved to be one of the most effective in our datasets [14].

4 The Grasshopper Algorithm

In this section we present the Grasshopper algorithm, that we developed based on the idea of the first author. Grasshopper have many similarities that resemble the Nar09 algorithm [21], however, there are also significant differences that need to be emphasized. The broad outline of Grasshopper is the same to Nar09: there is a propagation step that is iterated on the mapping between the two graphs until new mappings are registered. In each iteration, a node is selected and if there is an appropriate target candidate for mapping it, this is reversely checked. If the proposed mapping for the target candidate is the original node, the new mapping is accepted. Looking at the algorithms with a greater granularity level, differences emerge. The pseudo code of Grasshopper is presented in Algorithm 1, and now we present important features of Grasshopper.

Weighting mappings and scoring. We introduced a weighting scheme that is applied on existing mappings and denoted as ω. Nar09 used each existing mapping in μ with the same weight (i.e., 1.0), while we weight each mapping proportionally to number of mappings in their neighborhood. The intuition here is that mappings linking nodes having a significant number of mappings in their neighborhood should be considered as more valuable and are likely to be more accurate than others. We use ω in the scoring part (in the BestMatch function) instead of using the same score for each mapping.

Updating the mapping only once per iteration. The mapping μ is copied into η in each iteration step (could be interpreted as 'subsequent

mappings'), and only η is update during the propagation step; thus, new mappings are only considered in the next round. We note that this feature is not an absolute necessity, and turning it off leads to the greedy variant of the Grasshopper algorithm. This is slightly faster, but properties are less improved, e.g., there is a slight increase in the number of required seeds. For achieving the best results, we worked with the non-greedy setting.

Convergence criteria. Propagation is iterated while there is convergence in the results: new and important mappings are added in the propagation step. The Nar09 algorithm uses a similar criteria, and we found that it has a rather slow convergence: most mappings are found in the first few propagation steps, while there might be even tens of subsequent steps following when only a few new mappings are registered in each. This phenomena can also be observed in Grasshopper, however, the convergence is even slower. In order to rationalize our time needs, we stopped convergence if it reached 40 steps or run for more than 20 mins. These settings can be easily re-adjusted in other experiments (we found these to leading only negligible losses), and an attacker can simply ignore them to have the best results. In our future work we intend to introduce a mapping memory to find an improved convergence criteria.

Algorithm 1: Pseudo code of the Grasshopper propagation phase.

Require: Θ ▷ Threshold for accepting new matches.

1: **function** PROPAGATE(G_{src}, G_{tar}, μ_0)

2: $\mu \leftarrow \mu_0$

3: **repeat**

```
4:        (μ, Δ) ← PropagateStep(G_src, G_tar, μ)
5:    until Δ = 0
6: end function
7:
8: function PropagateStep(G_src, G_tar, μ)
9:    Δ ← 0
10:   ω_src ← {∀v_src ∈ V_src : v_src → 1.0}                ▷ Initialize weights.
11:   ω_tar ← {∀v_tar ∈ V_tar : v_tar → 1.0}
12:   for all v_src ∈ V_src  if  ∃μ(v_src) do
13:       for all v'_src ∈ G_src.NBRS(v_src) do
14:           if ∃μ(v'_src) ∈ G_tar.NBRS(μ(v_src)) then
15:               α ← SQRT(v_src.DEGREE() * μ(v_src).DEGREE())
16:               ω_src[v_src] ← ω_src[v_src] + 1.0/α
17:               ω_tar[μ(v_src)] ← ω_tar[μ(v_src)] + 1.0/α
18:           end if
19:       end for
20:   end for
21:   η = μ
22:   for all v_src ∈ V_src do                             ▷ Seek new possible matches.
23:       v_tc ← BestMatch(G_src, G_tar, ω_tar, v_src, μ)
24:       if v_tc ≠ None then
25:           v_sc ← BestMatch(G_tar, G_src, ω_src, v_tc, μ^-1)
26:           if v_sc = v_src  and  (∄μ(v_src)  or  ∃μ(v_src) ≠ v_tc) then
27:               η[v_src] ← v_tc
28:               Δ ← Δ + 1
```

```
            end if
        end if
    end for
    μ = η
    return (μ, Δ)
end function

function BestMatch(G_src, G_tar, ω, v_i, μ)
    S ← {}
    for all v'_i ∈ G_src.nbrs(v_i) if ∃μ(v'_i) do
        for all v'_j ∈ G_tar.nbrs(μ(v'_i)) do
            if v'_j ∉ S.keys() then
                S[v'_j] ← 0
            end if
            S[v'_j] ← S[v'_j] + ω[v'_j]
        end for
    end for
    if S.size() = 0 then
        return None
    end if
    if Eccentricity(S.values()) ≥ Θ then
        v_c ← pick(∀v ∈ S.keys() : S[v] = max(S.values()))
        return v_c
    end if
    return None
```

54: **end function**

55:

56: **function** ECCENTRICITY(S)

57: **return** $(max(S) - max_2(S))/\sigma(S)$

58: **end function**

5 General Evaluation

In this section we evaluate basic properties of the Grasshopper algorithm, and compare results to the the state-of-the-art attack, Nar09.

5.1 Characterizing Basic Properties

Parameter Θ has a vital role in both Nar09 and the Grasshopper algorithms, as the the best matching node is evaluated respecting Θ: it determines how outstanding a node should be to be accepted. In other words, this parameter controls the trade-off regarding the proportion of accurate and false matches. We measured the effect of Θ with several values, by using `random.25` seeding method. For Nar09 we used 1000 seeds and for Grasshopper we used 100, as these turned out to be stable for each (see related measurements below).

Our results are shown on Fig. 2, which tells two important findings. First, Grasshopper could achieve recall rates that was not possible with Nar09. However, this is not true in general, but in some cases the difference is quite significant (recall rates for multiple networks are depicted on Fig. 3a). Second, which is a general improvement, the error rate is only a fraction compared to the Nar09. This means that parameter Θ has a less

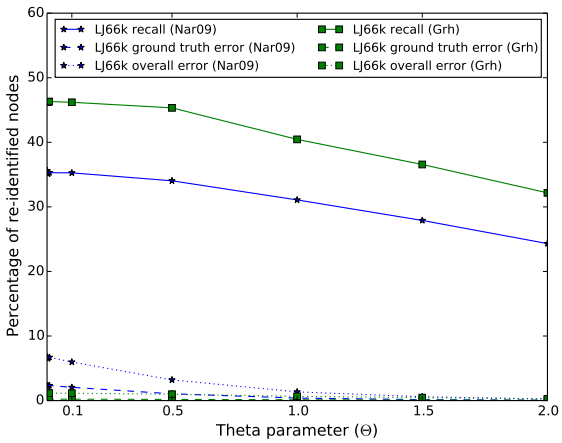
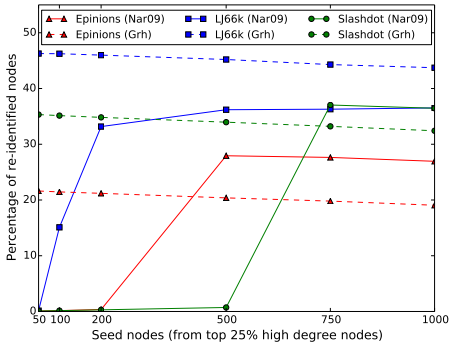


Figure 2: Measurements for different values of the Θ parameter.

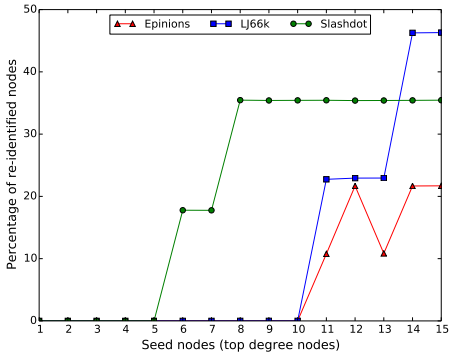
significant role in the algorithm, and matches produced by Grasshopper can be accepted unconditionally as the error rate is very small. For example, the LJ66k network showed on the figure had the highest error rates in our experiments (1.16% and decreasing), while in other networks it was well below 0.38%.

In our previous work [14], we showed that the seeding method should be carefully chosen for the given network. Here, we have measured the differences in the transition phase of Grasshopper (i.e., minimum number of seeding to have stable large-scale propagation) compared to Nar09; our results are summarized on Fig. 3a. It turned out that Grasshopper needs only a fraction of seeds compared to Nar09. In our experiments just a small set of 50 `random.25` seeds proved to be enough to have stable large-scale propagation. Even the lowest value was at 200 for Nar09.

Subsequently, we measured the minimum number of top degree nodes (`top`) that Grasshopper require for initialization (see Fig. 3b). For the Nar09 algorithm, in previous work [14], we measured this to be 60-85, depending



(a) Grasshopper required a fraction of seeds compared to Nar09; in all cases only 50 seeds were enough to have stable large-scale propagation. Furthermore, Grasshopper can achieve significantly larger recall rates in some cases, like in the LJ66k network displayed on the figure.



(b) Grasshopper can be initialized successfully with only a handful of `top` nodes as seeds, e.g., 6 nodes was enough in the Slashdot network (unstable). Even in case of a background knowledge having a lower overlap with sanitized data, such a number of top seeds are likely to be identifiable. For comparison, we measured this to be 60-85 for the Nar09 algorithm [14](depending on the network) when having exactly the same settings.

Figure 3: Seeding properties characterized and compared for Nar09 and Grasshopper.

		Slashdot				Epinions				LJ66k			
		α_e				α_e				α_e			
	$\alpha_v \downarrow$	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0	0.25	0.5	0.75	1.0
Nar09	0.25	0.64	2.48	11.74	19.81	1.07	5.11	10.95	14.80	0.79	6.47	19.61	27.83
	0.5	0.47	19.88	36.60	47.58	0.95	17.15	25.90	32.73	0.72	24.75	35.83	54.55
	0.75	0.40	28.71	50.78	60.43	0.62	25.39	36.12	44.42	0.86	33.97	58.21	78.90
	1.0	0.35	15.53	58.94	68.36	0.45	31.19	43.25	52.60	1.82	37.85	75.28	88.54
Grh	0.25	0.18	14.83	23.36	28.02	0.26	7.80	14.90	17.84	0.11	14.75	25.75	33.28
	0.5	0.31	26.81	35.33	39.34	0.38	15.82	21.05	24.65	0.16	27.98	46.16	57.45
	0.75	6.18	33.26	40.47	44.10	2.83	18.78	24.22	28.46	0.25	35.64	58.40	68.20
	1.0	20.41	36.71	42.91	46.88	10.73	20.70	25.87	30.30	12.43	49.20	65.35	73.48

Table 1: Recall rates measured on the same datasets by both algorithms. We highlighted results when $R(\mu) > 10\%$ and difference was at least with 2%: green marks where Grasshopper was better, and red where Nar09 provided better results. In general, we observed that results depended on the scale of perturbation; Grasshopper provided better results when overlap was smaller and more noisy.

on the network, while current measurements confirmed these numbers to be significantly lower for Grasshopper. For example, there were cases when the Slashdot network could be re-identified by only mapping 6 top degree nodes initially ($R(\mu) = 17.7\%$ means that large-scale propagation could be started in one of the perturbed datasets).

5.2 Recall and Error Rates

We measured recall rates for different settings of perturbation, where α_v and α_e varied as $\alpha_v, \alpha_e \in \{0.25, 0.5, 0.75, 1.0\}$, and our results are summarized in Table 1. We highlighted cases when results differed at least with 2% assuming also $R(\mu) > 10\%$. In all networks results depended on the scale of perturbation. When the overlap, thus similarity was high between G_{src} and G_{tar}, Nar09 provided better results; while in cases with a lower similarity Grasshopper could achieve larger recall rates.

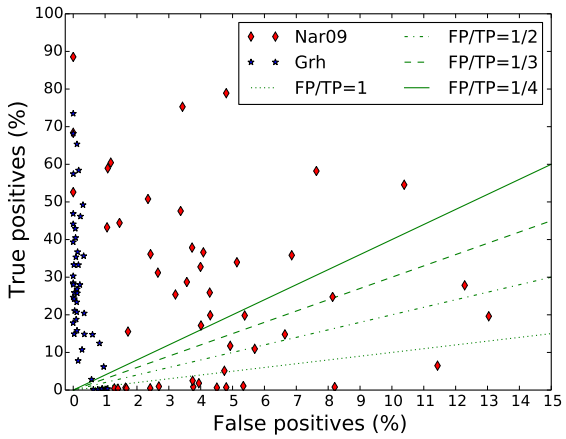


Figure 4: Error rates for the Grasshopper algorithm was significantly lower than in the Nar09 algorithm. Error rates were so low that these can be simply disregarded when executing an attack.

However, we observed slightly worse results in the Epinions dataset. Possibly this is due to structural differences: in the Epinions dataset the majority of the nodes (almost 70%) have a degree of $\deg(v) \leq 3$, while this is 58.8% in the Slashdot and 33.9% in the LJ66k network. This difference needs to be investigated in future work. Making decisions on which algorithm should be used is possible, as according to recent results, an adversary can estimate the overlap between the background and target datasets [10]. However, in general, having a lower overlap between the target dataset and background knowledge is a more realistic scenario, making the results of Grasshopper more relevant. We have additionally provided matching error rates on Fig. 4. Error rates for the Grasshopper algorithm were so low that these can be simply disregarded when executing an attack.

6 Measuring Anonymity

In this section we evaluate two anonymity measures that proved to be the most efficient in the evaluation of anonymity measures for the Nar09 algorithm [15].

6.1 Local Topological Anonymity: Definition and Variants

Large-scale structural re-identification attacks compare nodes against their 2-neighborhoods in their local re-identification (or propagation) phase, therefore, the more similar a node is to its neighborhood, the lower chance it has for being re-identified. It is important to capture this property by anonymity measures. Based on this, we defined LTA [12] as follows:

Definition 1 *A Local Topological Anonymity measure is a function, denoted as LTA*$(\cdot)$*, which represents the hiding ability of a node in a social network graph against attacks considering solely the structural properties of the node limited to its d-neighborhood*[1]*.*

Based on this definition, we can introduce LTA variants that are tailored for some specific attacks. These variants may have different ways of calculation or depend on different measures of similarity. The similarity function depends on the node fingerprinting function that the given attacks rely on. For example, Nar09 (and also the algorithm proposed in [20]) simply

[1]The following variants use $d = 2$, as this is a good trade-off between precision and computational requirements, and it is harmonized with typical network diameters.

compares the sets of neighbors of nodes (of G_{src}) to the neighbors of their friends-of-friends (in G_{tar}) by using cosine similarity[2].

Due to this reason, and that other evaluation also found cosine similarity to have best results in similar applications [26], we proposed multiple LTA variants based on cosine similarity in [12]. Both the evaluation in small [12] and large networks [15] showed that the variant labeled as LTA_A is the most appropriate for measuring anonymity for Nar09.

This variant specifies the average similarity of a node compared to others in its 2-neighborhood (i.e., friends-of-friends), and calculated as follows:

$$\text{LTA}_A(v_i) = \sum_{\forall v_k \in V_i^2} \frac{CosSim(v_i, v_k)}{|V_i^2|}, \tag{6}$$

As LTA measures are expected to indicate level of identification, LTA_A does that: the lower the LTA value is, the higher the chances are that the node will be re-identified.

While evaluating Nar09, we found less than 20% of nodes with $\deg(v) \leq 3$ were correctly re-identified, while this was around 80% for nodes with higher degree (approx. starting from $\deg(v) \geq 30$). This and other signs advised that node degree can be also an appropriate measure of anonymity; evaluation in [15] confirmed this, as node degree is proved to be useful for apriori comparing re-identification rates between nodes. Thus, we could use node degree as an anonymity measure, denoted as:

$$\text{LTA}_{deg}(v_i) = \deg(v_i). \tag{7}$$

[2] $CosSim(v_i, v_j) = \frac{|V_i \cap V_j|}{\sqrt{|V_i| \cdot |V_j|}}$.

In case of $\text{LTA}_{deg}(v_i)$ assessment of node anonymity works differently compared to $\text{LTA}_A(v_i)$: that the higher the node degree is, the higher the chance is that it can be de-anonymized, i.e., conversely than for LTA_A.

6.2 Evaluation

For the correlation measurement here we used the Spearman's rank correlation [2], as it is more important to see if an LTA metric correctly orders nodes in a decreasing or increasing order according to $S(v)$, rather than considering the exact difference between rankings. We measured the discussed variants on the same datasets we used for measuring recall rates in Section 5.2.

The results are shown on Fig. 5, where each dataset is distinguished by color and variants are plotted with different marker. As for our experiments both correlation values closer to 1.0 (ordered by decreasing anonymity) and to -1.0 (vice-versa) are considered to be appropriate, we displayed the absolute value of correlations.

Recall clearly has a significant effect on the correlation; however, the figure shows that both evaluated variants had acceptable correlation rates, and LTA_{deg} had better results except for a few cases. In the evaluation of Nar09 and anonymity measures, it turned out that network structure determines which anonymity measure should be used [15]. In the evaluation of Grasshopper, this could not be confirmed.

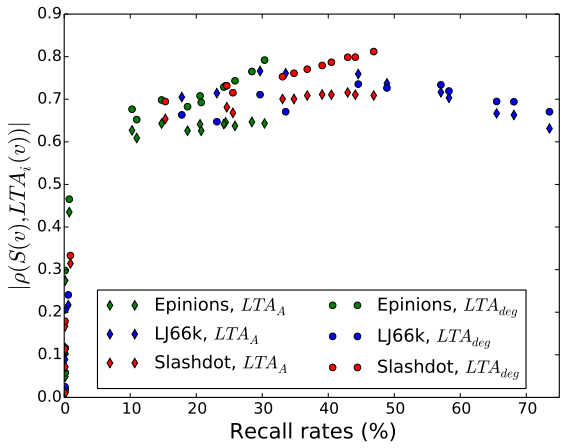
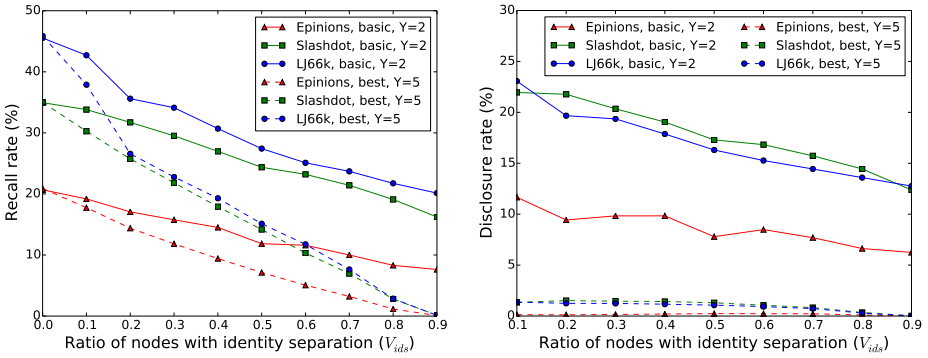


Figure 5: Measuring correlation between output of LTA variants and re-identifications rates of Grasshopper on nodes. Recall clearly has a significant effect on this, however, the figure clearly shows that both evaluated variants had acceptable correlation rates.

7 Characterizing Robustness of Identity Separation for Grasshopper

Recall rates presented in Table 1, shows that when the noise is higher between the background knowledge and the target dataset, Grasshopper provided equal or better results, at least for the Slashdot and LJ66k network. This fact makes the evaluation of identity separation interesting with the Grasshopper algorithm: is this novel algorithm more robust against this privacy-enhancing technique than Nar09?

We analyze this issue from two main aspects, and we aim to provide comparable results with the evaluation provided in [15]; however, the provided results are also interesting when they are regarded on their own.



(a) Recall rates (b) Disclosure rates

Figure 6: The Grasshopper algorithm is quite robust against features of identity separation, in particular for the basic model with $Y = 2$, i.e., the attack could not be defeated even with $|V_{ids}| = 0.9$. In case of the best model with $Y = 5$, Grasshopper can be defeated only with a very large fraction of participants; however, for the adopters of the technique user privacy is preserved in this case.

7.1 Non-Cooperative Identity Separation

The non-cooperative setting includes applying the basic and best models (proposed in Section 3.2) to nodes without considering the relation between them or with other nodes in the network. To experimentally measure how adoption rates influence the success of the attacker, we gradually increased the number of users applying identity separation (denoted with V_{ids}). On each generated dataset we run the algorithm with given settings, and measured recall rates; our results are shown on Fig. 6.

Regarding recall rates, Fig. 6a displays powerful robustness of the Grasshopper algorithm against identity separation. For the basic model when two identities are used, the attack is incredibly robust, the algorithm still capable of re-identifying a large fraction of users even when 90% of the users adopt the technique! Compared with results measured for the Nar09 algo-

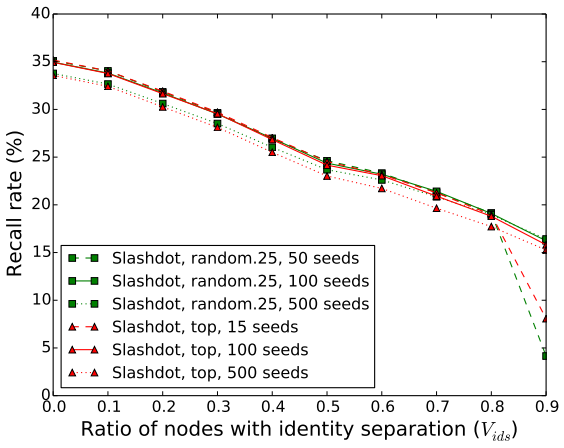


Figure 7: Seeding showed to have only limited effects on overall results of the Grasshopper algorithm.

rithm [15], this is a significant improvement from the attacker point of view. For Nar09, this is around $60 - 70\%$ for the Epinions and Slashdot network, and $80 - 90\%$ in the LJ66k dataset.

Identity separation operates more efficiently when used with the best model $Y = 5$, yet 80% of the users need to participate to repel the de-anonymization attack (in [15] this is around 50% for the Nar09 algorithm). While disclosure rates for the basic model stay also high almost regardless the adoption rate, results of the best model seem to be more promising: disclosure rates stay around 1% and below. We can conclude, that while non-cooperative identity separation was ineffective in tackling the Nar09 attack, it performs even worse against Grasshopper, but not for users aiming to protect themselves with the best model, $Y = 5$. In the latter cases user can effectively minimize possible information leakage.

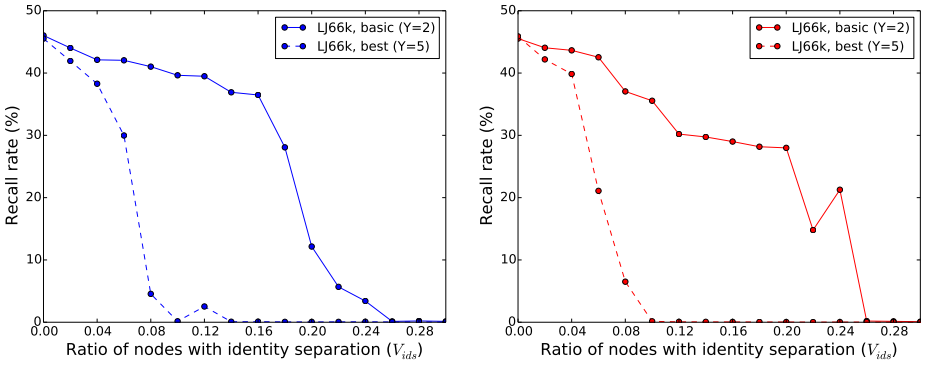
We have also analyzed the effect of the seeding method on the overall recall rate of the propagation phase of Grasshopper, as this parameter is

under the control of the attacker, and could be used to enhance results. Our measurements revealed (see example on Fig. 7) that seeding can not be used to significantly increase recall rates of the algorithm, but the number of seeds determine stability of the algorithm when the perturbation introduced by identity separation is high (e.g., $|V_{ids}| \geq 0.9$).

7.2 Global Cooperation for Identity Separation

There are several ways of organizing cooperative identity separation. In our evaluation, we choose to select nodes for cooperation that are globally important, where node importance can be measured from an adversarial point of view: nodes having a higher chance of re-identification should be selected first. Therefore, in our work we measure importance by adopting anonymity measures proposed in Section 6; namely LTA_A and LTA_{deg}. This additionally allows comparison of our results to [15], where analysis is provided with LTA_A based global cooperation on tackling Nar09.

In two separate series of experiments we ranked nodes accordingly to their LTA scores and created perturbed datasets where first nodes having the lowest anonymity rankings were selected for for identity separation. Our results are shown for each set of experiments for the LJ66k dataset on Fig. 8. Compared to the results of non-cooperative case, we can observe some progress, as even in the worst case (LTA_{deg}, basic model, $Y = 2$) at most $|V_{ids}| = 0.26$ proved to be enough for stopping the attack. Differences between results of LTA_A and LTA_{deg} are not outstanding, thus we can conclude that each method is feasible for tackling the Grasshopper attack by using this strategy



(a) Global cooperation based on LTA_A. (b) Global cooperation based on LTA_{deg}.

Figure 8: Global cooperation can significantly decrease the minimum number of adopting users required for tackling Grasshopper.

(results of LTA_A is only subtly better).

Compared to the measurements provided in [15] for Nar09, Grasshopper proved to be more robust in the case of the basic model. Regarding the LJ66k dataset, for tackling Nar09 with nodes using the basic model, $Y = 2$ the minimum number of participants was $|V_{ids}| \sim 0.15$, and $|V_{ids}| \sim 0.10$ for the best model, $Y = 5$. This also means that maintaining network privacy with targeting nodes having a low anonymity level can be still a working strategy, if cooperation can be organized.

8 Conclusion

In this paper, we provided the scheme of a novel structural re-identification attack called Grasshopper, and we experimentally compared it to the state-of-the-art attack, called Nar09. We have shown that in a number of cases when the attacker knowledge is rather noisy, Grasshopper can achieve significantly higher yield levels that was not possible with Nar09. It also turned out that

while Grasshopper also has parameter Θ for controlling the trade-off between yield and accuracy, our algorithm produces negligible error rates, typically around 1% and below. This is only a fraction that Nar09 produced under the same circumstances, meaning that matches proposed by Grasshopper are so low of error, all mappings could be accepted by an attacker without further filtering. Finally, we have shown that our algorithm can be initialized with the fraction of seed nodes compared to Nar09 to reach maximal re-identification. In all test networks 50 nodes selected from the top 25% by degree, or only 15 of top nodes proved to be sufficient, while these numbers were respectively around 750 and 85 for Nar09 (these measurements are provided in [14]). All these results prove that Grasshopper is a more suitable alternative when the goal is to have a low error in results, but also when the overlap between the sanitized and auxiliary datasets are low.

Achieving higher recall rates for noisy background knowledge is a sign of robustness. We also tested this by measuring how Grasshopper can defeat identity separation. It turned out that our algorithm is quite resistant to features of identity separation. When we simulated users creating two new identities (basic model), the algorithm was still capable of re-identifying a large fraction of users even when 90% of the users adopted the technique! This is quite high, compared to the Nar09 algorithm, where we measured this around $60 - 70\%$ for the Epinions and Slashdot networks, and $80 - 90\%$ in the LJ66k dataset in our previous work [15]. Even for a model with a higher number of identities and with edge deletion (best model, $Y = 5$) the required proportion of participant was around $80 - 90\%$ to defeat Grasshopper. Fortunately, in this case, the attacker could learn only a little

about nodes adopting the technique, highlighting the applicability of this identity separation strategy for protecting user privacy.

Finally, we have evaluated two anonymity measures that was originally proposed and evaluated for the Nar09 attack [15], and showed that these are also useful for Grasshopper, as a high correlation was present in our experiments between the estimated level of anonymity and re-identification rates. Based on these measures, we have tested two global cooperation schemes, where nodes having a lower anonymity level were selected for adopting identity separation. While Grasshopper turned out to be more robust than Nar09 in this case (compared to results in [15]), it could not tackle users adopting the best model with $Y = 5$ with significantly lower participation rates. We can conclude that using the best model ($Y = 5$) is a feasible privacy enhancing strategy that minimize information disclosure also against the Grasshopper algorithm, and if network level cooperation can be organized, for protecting network privacy, too.

We have also pointed out several interesting research issues for future work. First of all, it would be important to refine the convergence criteria for stopping the algorithm. At the time of writing this paper, using a memory of past mapping seems to be the best candidate to count truly new mappings for convergence. Furthermore, it should be investigated why we observed differences in recall between Epinions and others networks. This could also help us in improving Grasshopper to achieve even higher recall rates when the overlap is higher.

References

[1] Pearson correlation. http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient. Accessed: 2014-04-22.

[2] Spearman's rank correlation. http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient. Accessed: 2014-04-22.

[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] F. Beato, M. Conti, and B. Preneel. Friend in the middle (fim): Tackling de-anonymization in social networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 279–284, 2013.

[6] F. Beato, M. Kohlweiss, and K. Wouters. Scramble! your social network data. In S. Fischer-Hbner and N. Hopper, editors, *Privacy Enhancing Technologies*, volume 6794 of *Lecture Notes in Computer Science*, pages 211–225. Springer Berlin Heidelberg, 2011.

[7] S. Clauß, D. Kesdogan, and T. Kölsch. Privacy enhancing identity management: protection against re-identification and profiling. In *Proceed-*

ings of the 2005 workshop on Digital identity management, DIM '05, pages 84–93, New York, NY, USA, 2005. ACM.

[8] L. A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101, 2009.

[9] G. Danezis and C. Troncoso. You cannot hide for long: De-anonymization of real-world dynamic behaviour. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 49–60, New York, NY, USA, 2013. ACM.

[10] P. Govindan, S. Soundarajan, and T. Eliassi-Rad. Finding the most appropriate auxiliary data for social graph deanonymization. 2014.

[11] G. G. Gulyás and S. Imre. Analysis of identity separation against a passive clique-based de-anonymization attack. *Infocommunications Journal*, 4(3):11–20, December 2011.

[12] 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.

[13] 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*. Springer Berlin Heidelberg, 2013.

[14] 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*, 2014.

[15] G. G. Gulyás and S. Imre. Using identity separation against de-anonymization of social networks. Submitted to the Journal of Transactions on Data Privacy. Available at: `http://gulyas.info/upload/Gulyas_TDP2014.pdf`, May 2014.

[16] G. G. Gulyás, R. Schulcz, and S. Imre. Modeling role-based privacy in social networking services. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, pages 173–178, June 2009.

[17] M. Hansen, P. Berlich, J. Camenisch, S. Clauß, A. Pfitzmann, and M. Waidner. Privacy-enhancing identity management. *Information Security Technical Report*, 9(1):35–44, 2004.

[18] 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.

[19] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 185–196, New York, NY, USA, 2010. ACM.

[20] 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.

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

[22] 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.

[23] W. Peng, F. Li, X. Zou, and J. Wu. Seed and grow: An attack against anonymized social networks. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 587–595, 2012.

[24] 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.

[25] K. Sharad and G. Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '14, New York, NY, USA, 2014. ACM.

[26] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 678–684. ACM, 2005.

[27] 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.

[28] B. van den Berg and R. Leenes. Audience segregation in social network sites. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 1111–1116. IEEE, 2010.

[29] B. van den Berg and R. Leenes. Keeping up appearances: Audience segregation in social network sites. In S. Gutwirth, Y. Poullet, P. De Hert, and R. Leenes, editors, *Computers, Privacy and Data Protection: an Element of Choice*, pages 211–231. Springer Netherlands, 2011.