A Probabilistic Inference Attack On Suppressed Social Networks

by

BARIŞ ALTOP

Submitted to the Graduate School of Sabancı University
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

Spring, 2011

A Probabilistic Inference Attack On Suppressed Social Networks

Approved by:

Assist. Prof. Dr. Mehmet Ercan Nergiz .....
(Thesis Supervisor)

Assoc. Prof. Dr. Yücel Saygın .....
(Thesis Supervisor)

Assoc. Prof. Dr. Berrin Yanıkoğlu .....

Assoc. Prof. Dr. Erkay Savaş .....

Assoc. Prof. Dr. Cem Güneri .....

Date of Approval: 08.08.2011

A PROBABILISTIC INFERENCE ATTACK ON SUPPRESSED SOCIAL NETWORKS

Barış ALTOP

Computer Science and Engineering, Master's Thesis, 2011

Thesis Supervisors: Assist. Prof. Dr. Mehmet Ercan Nergiz,
Assoc. Prof. Dr. Yücel Saygın

Keywords: Social Networks, Inference Attack, Privacy, Private Data Protection, Classification

Abstract

Social Networks (SNs) are now widely used by modern time internet users to share any personal information. Such networks are so rich in information content that there is public and commercial benefit in sharing them with other third parties. However, information stored in SNs are mostly person specific and subject to privacy concerns. One way to address the privacy issues is to give the control of the data to the users enabling them to suppress data that they choose not to share with third parties.

Unfortunately, above mentioned preference-based suppression techniques are not sufficient to protect privacy mainly because they do not allow users to control data about other users they are linked with.

Information about neighbors becomes an inference channel in an SN when there is known correlation between the existence of a link between two users and the users having the same sensitive information. In this thesis, we propose a probabilistic inference attack on a suppressed social network data, that can successfully predict a suppressed label by looking at neighboring users' data. The attack algorithm is designed for a realistic adversary that knows, from background or external sources, the correlations between labels and links in the SN. We experimentally show that it is possible to recover majority of the suppressed labels of users even in a highly suppressed SN.

BASTIRILMIŞ SOSYAL AĞLARDA OLASILIKSAL BİR ÇIKARIM SALDIRISI

Barış ALTOP

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2011

Tez Danışmanları: Yard. Doç. Dr. Mehmet Ercan Nergiz, Doç. Dr. Yücel Saygın

Anahtar Kelimeler: Sosyal Ağlar, Çıkarım Saldırısı, Mahremiyet, Kişiye Özel Veri Güvenliği, Sınıflama

Özet

Sosyal Ağlar günümüz internet kullanıcıları tarafından kişisel bilgilerin paylaşımı amacıyla yaygın olarak kullanılmaktadır. Bu tür ağların, bilgi içerikleri çok zengin olduğundan, diğer üçüncü partiler ile paylaşımı kamusal ve ticari fayda getirmektedir. Ancak, sosyal ağlarda saklanan bilgiler çoğunlukla kişiye özeldir ve gizlilik endişelerine tabidir. Gizlilik sorunlarını gidermenin bir yolu, kullanıcılara kendi verilerinin kontrolü vermek ve istedikleri verileri bastırarak üçüncü kişilerden gizlemelerini sağlamaktır.

Ne yazık ki yukarıda bahsedilen tercihe dayalı bastırma teknikleri gizliliği sağlamaya yetmemektedir. Bunun temel sebebi, bu tür koruma sistemlerinin kullanıcılarına, bağlantılı oldukları diğer kullanıcıların paylaştıkları veriler üzerinde kontrol izni vermemeleridir. Aralarında bağlantı bulunan kullanıcılar arasında veri benzerliği açısından ilişki mevcuttur; bu ilişki de iki komşu kullanıcı arasında veri çıkarsama kanalı oluşturur. Bu tezde bastırılmış sosyal ağlarda komşu kullanıcıların verilerine bakarak kişilerin bastırılmış bilgilerini bulabilen olasılıksal bir çıkarsama saldırısı öneriyoruz. Bu saldırı algoritması sosyal ağdaki etiketler arası bağıntıyı ve bağlantıları bilen gerçekçi bir düşmana göre tasarlanmıştır. Yüksek derecede bastırılmış sosyal ağlarda bile kullanıcıların bastırılmış etiketlerinin çoğunluğunu çıkarsamanın mümkün olduğunu deneysel olarak göstermekteyiz.

to my beloved family

Acknowledgements

I wish to express my sincere gratitude to Assoc. Prof. Yücel Saygın and Assist. Prof. Dr. Mehmet Ercan Nergiz, for their continuous support and wortwhile guidance thoughout my masters studies. Assoc. Prof. Yücel Saygın's support starting from my application process till today, was always a great motivation during my studies. Also I am thankful to Assist. Prof. Dr. Mehmet Ercan Nergiz for believing in the topic I was interested in. In addition, I am thankful to my thesis defense committee members: Assoc. Prof. Berrin Yanıkoğlu, Assoc. Prof. Erkay Savaş and Assoc. Prof. Cem Güneri for their support and presence.

I appreciate Duygu Karaoğlan for her help during the implementation process. I would also like to thank to my friends Emre Kaplan, İsmail Fatih Yıldırım, Burcu Özçelik, Yarkın Doröz and Erman Pattuk for their help in the cirruculum courses. Duygu Karaoğlan deserves special thanks for her precious and continuous support.

Last, but not the least, I am immensely thankful to my family, for being there when I needed them to be, for believing in me and supporting me throughout all my decisions.

Contents

List of Figures

List of Tables

1 Introduction

Social Networks (SNs) [19] are among the most popular communication and sharing platform on the Internet in the modern world. SNs are vast in size and can carry personal and sensitive information of an individual such as political views, religion, sexual orientation, etc. This raises every privacy concern when SN data is published for research purposes or released to third parties for business purposes. Without a direct transfer of SN data, even a simple internet user can easily get access to lots of profiles and information by just searching for publicly available SN data, i.e. by finding people with open profiles using web crawlers, elaborated in [38, 14].

Given such a threat, most service providers offer various privacy policies for their registered users most of which allow users to choose what information to share and whom to share with. For example, a user can specify her age to be publicly available while suppressing the political group he/she is a member of. However, to what extend such policies address privacy concerns remains to be an open question. The main problem with such preference-based protection mechanisms is that the users cannot decide what other people, that they are connected with, are sharing. Additionally, the user may share some information without the exact knowledge of its consequences. Or, just connected people may share information about the user also without considering the aftermath [23, 8, 17].

As we know SNs are not just a way to keep records, like hospital databases or voter lists. In SNs people mimic their daily social life onto the internet

public. As in real life, people do make mistakes and can cause an information breach for someone else. Such as publicly asking someone about his/her private disease. There is also another way to breach privacy in SNs, which is caused by emotions. People act, just like in real life, on some emotions like anger, sadness, grudge, etc. Hence they are more willing to share private information about other purely based on the emotions they have against them. For example if two best friends start to hate each other, they may post information publicly against each other. But beyond these two factors sometimes the user itself discloses his/her information with the help of his/her neighbours. This is because people tend to build relations with similar backgrounds or facts, like school, age, political views, religious views, sexual orientation, etc. A person may hide his/her information, but the network he/she creates around him/her-self is a way to define him/her.

Such information disclosed by 'neighbours' serves as an inference channel for any suppressed data if the adversary knows that some correlation exists between the existence of a link among two users and the users having the same sensitive information. For example, even though the user chooses to suppress his/her membership to a political group, the adversary can look for memberships disclosed by his/her friends. If a sufficient number of her friends specify their membership to the same political group, an adversary, assuming such groups tend to form cliques in the social networks, can predict her membership with high probability. Besides these information retrieving techniques, an adversary can also be a moderator or owner of such groups in a SN, giving him/her the ability to collect more accurate data and to extend his/her prediction radius among the SN.

In this thesis, we propose a probabilistic inference attack, which predicts the suppressed sensitive information from a highly suppressed SN with high success rate given the network structure and the degree of correlation between links and labels. The attack algorithm returns, for each node, a probability that the node has a specific label (e.g., being a member of a group). The sketch of the algorithm is as follows: For each node and label (e.g., sensitive information) in the SN, the attack algorithm assigns a probability function for the likelihood of the node to have the label. As the correlations are known, the probability function for a node is defined in terms of probabilities of neighbouring nodes (e.g., probabilities that they have the label). This creates a system of equations to solve for the probabilities. In order to solve the large system of equations, we propose an iterative algorithm. Basically, we start with an initial state for all probabilities and iteratively update probabilities based on the probability function. The algorithm returns the probabilities when the system converges to a final state. We experimentally show that the attack algorithm predicts the suppressed labels with high success rates even in a highly suppressed social network.

The rest of the thesis is organized as follows: Section 2 gives background on social networks, followed by related work on data publishing. In Section 3, we present the motivation of the thesis and its contributions. Then we describe our algorithm in detail in Section 4. Section 5 evaluates the proposed attack algorithm based on test cases. Finally, we conclude in Section 6.

2 Background Information and Problem Definition

In this section, we formally define a social network in our domain, state what the adversary knows, and formally present the problem definition.

2.1 Structure of Social Networks (SNs)

SNs can be observed as graphs [6], which consists of vertices or nodes and edges. On a SN each user (i.e. profile owner) is a node on the graph and any relationship between two users is an edge between them. Depending on the SN these edges can vary in weight and directivity, e.g. the "friendship" relationship between two users on a SN is an undirected edge, in contrast to a "following" / "follower" [15] relationship being a directed edge. If the SN has different types of relationship among two users, then each type of relationship can be represented with a different weight on each edge.

Social Network: In our domain a social network is an undirected graph $SN = (V, E)$ where each node $v \epsilon V$ is a user and $e \epsilon E$ is an edge, defined as $e = (v_i, v_j)$ with $v_i, v_j \in V$. There is an edge between v_i and v_j if and only if there exist an $e \in E$ such that $e = (v_i, v_j)$. A node can have multiple edges to different nodes, but there can't be a node without having any edges. For the network we have set of labels L representing a sensitive information. For each user $v \in V$, and label $\ell \in L$ either the user has the label which we denote as $v.\ell = 1$, or does not have the label which we denote as $v.\ell = 0$.

For example the set of labels could be:

$$L = \{age > 30,\, location = Europe,\, political\, view = right\} \tag{2.1}$$

Each one of the labels will be referred to as ℓ_i and $i = [1, 3]$. Hence the notation like $v.\ell_2 = 0$ would mean that the user v is not in Europe, $v.\ell_3 = 1$ would mean that v has a right-wing political view.

Suppressed Social Network: We say a suppressed social network $SN^* = (V', E')$ is derived from a social network $SN = (V, E)$ iff the following conditions are met: 1. There is a one to one correspondence between $v \in V, e \in E$ and $v' \in V', e' \in E'$ 2. For all matched v, v', and $\ell \in L$; if $v.\ell = 1$, either $v'.\ell = 1$ or $v'.\ell = *$ (representing unknown) . Else if $v.\ell = 0$, then $v'.\ell = *$.

So a suppressed SN^* has the same network structure as its corresponding SN. The only diffirence is some of the labels in SN^* is set to * representing unknown. An example of an SN subgraph and its suppressed version can be seen in Figure 1 and 2.

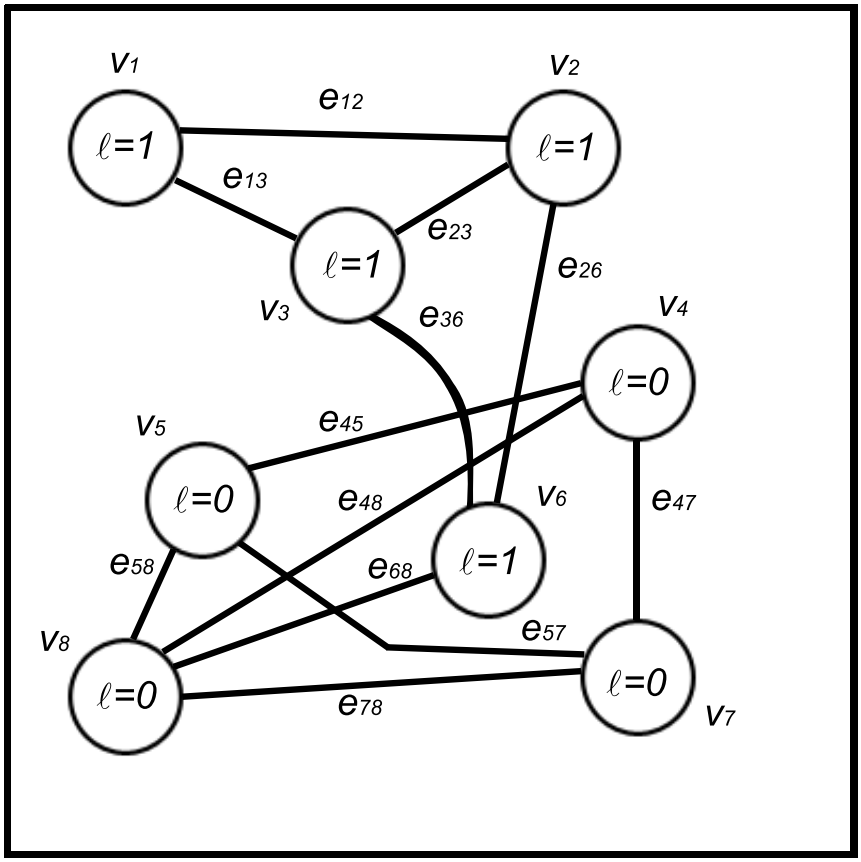

Figure 1: Example of a Social Network

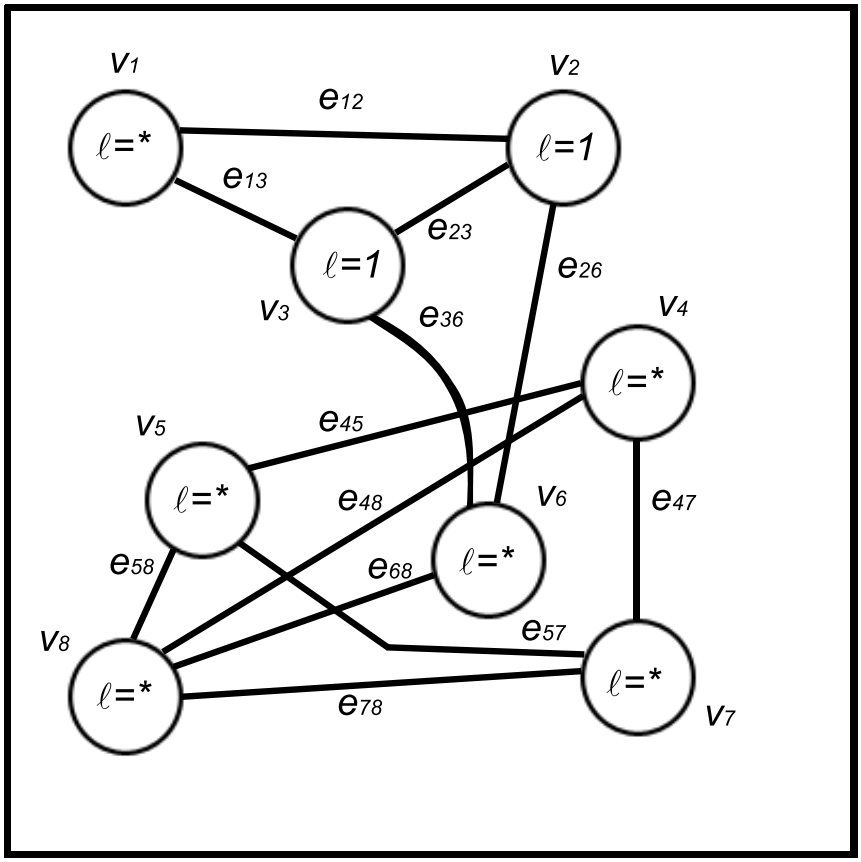

Figure 2: Suppressed version of SN from Figure 1

Neighbour Set: The β-neighbour set N_v^β of a node v w.r.t. a label ℓ in a social network $SN = (V, E)$, is defined as the set of nodes that are connected to v and have label β: $N_v^\beta = \{v' | \exists e = (v, v') \in E, v'.\ell = \beta\}$. N_v returns all neighbours of node v. (E.g., $N_v = N_v^0 \cup N_v^1 \cup N_v^*$)

In Figure 2, $*$-neighbour set of v_3, $N_{v_3}^* = \{v_1, v_6\}$. Similarly, $N_{v_3}^1 = \{v_2\}$.

2.2 Problem Definition

In our domain, the data holder has a social network SN, however only a suppressed version SN^* of SN is released due to preference based privacy policy. To ease discussion, we assume, without loss of generality, the network has only one label ℓ. We assume the adversary has access to the following information:

1. K_1: The released suppressed network SN^*.
2. K_2: For a node v in the unsuppressed SN, $P(v.\ell = 1 \mid \frac{|N_v^1|}{|N_v|} = r)$ for all r.

Note that the above adversary realistic. The knowledge in item 2 can be obtained approximately by an adversary which is a user in the social network that can see a subgraph of the network. Or it can be obtained from other public networks or derived from domain knowledge. In this thesis, we propose an attack algorithm for such an adversary that will compute the following probability:

$$P(v.\ell = 1 \mid K_1, K_2)$$

2.3 Related Work

Privacy breach in published data sets was first shown in [39], where the authors were able to obtain sensitive information of people from datasets without unique identifier such as names, SSNs, $\cdots$. Since then, many different

privacy models and anonymization techniques [34, 36] have been proposed to prevent attacks by different adversaries.

The first set of solutions for privacy preserving data publishing focused primarily on tabular data in which each individual has a single record. We now summarize the earlier research on tabular data publish but it should be noted that since the SN data inherits a network structure and the location of the individuals inside the structure gives away sensitive information, the techniques proposed for tabular data cannot be used to de-identify SN data.

2.3.1 Tabular Data Publishing

Tabular data [12] is a way of organizing data in rows and columns, where rows represent the records and columns represent the attributes of each record. In contrast to graphs, individual records are not linked to each other. Every record consists of many attributes and depending on the dataset there may be a number of sensitive attributes[1] [7] for each record. Table 1 and 2 is an example for tabular data and its publishing methods, where all attributes are considered as sensitive information. These attributes are considered sensitive due to their nature for linking them with information on different tables, hence making them *quasi-identifiers* [40].

[1]is a personal information or opinion, that can be used to classify people into groups after re-identification, e.g. diseases, memberships, etc.

Table 1: A Fictitious Tabular Data

Name	Age	Sex	Zip
John Doe	25	M	34141
Jane Doe	22	F	34140
Mark Johnson	34	M	34138
John Smith	19	M	34139
Sue Anne	43	F	34141

Table 2: Suppressed version of Data from Table 1

Name	Age	Sex	Zip
*	[25, 34]	*	3414*
*	[15, 24]	*	3414*
*	[25, 34]	*	3413*
*	[15, 24]	*	3413*
*	[35, 44]	*	3414*

Anonymization techniques like k-anonymity [40, 39], ℓ-diversity and δ-presence try to anonymize the tabular data before releasing them publicly, but also try to keep a level of information available in the suppressed versions for research. Meaning that, if the data is over-anonymized then the released data will not contain any information on the table itself, opposed to under-anonymizing which will lead to total re-identification of the data (Table 3, 4).

Table 3: Over-anonymized Data

Name	Age	Sex	Zip
*	< 50	*	341**
*	< 50	*	341**
*	< 50	*	341**
*	< 50	*	341**
*	< 50	*	341**

Table 4: Under-anonymized Data

Name	Age	Sex	Zip
*	≤ 25	M	34141
*	≤ 25	F	34140
*	≤ 35	M	34138
*	≤ 20	M	34139
*	≤ 45	F	34141

The anonymization techniques must be improved or revised according to the new adversary knowledge. Adversaries gather information from all sorts of sources and combine them into one big table for future use. Their main goal in tabular formed published data is to link the suppressed records with the data they have in hand.

Anonymization

k-Anonymity [40] is the first technique offered to anonymize datasets to make them resistant against re-identification [2]. The re-identification process in tabular data publishing is accomplished through combining two different datasets with similar attributes along with different anonymized

attributes. Tabular data like U.S. voters' lists were one of these records and were prefered in view of the fact that anyone could buy it from the government agencies. It contained sensitive information such as age, sex, zip code, etc., and is used to match information from different records to re-identify the suppressed data.

When k-Anonymity is applied to these datasets, it ensures that any combination of the quasi-identifiers would be matched to k indistinguishable records. In other words, when a specific value is queried on the dataset, the result set will contain k identical records for any attribute or queried attribute set. This is achieved through domain generalization hierarchy (DGH) [40] on each sensitive attribute list, where levels of generalization are viewed as a tree. The lesser the height, i.e. towards the root, the more general values are reached, as seen in Figure 3.

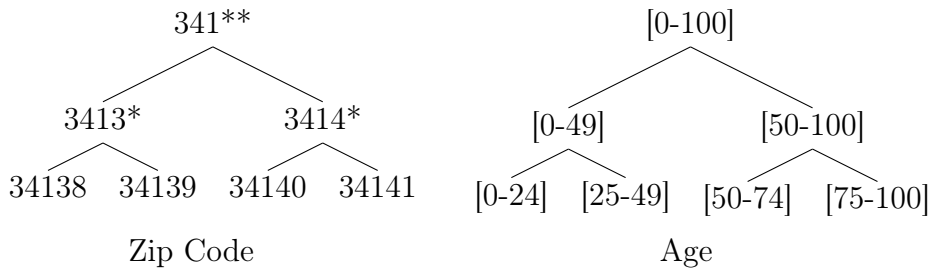

Figure 3: Sample Domain Generalization Hierarchy

Using DGHs, the k-Anonymity algorithm will produce a suppressed dataset, where any record would have $(k-1)$ identical records regardless of any attribute combination queried on.

Despite this new technique Machanavajjhala, et. al. [34] has proven that sensitive information is not secure to re-identification attacks and stresses the adversary knowledge as the cause for this and proposes a new algorithm, ℓ-diversity. In k-anonymity, any query will return k indistinguishable records. If these k records share the same value for a quasi-identifier it would mean that this information is leaked. In other words if an adversary knows that the person he/she is searching for is in the returned set of k records, then the adversary can conclude that for that quasi-identifier the attribute value is definite. To cover this defect, the authors propose the algorithm of ℓ-diversity, where in each set of k records each quasi-identifier has at least ℓ values for the sensitive attribute. This means that any attribute within the k records is $\frac{1}{\ell}$ diverse (Table 5, 6).

Table 5: A dataset without personal identifiers

Age	Sex	Zip	Disease
16	M	34106	Cancer
25	F	34107	Flu
20	M	34107	Cancer
30	F	34106	Cold

Table 6: 2-anonymous and 2-diverse version of Table 5

Age	Sex	Zip	Disease
[16-25]	*	3410*	Cancer
[16-25]	*	3410*	Flu
[20-30]	*	3410*	Cancer
[20-30]	*	3410*	Cold

Yet, this new level of anonymity was also not sufficient, caused by continuesly increasing adversary knowledge. As a result, the third major anonymization method is offered, again based on k-anonymity: δ-presence [36]. On the contrary of ℓ-diversity, δ-presence proves that for some sensitive information it may not be possible to achieve ℓ-diversity. If the sensitive data has only two unique attributes $v_i.sen = \{0, 1\}$, i.e. is either true or false for each record, then it is impossible to reach ℓ-diversity in k anonymous dataset. Let us consider that there are n records in which m of them satisfy $v_i.sen = 1$, hence $(n - m)$ will be satisfying $v_i.sen = 0$, and $m \geq (n - m)$. According to these values, the average value for ℓ would be calculated as $\ell = \frac{n-m}{n}$, therefore making the data $\frac{n-m}{n}$-diverse. In order to overcome this deficiency and to make the dataset resistant to publicly available sets of information, the δ-presence algorithm ensures that any record from the publicly available set will have the probability to be linked to the original data between $(\delta_{min}, \delta_{max})$. This algorithm relies on the adversary knowledge, however adversary knowledge may increase in time and must be updated regularly for the algorithm to produce the same level of anonymity every time.

As explained in this section the early phases of privacy protection was based on tabular data publishing. The main threat for privacy was the different tabular data published with different anonymizations, causing adversaries to link the two corresponding tables in order re-identify the original data.

2.3.2 Complex Data Publishing

Differentiating from tabular data, complex data is a set of records, where multiple records combined create a new record set. Complex datasets can be represented in multiple tabular forms or in different forms, such as graphs. The reason for the complexity comes from what information is stored within the dataset and the relation among records, known as relational databases [35]. Let us assume we have a table similar to Table 1 and we would also like to store the spouse relationship. Hence using the data from the main table we create a second tabular data, e.g. *MarriedTo* (Table 7).

Table 7: Tabular representation of spouse relationship

Spouse1	Spouse2
John Doe	Jane Doe
Mark Johnson	Tara Johnson
Brad Smith	Sarah Smith

Compared to a single tabular data, complex datasets contain more information on a single record, through the multiple relation sets between tabular datas. The information and its meaning is being researched instensively under the topic of data mining [25, 21]. Using data mining techniques the information within the tables are interpreted into meaningful conclusions. For example in a supermarket each transaction could point out different information using correlating information, such as *"People who buy diapers also buy milk"*. Such and more examples could be found even in daily use within most data.

Complex data is also being studied for privacy because most datasets can be used by adversaries for background knowledge for data mining applications. One way to anonymize the data is using the k-anonymity technique on multiple and related tables [37]. However datasets may differ and only k-anonymity would not be enough to securely publish the data. Especially datasets like spatio-temporal data [20] are the most important ones. They store coordinates and timestamps for a person, which can be collected through GPS-enabled devices, such as GPS navigator, smartphone, GSM carrier, digital cameras, etc. These informations can be expressed in a multi-relational table, where one table would hold the information on the individual (Table 1) and the other would hold the paths as coordinates (Table 8). In Table 8 the paths are stored as a comma-seperated format and each element of it, is a coordinate in (x_i, y_i, t_i) format, where x_i is the horizontal- and y_i is the vertical displacement and t_i is the timestamp at which the person was located at those coordinates.

Table 8: Path coordinates in spatio-temporal data

ID	Path
1	$\{(x_1, y_1, t_1), (x_2, y_2, t_2), (x_{3,}, y_3, t_3)\}$
2	$\{(x_{12}, y_{12}, t_{12}), (x_{13}, y_{13}, t_{13}), (x_{14,}, y_{14}, t_{14}), (x_{15}, y_{15}, t_{15})\}$

As mentioned above there are adversaries for this information, too. It is proven that even when anonymized, paths of individuals can be retrieved [30, 32, 31]. This information can be used against the person to leak private information, such as *"Person X is going to the hospital every week"* will translate into *"Possible Chronic Disease Carrier"* by an insurance company.

The problem of anonymizing the data comes from its information. Each coordinate must be handled individually in order to suppress key information, such that an adversary can not recreate the original path correctly. So the path becomes a function on the x, y-coordinate system and there may be many ways to recreate the path. One way to visualize the paths could also be graphing them onto a map, but graphs are mostly used to display networks rather than directional single-line paths.

2.3.3 SN Data Publishing

Once SNs became popular among internet users, the number of accounts increased and SNs were holding more data than any other datasets. Hence adversaries created crawlers [38] to harvest the data on SNs, but they weren't just storing them in their previously created tabular data, they were also storing it in a graph form in order to analyze the SN. The main reason for that is, that each user acount is connected with some other accounts, which makes inference between these connected users possible. This can be explained by the graph structure of the SNs, where the SN is not just a collection of profiles row-by-row, but is an interconnected network where people from same interest groups, same schools, same locations, etc. relate to each other by friendship links. Also when opening an account for the first time in a SN, the privacy settings are always set as public by default. During the time the user figures out, how to set his/her privacy preferences most of his/her sensitive data, e.g. age, sex, location, education, religious and political views, etc. can be retrieved by adversaries.

Adversaries change by type or the information they are seeking for, though the information they retrieve gets summoned and distributed through the internet. Inexperienced adversaries search the SNs by jumping through links while more advanced ones use crawlers to harvest the data. Crawlers are web scripts which do the same job as the inexperienced adversaries in an automated fashion.

In the recent years many publishing methods for SN data has been discussed [26, 41, 42]. The main goal of these anonymization-based publishing techniques were also creating a version of the original graph that would mimic the relationships and label distribution of its source. Let SN be the original network and SN^* be a anonymized version of it, i.e. $SN^* \subset SN$. The desired SN^* would then be such that the probability of identifying any node on SN given SN^* is smaller than a threshold value ε (Equation 2.2) . In the mean time, the anonymized network SN^* must also return the same probabilistic results as the original network SN for each query, again with a very small noise factor Δ (Equation 2.4). N' will only return results for countable queries, e.g. "Probability of any node v_i having label $\ell_j = 1$" (Equation 2.3), due to the individual privacy factors it has to meet.

$$P(v_i \mid N) < \varepsilon,\ v_i \in N' \tag{2.2}$$

$$q \equiv [v_i.\ell_j = \{1\}] \tag{2.3}$$

$$Q(q,\, N) = Q(q,\, N') \pm \Delta \tag{2.4}$$

When publishing graph data, the key of anonymization does not rely on suppressing labels of vertices, instead it considers the position of the vertices, which is expressed through its *neighbours* [42]. Neighbours are the vertices that are linked to a node v_i through the edges. For instance in Figure 1, the node v_3 has the neighbours $\{v_1,\, v_2,\, v_6\}$ over the edges $\{e_{13},\, e_{23},\, e_{36}\}$. Backstrom, et. al. [16] explains passive and active attacks using the neighbouring property of the SNs. Adversaries may actively use the SN and create a small group of network, which they would match to the graph that is going to be publicly published. If they are able to find their group within the anonymized version of the graph they would be able to extend from that point to label anonymized nodes.

Hay, et. al. [26] uses perturbation on links in order to obscure the neighbouring relationships, causing the graph to be more securely anonymized. They have concentrated on anonymizing the edges, by removing or adding edges in order to create similar vertices based on an algorithm. Similarly Zhou and Pei [42] create a k-anonymous graph based on the neighbours of vertices, i.e. making k identical nodes for any query. Differentiating from these two researches, Wei, et. al. [41] anonymize both the labels and the neighbours of the nodes. They propose three algorithms to achieve their anonymization. First they create subgraphs, in which each node has the same label. Following this step, they add or remove edges based on subgraph average such that at the end, each node has exactly the same degree

of its neighbours. Finally, they conclude by anonymizing the connectivity among subgraphs, again based on k-anonymity. Despite these anonymization methods, we concentrate on such nodes that do not contain any label at all. In addition to this fact, anonymized graph data is not what the adversaries are going to deal in our proposed method because we assume, that the adversaries collect their data with crawlers. Using crawler data with our algorithm we will infer on the graph. Hence no third party anonymized published data will be used.

It was He, et. al. [27] that brought the idea of inference into SNs. In their research they have assumed that people in a SN tend to build relationships with others such as classmates, co-workers, fellow townsman, etc. Using these relationships and the Bayesian network [28, 29, 24] representation, they inferred on one label of each node. They assumed that the adversary would be aware of one's content of relationships with others, i.e. the adversary is aware of all the neighbours, and social groups of the neighbours of the given $v_i \in V$. Using this information, they prove that for a given node v_i they infer on the labels ℓ_j by analyzing the correct group of neighbours of it. Thus, for each node of $v_i.\ell_j = *$, they select the neighbours with the correct relevance to this label and come to a decision based on this deductive algorithm. In contrast to He, et. al. [27], our inference algorithm does not concentrate on specific groups to infer on labels of nodes, this is due to the fact that we assume adversary does not have the knowledge of which node belonging to which social group.

Recently Lindamood, et. al. [33] showed that inference attacks are a major threat to SNs. They proposed a classification algorithm and a way to prevent inference attacks. They were able to keep the classification algorithm from classifying by looking at the information available after removal of edges and some label information. Basically, by removing edges between the nodes and suppressing labels on the nodes they made the classification algorithm to come to a position where it can't make an decision.

The authors [33] collect the data through a SN crawler, which is described in [38, 14]. They perform three different tests in order to reach the most attack tollerant version of suppression. When removing 10 links per node and 10 labels from each node the classifier ends up in an decision to make between 0.52 and 0.48, which in this case is impossible to infer about the decision of any node. We differ from this work right from the beginning because they are suppressing data which is collected through crawlers. What we want to show is that after adversary collects the data using a crawler it is possible to infer on the remaining unidentified nodes and their labels.

3 Motivation and Contribution of the Thesis

This section includes information on why we selected this subject and what contributions we made.

3.1 Motivation

As SNs come to be popular, many different ways of collecting data became possible. Despite all the preference-based privacy options, Social Network systems are still weak in protecting personal sensitive data. Although some algorithms have been offered on suppressing networks, they always assumed that these networks consist only of edges and nodes. These algorithms ignore the adversaries that try to recover some key sensitive information about that node.

As described in Section 2.3, the relationships among users can point out some key information about the connected parties. If the relationship between some users is more dense, the information retrieved among them can be more precise, regardless of the number of captured profiles. Even if a user suppresses all of his/her sensitive information to any third party, i.e. anyone besides his/her friends can not see any info on the profile, his/her friends' information may point out about his/her sensitive facts, like age, location, political view, etc.

The inference possibility of using information on neighbouring nodes does not need much of an adversary knowledge; however the publicly available

data is more than adequate to conclude such an inference attack. The idea behind inference depends on the fact that people can not control what others are sharing and how this information can relate to their private data. One may decide to not share a fact about his/her personal life, but it can be shared publicly by other users in many different ways. Some may give a personal fact about him/her self, that can unknowingly or willingly depict other related users. This kind of privacy breaches are called *neighbour sharing*, because an adversary is informed about a sensitive information of an user by a neighbouring node.

This breach in privacy is detected by active adversaries, which not only rely on data, that is gathered by crawlers, but also personally view accounts for such information. The adversary is aware of such situations due to the fact that in social networks people may act on emotional factors and share some key information about their neighbours. However, most of the time such neighbour sharing is not based on emotional factors, where users act out without thinking of the consequences, in contrast it is the basic information that the user shares. In other words, people tend to have friends and relations that are based on common factors, such as school, work, political view, religious view, sexual orientation, etc. Using this kind of related neighbours the adversary can guess on information that is suppressed by some users.

Our primary objective is to use this structure of relationships and prove that it is possible to infer on suppressed information without using any other data sets or related information.

3.2 Contribution of the Thesis

What we did is, we try to show that in SNs not sharing any information is not a definite solution for privacy protection. Among the proposed network suppression algorithms, the main issue is to anonymize the network in such a way that it still makes sense and carries similar information of the original network N. We are going to prove that if the data holds correlated information to its original state, defined by Equation 2.4, then simple adversary knowledge may be enough to recover key information on each person individually.

As mentioned in Section 3.1, people that are neighbours have a high possibility of having common factors or having the same information on one or more cases. When considering big social networks, it would be very difficult to view each account for neighbour sharing. Hence using the similar details among users would be a much more efficient way to conclude our proposal.

In the early phases, we concentrated on special interest groups on SNs, which shows the persons belonging to an idea or ideological group. This method was selected as the primary adversary behaviour in order to classify people, even if their profiles were closed to outside viewing. However, this produced a low level of connectivity among users. The low level of connectivity means, that through the special interest groups we were able to access many peoples key information, but the relations among them and their mutual relations was very low to use in an inference attack. Therefore, we changed our objective to the network itself, rather than the information

providing entity, in our case the special interest groups. When using the network itself the number of mutual neighbours among the vertices increases, which will allow better results within each subgraph. In other words, the more related or connected the networks is the better it can be inferred.

Our adversary knowledge was based on only the measurable fact of tendency of users connecting to users with similar formations. When considering each label seperately adversaries can easily conclude to the ratio of connections, i.e. edges, among users that have or not have the label, as in Equations 3.1 and 3.2.

$$P\left(v_j, \ell = 1 \mid v_i.\ell = 1\right) \tag{3.1}$$

$$P\left(v_j.\ell = 1 \mid v_i.\ell = 0\right) \tag{3.2}$$

Although we started our research in the direction of a binomial distribution [1] attack, it evolved into a multinomial inference attack due to the change in information source.

We developed a probabilistic inference attack that can recover a highly suppressed SN. We assumed that an average adversary is capable of creating a crawler for a SN. Although many accounts would be closed to public viewing, the adversary may use his/her account for the crawler to retrieve better results. Many researches on SNs [33, 27] did also produce a crawler, returning more than 50000 accounts in each case. Hence the assumed adver-

sary knowledge is fair in our case. We probabilistcally re-identify the label values of each node individually by looking at their neighbours that do not have suppressed labels.

Each node, v_i, may or may not have neighbours with suppressed label ℓ_j; however, we can conclude in both of the situations the value of $v_i.\ell_j$. The key point here is that the adversary has the knowledge of label distribution within the SN or sub-SN, meaning that the adversary knows by ratio how many of which label is present in the graph (Equation 3.3). By knowing this value the adversary can attack the suppressed graph SN^* and re-identify the graph even if edges are removed or perturbed, too.

$$R_\ell = \frac{\mid v_i.\ell = 0 \mid}{\mid v_i.\ell = 1 \mid} \tag{3.3}$$

$$R_{l=0} = \frac{|v_i.\ell = 0|}{n} \tag{3.4}$$

$$R_{\ell=1} = \frac{|v_i.\ell = 1|}{n} \tag{3.5}$$

4 Proposed Probabilistic Inference Attack (PIA)

This section will explain our contribution in detail. First, we will define our assumptions. Then we will describe the evolution of our methods. Finally, we will explain our algorithm in detail.

4.1 Methodology

Our aim is, when given K_1, K_2, to find the probability for any suppressed node, of having label 1. However, this would have caused a recursive call cycle as the number of suppressed nodes increased. If a suppressed node has a suppressed neighbour, that also has suppressed neighbours and continuing like this, we will recursively reach every node by neighbour relations and keep on going until we reach the last node or worst, if there is a cycle, never reach an end. Hence we changed our model to a heuristic one, where instead of considering all suppressed nodes at once, we look one by one and update probabilities accordingly. The heuristic model, which will be explained shortly, relies on single comparisons and is faster. So our proposed algorithm works in an iterative mode using the distribution of the labels, but consists of different phases for computing the inference rates.

We assume that we obtained a part of a SN with n nodes, that can be classified into m labels.

Table 9: Defined Symbols used in the algorithm

Value	Symbol
Number of nodes	n
Number of labels	m
Label set	ℓ
Unique node	v_i
List of nodes	$L[n]$
Number of connections of node	N_i
Connected nodes of node v_i	F_{ij}
Unique connection of a node	NF_k
Unique label	ℓ_j
Ratio of connections a node having same label value	RN_j
Label of node	$v_i.\ell$
Inference ratio of a node	IRN_i
Anonymization rate	A

4.1.1 Anonymization Process

This anonymization process is developed for testing of the attack algorithm, which will be explained in Section 4.2. While creating the network graph or getting it as an input by randomly selecting some nodes anonymization can be performed. In Algorithm 6 we showed how a single label classification can be generated. Using the same algorithm with an addition we can create an anonymized version of the network. It would have the same number of friends per node and the same connections.

We will use a user input to determine the rate of anonymization we will produce. If a random value is smaller than the rate then, that node will be anonymized, except it would be stored in a different list. It is detailed in Algorithm 1.

Algorithm 1 Anonymization Process in Network Generation

```
(1)  a ← 0
(2)  while a < n do
(3)      b ← readLineFromFile(filePath)
(4)      numberOfFriends ← random(n/100)
(5)      label ← random(0, 1)
(6)      if label < P(ℓ) then
(7)          nodeLabel ← 0
(8)      else then
(9)          nodeLabel ← 1
(10)     end if
(11)     anonymization ← random(0, 1)
(12)     if anonymization < A then
(13)         L′[a] ← node(b, ∗, numberOfFriends)
(14)     else then
(15)         L′[a] ← node(b, nodeLabel, numberOfFriends)
(16)     end if
(17)     L[a] ← node(b, nodeLabel, numberOfFriends)
(18)     a ← a + 1
(19) end while
```

When using real data instead of generated synthetic data, then this process is done after the nodes are created. This is because of the differentiating input methods. In the real data version, each node has a seperate file for its connections, also nodes from each label outcome are seperated in different files, e.g. class1.txt, class2.txt, etc. In addition to that, since these parts of the algorithm are for generating test cases, they will be excluded when a

real anonymized dataset is going to be used. As mentioned in Section 2 the suppressed data will mimic the original data, due to the reason that even when anonymized this set of records must make sense, without breaching the privacy of individuals.

4.2 Algorithm

In this section we will describe the PIA algorithm in depth. The PIA algorithm shown in Algorithm 4 is designed for a singe label classification, e.g. {age<30, age>=30}, {left-wing, right-wing}, etc. This algorithm runs on the anonymized network data. It searches for anonymous nodes and calculates its probability of belonging to a class.

Our algorithm consists of three parts. First, it finds the unlabeled nodes in the network. After that for each unlabeled node, it checks, how many unlabeled/suppressed friends the node has, depending on the outcome it choses one of two probability functions and computes the probability of this node belonging to a class. Finally, after each unlabeled node has computed a probability, it compares these with the threshold values and comes to a decision about the node.

The first part of the algorithm, shown in Algorithm 2, works in an iterative fashion. It goes over the list of nodes, and searches for the ones that are suppressed, explained as in Algorithm 1.

Algorithm 2 Get Suppressed Nodes

```
(1)  a ← 0
(2)  while a < n do
(3)      if F_ia.ℓ = * then
(4)          L'.push(L[a])
(5)      end if
(6)      a ← a + 1
(7)  end while
(8)  return L'
```

Algorithm 3 Total number of suppressed nodes in the graph

```
(1)  a ← 0
(2)  i ← 0
(3)  b ← 0
(4)  while i < n do
(5)      while a < N_i do
(6)          if F_ia.ℓ = * then
(7)              b ← b + 1
(8)          end if
(9)          a ← a + 1
(10)     end while
(11)     i ← i + 1
(12) end while
(13) return b
```

The second part of the algorithm uses the nodes returned from part 1 with two different probability equations, depending on the number of suppressed connections the selected node has. If the node in question isn't connected to any other suppressed node, then the inference is based purely on the distribution of its connected peers. As equation 4.1 describes, we check the labels of each connected node to determine the connectivity of our selected node to this label. Again it is a single label classification version of the equation.

Then by comparing these values with each other and the probabilities of each class occurance rates, as in Equation 4.2, we can infer the label value this node belongs to.

$$\begin{cases} c1 = N^0_{v_i} \\ c2 = N^1_{v_i} \end{cases} \tag{4.1}$$

$$\frac{c1}{c2} \begin{cases} > \frac{1-K^1_{2,v_i}}{1-K^0_{2,v_i}} \Rightarrow & v_i.\ell = 0 \\ < \frac{1-K^1_{2,v_i}}{1-K^0_{2,v_i}} \Rightarrow & v_i.\ell = 1 \end{cases} \tag{4.2}$$

Aside from this case, a node may be connected to other suppressed nodes. In this scenerio the ratio equation changes, which also varies according to the connected nodes' label values. Depending on the count of nodes' label values from Equation 4.1 we choose one of the ratio calculation methods. If $c1 > c2$ then we use Equation 4.3. If $c2 > c1$ then we use Equation 4.5. The result of these equations are kept in a different list and will be updated after each turn for all suppressed nodes are finished.

$$IRN'_i = IRN_m \times \frac{\left(\mid N^0_{v_i} \mid + \sum_{t=0}^{N_{v_i}} IRN_t \right)}{\mid N_{v_i} \mid} + (1 - IRN_m) \times W_1 \tag{4.3}$$

$$W_1 = \left(1 - \frac{\left(\mid N^1_{v_i} \mid + \sum_{t=0}^{N_{v_i}} IRN_t \right)}{\mid N_{v_i} \mid} \right) \tag{4.4}$$

$$IRN_i' = IRN_m \times \frac{\left(\mid N_{v_i}^1 \mid + \sum_{t=0}^{N_{v_i}} IRN_t\right)}{\mid N_{v_i} \mid} + (1 - IRN_m) \times W_0 \qquad (4.5)$$

$$W_0 = \left(1 - \frac{\left(\mid N_{v_i}^0 \mid + \sum_{t=0}^{N_{v_i}} IRN_t\right)}{\mid N_{v_i} \mid}\right) \qquad (4.6)$$

The key element of the PIA algorithm is that it runs iteratively and as it repeats itself the IRN for each node converges to a value. After the algorithm is finished these ratios would be used to compare it with the threshold values. In Algorithm 4 we can see that the previously mentioned equations are called within the algorithm using the names $Eq\_4.X()$. Until line `(6)` of the PIA algorithm we handle the case, where the suppressed node has no anonymized connections.

The second case, in which a node has also suppressed connections, is run iteratively in order to see the convergence of the inference ratio for each anonymous node. If a suppressed node v_i is connected to another suppressed node v_j then during the runtime of the algorithm the new value of IRN_i is calculated w.r.t. IRN_j and vice versa shown in Equation 4.3 and 4.5. Since it utilizes a probabilistic method, the more this equation is calculated at a single step the more precise the ratio converges. When the suppression rate is low it important that this part of the algorithm iterates sufficient number of times, since the size of the set of suppressed nodes with connections to other suppressed nodes will be low, and we must guarantee that for each

such node the algorithm iterates z times, where z is the user input for the minimal iteration count.

Algorithm 4 Probabilistic Inference Attack

```
(1)  a ← 0
(2)  while a < n do
(3)      x[] ← getSuppressedNodes(v_a)
(4)      if count(x[]) = 0 then
(5)          Eq_4.2 (Eq_4.1(n_a))
(6)      else then
(7)          y ← 0
(8)          z ← 0
(9)          while z < 15 do
(10)             y ← 0
(11)             while y < count(x[]) do
(12)                 cr[] ← Eq_4.1(n_a)
(13)                 if cr[0] > cr[1] then
(14)                     Eq_4.3(n_a, x[y])
(15)                     crTemp[] ← Eq_4.1(x[y])
(16)                     if crTemp[0] > crTemp[1] then
(17)                         Eq_4.3(x[y], v_a)
(18)                     else then
(19)                         Eq_4.5(x[y], v_a)
(20)                     end if
(21)                 else then
(22)                     Eq_4.5(v_a, x[y])
(23)                     crTemp[] ← Eq_4.1(x[y])
(24)                     if crTemp[0] > crTemp[1] then
(25)                         Eq_4.3(x[y], v_a)
(26)                     else then
(27)                         Eq_4.5(x[y], v_a)
(28)                     end if
(29)                 end if
(30)                 y ← y + 1
(31)             end while
(32)             z ← z + 1
(33)         end while
(34)     end if
(35)     a ← a + 1
(36) end while
```

As we can see each pair of suppressed neighbours are calculated together. In other words, for each $v_i.\ell = *$ and $e = (v_i, v_j)$ where $j \in \mathbb{N}$ and $v_j.\ell = *$ we will write the equation 4.3 or 4.5 and recalculate it z times. If we had gone with the deterministic model, we would have to write the equation such that, that each $v_j.\ell = *$ should be a part of it. In the mean time we should also write the eqution in the same loop for each v_j and their suppressed neighbours, and then their neighbours, too. As we can understand this method will call itself recursively to find all suppressed vertices of a give $v_i.\ell = *$. Hence the runtime difference will increase polynomially between the two algorithms and there is the probability of creating an infinite recursion in the deterministic algorithm.

Once the algorithm runs and calculates each suppressed nodes' probabilistic value, the inference decision is based on two thresholds. The threshold selection is based on the fact of seperation of probabilities each suppressed vertice will have after the algorithm runs. We will have two thresholds, one to represent $\ell = 0$ for a given vertice and the other to represent $\ell = 1$ again for any given vertice. Our aim is to succeed in seperating the probabilities very distinctly, hence when choosing a very small and a very big threshold we will identify each vertices' label value with the highest accuracy. For example, if the threshold values are chosen as $t_{small} = 0.25$ and $t_{large} = 0.75$ and there is no or very few vertices inbetween, then we say we have concluded the labels with high accuracy for each vertice. Any probability between the thresholds will be considered as non inferable.

Depending on the relation between vertices that have the label or not (Equation 3.3), the threshold values can be changed. The values can converge to each other at $t_{small} = t_{large} = 0.5$. The more apart these values are with low error rates then the seperation of labels is more precise and accurate.

4.3 Complexity Analysis

In our algorithm we expect to see a polynomial increase of degree 2, in other words our algorithm will run in $O(n^2)$. The reason for this is the two-way calculation of the PIA algorithm. As mentioned in Section 2.1, if a suppressed node v_i is neighbours with other suppressed nodes, e.g. v_j, then PIA calculates the new probability (Equation 4.3 or 4.5) of both vertices. Between lines `(13)-(20)`, `(21)-(28)` of Algorithm 4 we can see that depending on the number of friends with $\ell = 0$ and $\ell = 1$ the Equation 4.3 or 4.5 is calculated for both set of nodes: $(v_i, v_j) \wedge (v_j, v_i)$. When the number of suppressed nodes increases the possibility of a node v_i having more than 1 suppressed neighbour nodes. Considering that this calculation is repeated iteratively for each suppressed node z-many times, the amount of calculation gets bigger and bigger. The complexity of it can be seen in Equation 4.7, which proves that when the total number of suppressed nodes (Algorithm 3) increases the number of suppressed nodes per suppressed node v_i ($\mid Algorithm\, 2 \mid$) also increases. Since the second part is repeated z times, in total the number of calculations increases exponentially. Considering the values from Table 10 and 13 we can conclude Equation 4.7 into Equation 4.8, which gives a numerical representation of the complexity using the average number of con-

nections. $\phi(n,\, z)$ calculates the constant variables for a given network size n and iteration count z.

$$Algorithm\_3 \times [z \times (2 \times (\mid Algorithm\_2(v_i) \mid))] \tag{4.7}$$

$$(n \times A_s) \times [z \times 2 \times (averageNumberOfEdges \times A_s)] = A_s^2 \times \phi(n,\, z) \tag{4.8}$$

Table 10: Facts & Figures for synthetic data

Value	
Count of $\ell = 0$	17500
Count of $\ell = 1$	7500
Average number of friends $\forall v_i$	~ 18.5
Minimum number of friends $\forall v_i$	5
Maximum number of friends $\forall v_i$	33

Table 11: Facts & Figures for real data

Value	
Count of $\ell = 0$	93
Count of $\ell = 1$	690
Average number of friends $\forall v_i$	~ 39.55
Total connectivity $\mid e_{ij} \mid$	30970
Maximum number of mutual friends between v_1 and v_i	207

Considering the complexity eqution (Equation 4.7) both synthetic and real data application of PIA have the same complexity, yet the real data complexity functions' coefficient $\phi(n_{real},\, z)$ is smaller. Using Equations 4.9 and 4.10 we prove the difference of complexity between synthetic and real data, i.e. between $\phi_{synth}\, and\, \phi_{real}$ (Equation 4.11).

$$\frac{n_{real}}{n_{synth}} = \frac{783}{25000} \approx 0.031 \tag{4.9}$$

$$\frac{AverageFriendCount_{real}}{AverageFriendCount_{synth}} = \frac{39.55}{18.5} \approx 2.14 \tag{4.10}$$

$$\phi(n_{real},\, z) = Eq\_4.9 \times Eq\_4.10 \times \phi(n_{synth},\, z) \approx 0.67 \times \phi(n_{synth},\, z) \tag{4.11}$$

5 Performance Evaluation

In this section, we will discuss our test cases and the results of recovering the information using probabilistic inference attack, PIA.

5.1 Test Bench

We have developed our algorithm in Eclipse [22], in order to have a cross-platform application. Any computer on JRE 1.5 or newer is able to run the code.

Besides, the simulations are run on a personal computer with the following specifications:

- Mac OS X 10.7 (x86/64)
- Intel Core 2 Duo Processor at 2.4GHz
- 4 GB 667 MHz DDR2 SDRAM
- JRE 1.5
- Eclipse 3.6 “Helios”

5.2 Test Cases

As mentioned in Section 4.1, we used two different datasets. One was generated from a Facebook [3] crawler data [18] and the other one is populated from a Facebook account.

5.2.1 Synthetic Data Creation

To create a random network, a list of node names can be used. In our case, we used a file, publicly known as the "*100 Million Facebook List*" [18], which is a text file with just above hundred million usernames and the number of repetitions of these usernames, shown in Table12. The original username list consists of 170879859 names and their links.

Table 12: Part of file for synthetic data creation

17204	john smith
7440	david smith
7200	michael smith
6784	chris smith
6371	mike smith
6149	arun kumar
5980	james smith
5939	amit kumar

Using this data, we can generate a seperate file with n records. We randomly select a number in the range [0, 100128458] and retrieve the record on that line. According to the occurance number, we project it to the size

of data we are generating, which is n. The details can be examined from Algorithm 5.

Algorithm 5 Random Network Generator with n records

```
(1)  a ← 0
(2)  x ← 100128458
(3)  y ← 170879859
(4)  while a < n do
(5)      b ← random(x)
(6)      c[] ← readLine(filePath, b)
(7)      repeatCount ← c[0]×n / y
(8)      z ← 0
(9)      while z < repeatCount do
(10)         writeToFile(outputFile, c[1])
(11)         z ← z + 1
(12)     end while
(13)     a ← a + 1
(14) end while
```

The generated output file alone would not be enough to run the test. In order to map this file to a network we must also generate connections between the nodes. In addition to that a classification must be done while converting the file into a network. For each node created we randomly select how many friends, i.e. connections, it would have, which is shown in Algorithm 6. User will input how many labels there will be and their probabilities. According to these probabilities each node will have a label or set of labels. After all nodes are created the number of friendship connection are fulfilled. During this process user input decides on the correlations of the connections, i.e. the user sets how many of which class the node will have a connection to. This part is explained in Algorithm 7.

Algorithm 6 Random Friendship Generation with single label

```
(1)  a ← 0
(2)  while a < n do
(3)      b ← readLineFromFile(filePath)
(4)      numberOfFriends ← random(n/100)
(5)      ℓabel ← random(0, 1)
(6)      if label < P(ℓ) then
(7)          nodeLabel ← 0
(8)      else then
(9)          nodeLabel ← 1
(10)     end if
(11)     L[a] ← node(b, nodeLabel, numberOfFriends)
(12)     a ← a + 1
(13) end while
```

Using Algorithm 5 we can generate a smaller version of the source. Since the source is a very big data in size, (approx. 2.5GB) read operation, combined with the random line seek is very costly. We were able to read a random line and append it to the new file in 8.93 seconds on average. Thus creating a list of 25000 nodes took around 62 hours (Figure 4). This algorithm is the most costly part of the whole project.

In order to mimic the actual data, we analyzed the facts of Facebook from back the time, in which this crawler was active. First, a twenty-five-thousand line data is generated by using Algorithm 5. The generated data is read and converted into a graph using the facts from Table 10 and with Algorithms 6 and 7. These values are calculated according to the ratio of the size of generated data against the size of the SN itself [11]. Then these values are randomly selected for suppression as shown in Algorithm 1.

Algorithm 7 Connection Generator with single label

```
(1)  a ← 0
(2)  b ← 0
(3)  x ← 0
(4)  while a < n do
(5)     while b < N_a do
(6)        sameLabelConnections ← N_a × RN_{v_i.ℓ}
(7)        while c = v_a do
(8)           c ← random(n)
(9)           if v_c.ℓ = v_a.ℓ and
                 x < sameLabelConnections then
(10)             addConnection(v_a, v_c)
(11)             x ← x + 1
(12)          else if x >= sameLabelConnections and
                 v_c.ℓ ≠ v_a.ℓ then
(13)             addConnection(n_a, n_c)
(14)          end if
(15)       end while
(16)       b ← b + 1
(17)    end while
(18)    b ← 0
(19)    a ← a + 1
(20) end while
```

5.2.2 Real Data

The real data is based on one Facebook account. The friends of the account and the mutual friend list for each of the friends of the account have been gathered. The list of friends are stored in a file, in which every friends' user-ID, that acts as a unique key [13] for each account on Facebook, is stored. Assume v_1 is the account that the information is harvested from, and the mutual friends of a friend v_i of v_1 is stored just like the friends list file.

```
Turn number: 24995
Time elapsed: 17.666840076447
---------------------------------------
Turn number: 24996
Time elapsed: 13.213876008987
---------------------------------------
Turn number: 24997
Time elapsed: 8.5846939086914
---------------------------------------
Turn number: 24998
Time elapsed: 4.9137871265411
---------------------------------------
Turn number: 24999
Time elapsed: 3.2475779056549
---------------------------------------
total time:     223350.36390996
avg time:       8.9340145563984
```

Figure 4: Algorithm 5 runtime

In order to gather this information, a crawler script is run on the account of v_1. The crawler uses the Graph API [5] and the REST API [10] from the Facebook Developers library [4]. First the crawler retrives the user-ID's of the given account and then for each returned user-ID it collects the mutual friends list (Algorithm 8). This data is also suppressed randomly according to Algorithm 1. Table 11 shows the important values of this dataset.

5.3 Results

In this section present the results of the tests according to the datasets mentioned above. For each dataset, we show the success rate and the runtime of the attack algorithm. We test both datasets with different suppression rates. We test the PIA algorithm based on a single-label inference. In other words,

Algorithm 8 The Facebook Crawler

```
(1)  friendList[] ← friends.get(accountID)
(2)  i ← 0
(3)  while friendID ← friendsList[i] do
(4)      mutualList[] ←
           friends.getMutualFriends(accountID, friendID)
(5)      writeToFile(accountIDFile, friendID)
(6)      j ← 0
(7)      while mutualID ← mutualList[j] do
(8)          writeToFile(friendIDFile, mutualID)
(9)          j ← j + 1
(10)     end while
(11)     i ← i + 1
(12) end while
```

the aim is to determine a single information of each node like "Is this person an enrolled student". If the person is an active student $v_i.\ell = 1$ will hold and if opposite then $v_i.\ell = 0$ will hold.

5.3.1 Synthetic Data Results

As described in Table 10, the synthetic data has 25000 nodes and on average each node has 18.5 edges. The minimum number of friends on any node is 5 and the maximum is 33. In this test case 70% of the data generated has $\ell = 0$. The average time required for generating the graph and suppressing it is around 0.67 seconds.

To evaluate performance, we vary the number of suppressed nodes in the synthetic data and in Table 13 plot the average time it takes to run the PIA algorithm. Visualizing the runtimes in Figure 5 shows that the runtime of the

PIA algorithm increases, with increasing suppression rates, as a polynomial function [9] of degree 2, i.e. $O(n^2)$.

Table 13: Suppression rates for synthetic data

A	%	Avg. time (sec)
A_1	0.15	1.489
A_2	0.25	5.273
A_3	0.35	14.759
A_4	0.50	33.568
A_5	0.60	58.663
A_6	0.75	116.567
A_7	0.875	204.382

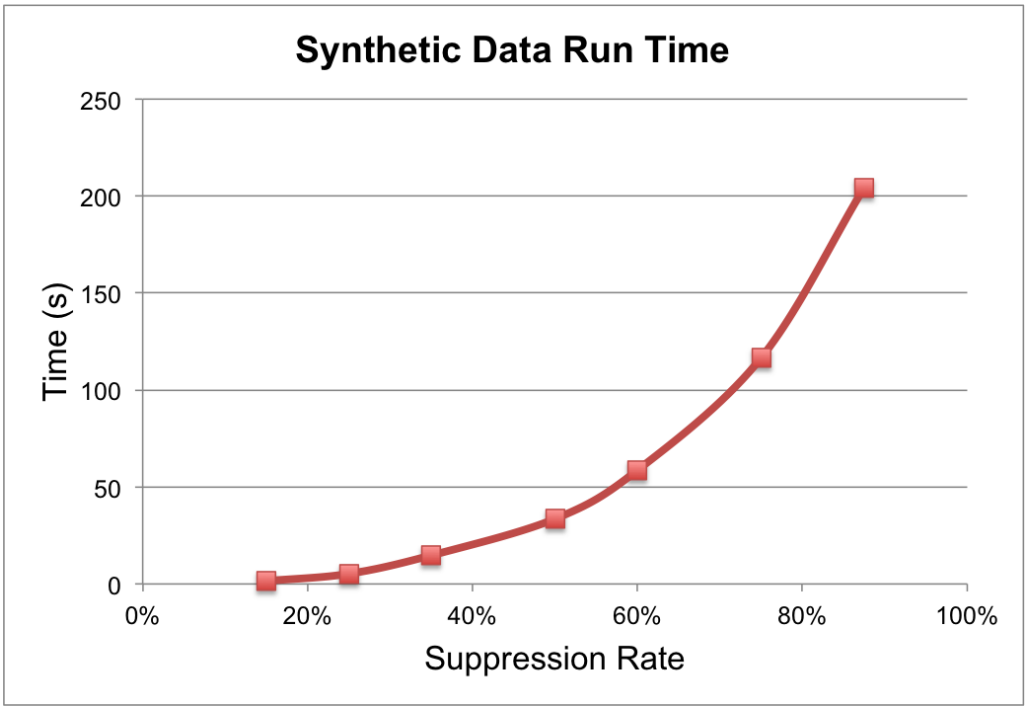

Figure 5: Run Time of PIA with Synthetic Data

Nevertheless when running a suppression or re-identification algorithm, time is not the major issue, due to the fact that these algorithms have to be executed only once. But when these algorithms operate only once for every dataset the key issue becomes that the end product matches the expectations of its context. Since the PIA algorithm is trying to re-identify labels of nodes it is important that it should finish its execution with very low rate of errors. We can see in Figures 6 and 7, that the error rate (i.e. the percentage of node labels predicted incorrectly) also has a polynomial equation of degree 2. We see that as the suppression rate A_s increases the error rate increases as well. This is because the more knowledge the adversary has, the less errors would be made in the re-identification process since there are more known nodes that can be used to infer on the suppressed nodes. In the results of PIA algorithm, we can clearly see that there is a turning point after $A = 0.75$, where the error rates take a steep rise. However, even when the graph is 75% suppressed the total number of falsely identified nodes are just over 1000, i.e. the error rate is slightly over 5%. In other words PIA was able to re-identify $\approx$ 17250 of 18500 suppressed nodes and match almost 23750 nodes exactly to the original SN.

In order to evaluate the success rate , we must compare the results with the a naive adversary. Let us assume that there is another adversary that also knows 70% of all nodes have the label $v_i.\ell = 0$. Using this fact the adversary could try to re-identify all label, but infers all $v_i.\ell = 1$ as $v_i.\ell = 0$, reaching the biggest possible error rate, 30%. Hence when we plot our results for very high rates of suppression, we are very far away from the worst case error rate 30%.

Looking at the label outcomes individually (Figure 8) shows an inharmonious error rate. Although the result of $\ell = 0$ mimics the overall results, the error rates of $\ell = 1$ are on a different path. This is caused by the outcome of the distribution between $\ell = 0$ and $\ell = 1$, which is in this case 70% and 30%, respectively. One must also take into account that people with $\ell = 1$ also tend to be connected to people with also $\ell = 1$. Since the vertices with $v_i.\ell = 1$ are less in number, and internally connected with each other, the error rate difference heightens slower.

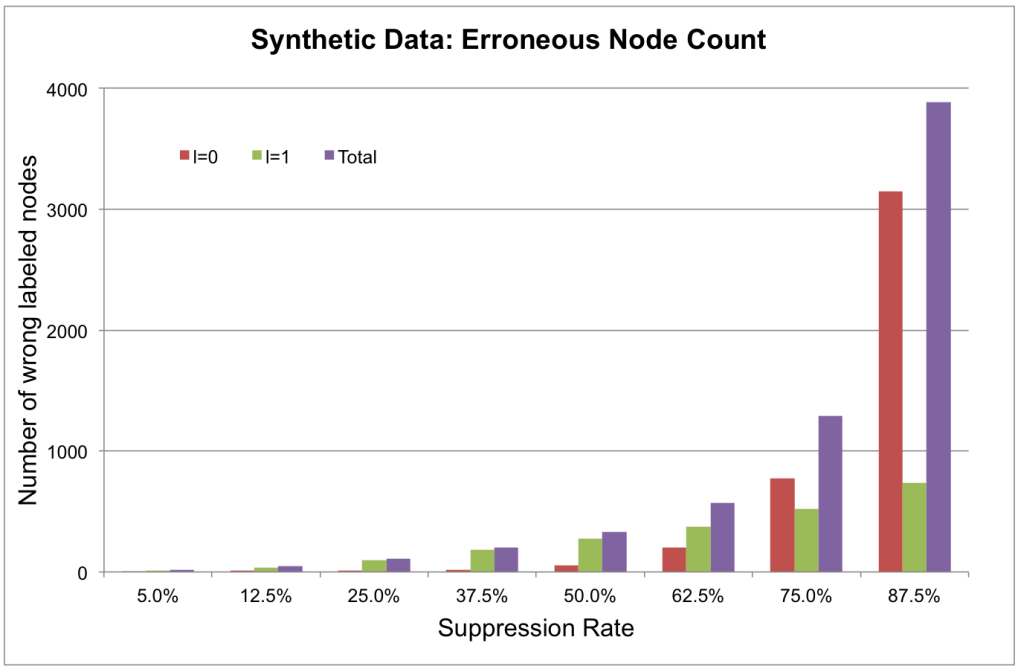

Figure 6: Erroneous Node Count with Synthetic Data

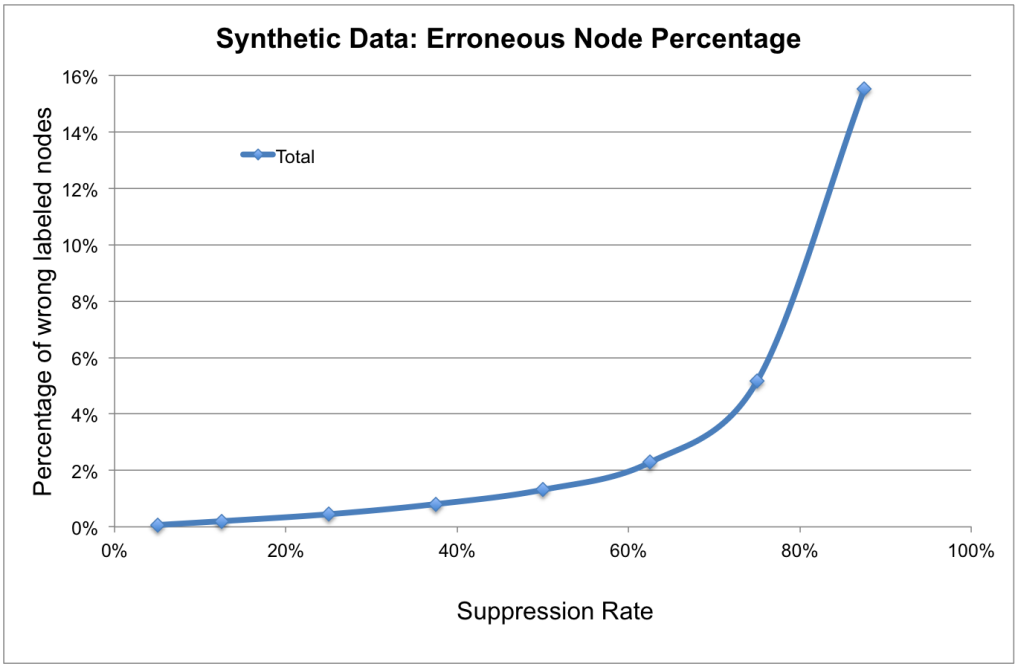

Figure 7: Erroneous Node Percentage with Synthetic Data

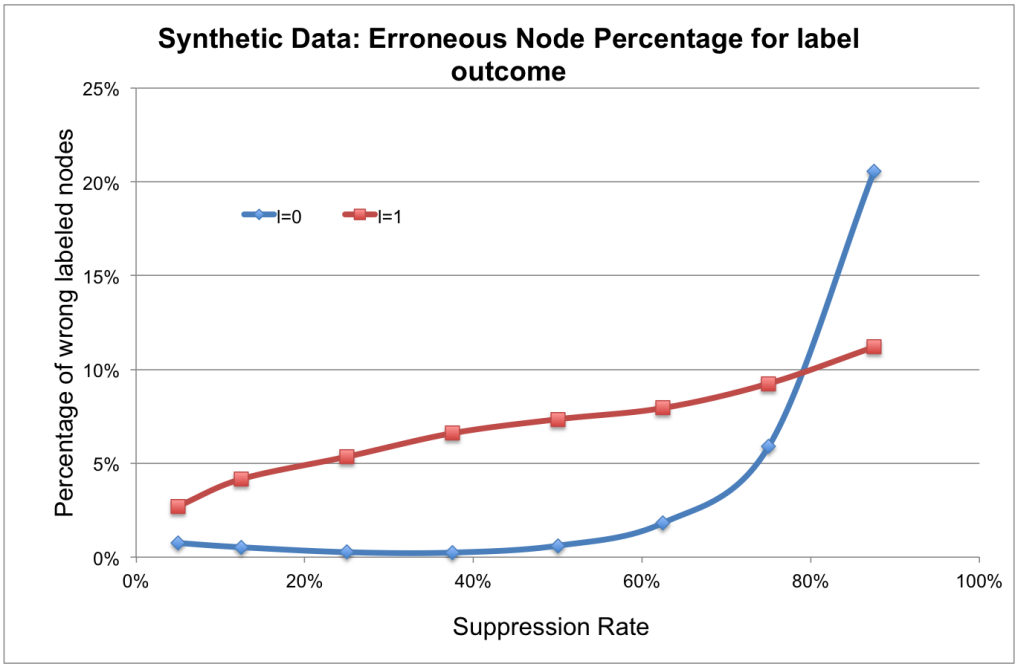

Figure 8: Erroneous Node Percentage in Synthetic Data for label outcome

5.3.2 Real Data Results

Although the real world data is small in size, its tests have similar outcomes as the ones on the synthetic data. Table 11 describes the smaller size, $n_{real} = 783$. We must point out that the average number of connection per node v_i is approximately 40, in contrast to the 18.5 of the synthetic data. The reason for this situation is that the real data is populated from one singular profile and its mutual relations with other nodes. The doubling in node connectivity increases the precision of the inference, despite the small size of the graph. In this test the label $\ell = 1$ represents nodes with $age \leq 30$.

The runtime of PIA with the smaller real data shows also a polynomial increase with the increasing A_i value (Table 14). Comparing Figure 9 with Figure 5 one can conclude that the degree of the polynomial is slightly smaller in the real data case.

Table 14: Suppression rates for real data

A	%	Avg. time (sec)
A_1	0.15	0.157
A_2	0.25	0.350
A_3	0.35	0.621
A_4	0.50	1.191
A_5	0.60	1.669
A_6	0.75	2.254
A_7	0.875	3.229

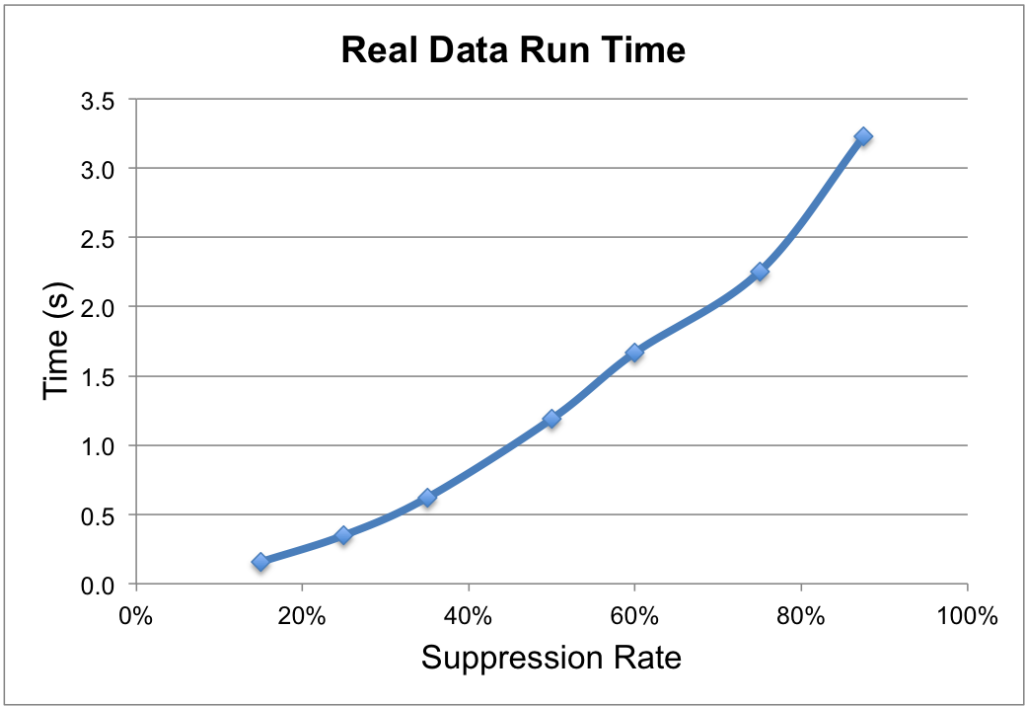

Figure 9: Run Time of PIA with Real Data

In the real world dataset test we see (Figures 10 and 11) a more linear rate of increase in the erroneous node count. As explained this is caused by the size of the data (Equations 4.8 and 4.11) and distribution of the labels. In the synthetic data test the distribution was $P(v_i.\ell = 0 \mid N) = 0.7$, and in this real data test it is $P(v_i.\ell = 1 \mid N) = 0.88$. Altogether create this linearity, however choosing a different label may increase the complexity again towards a degree 2 polynomial, but in the case $O(n^x)$ as $lim_{P(v_i.\ell|N)\to 1} x \to 1$, where $x \in [1, 2]$. Besides these the PIA algorithm manages to re-identify more than 90% of the graph even when the suppression rate was $A = 0.75$.

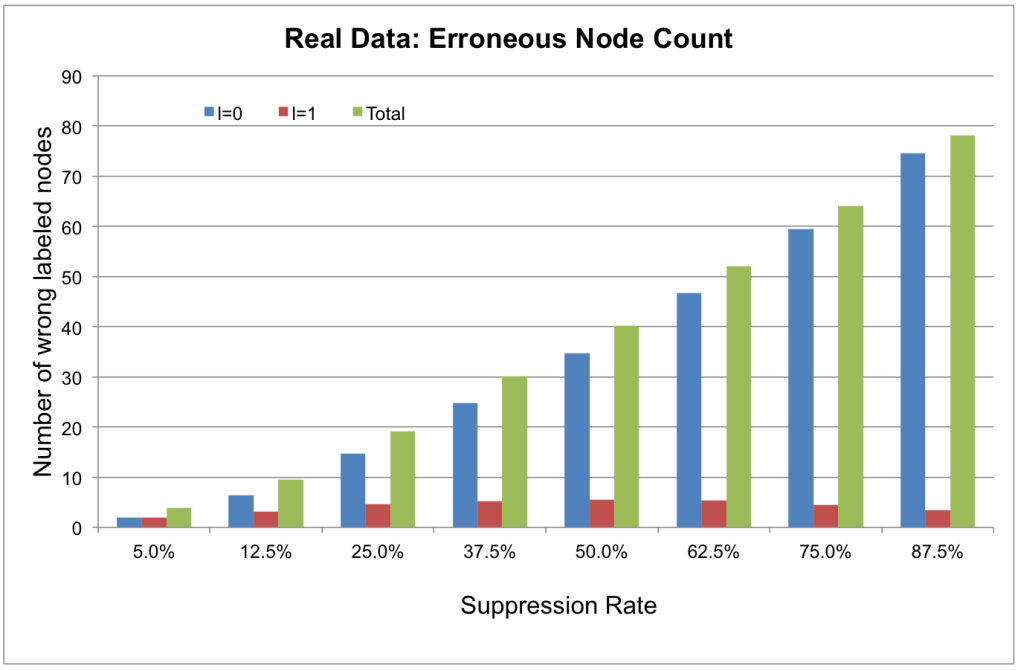

Figure 10: Erroneous Node Count with Real Data

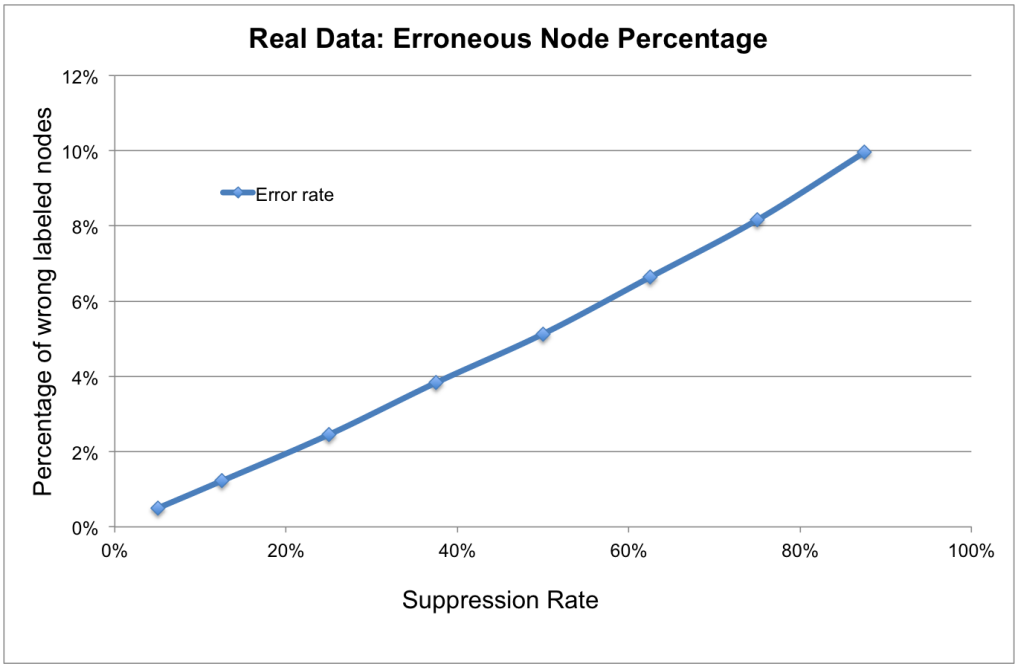

Figure 11: Erroneous Node Percentage with Real Data

Figure 12 shows that both lines are going to the opposite direction, due to the fact that with smaller data the error rates increase faster. If the worst case of $P(v_i.\ell = 0 \mid K_1, K_2) = 0.12$ is calculated as $v_i.\ell = 1$. This is due to the fact, that even when the entire graph is suppressed the adversary is aware of $P(v_i.\ell = 1 \mid N) = 0.88$ and $P(v_i.\ell = 0 \mid N) = 0.12$. Hence the PIA algorithm would favour $v_i.\ell = 1$ almost entirely during its runtime, not taking the probability of $\ell = 0.12$ into account. In such a case the graph would be re-identified with label $\ell = 1$, inferring all 93 of $\ell = 0$ incorrectly and therefore the error rate would converge to 100% for all $v_i.\ell = 0$ and to 0% for all .

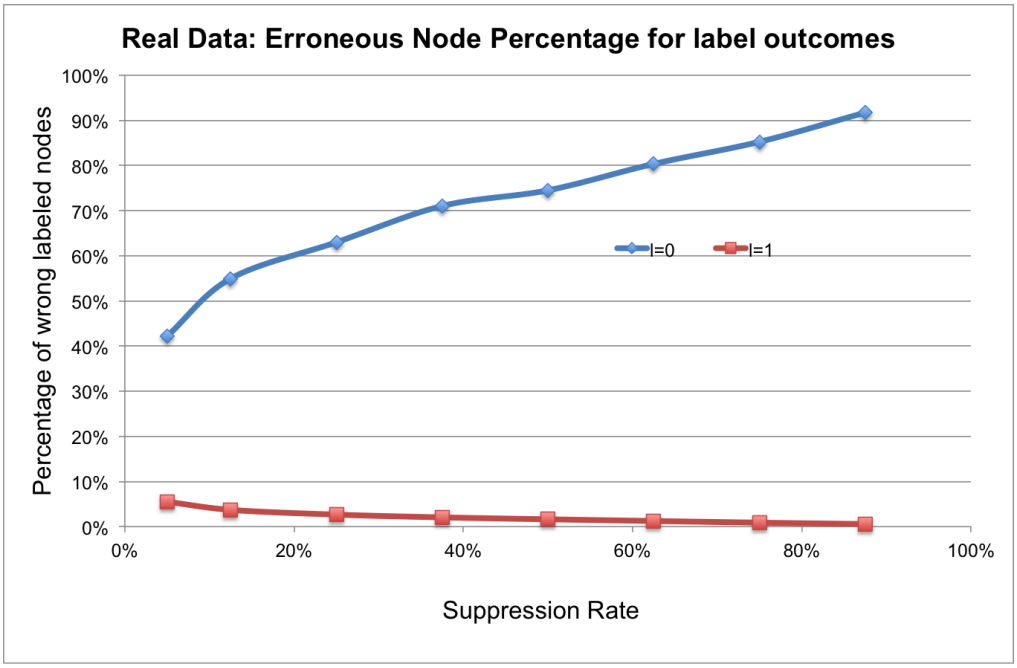

Figure 12: Erroneous Node Percentage in Real Data for label outcome

5.4 Evaluation

The tests show that even with very different data sizes and very different distributions among the datasets our PIA algorithm can re-identify labels for each node with very low error rates. These tests prove our initial belief, that anonymizing the graph by generalizing labels or suppressing edges is not enough to secure personal privacy. We have tested on a big graph with relatively small connectivity and on a small graph with relatively high connectivity. In either case PIA was able to put the graph together with less than 10% error when the graphs were about 80% suppressed.

On a SN an active adversary could gather more information than what we assumed at Section 2.3 and could lower the error rate, but even with only having the information on label distribution an adversary can gain the entire network.

6 Conclusion and Future Work

Social networks are one of the most popular communication tools of our time. In our daily activities, checking our social network activity and sharing thoughts on this platform has become a regular task. However the personal information shared on SNs are not secure. The privacy settings allow account holders to decide what to share and where to share and with whom you want to share. Yet nobody can control what is shared publicly by their friends on the SN. People might be sharing sensitive information about any account holder without considering risks or the account holder can share without realizing the consequences it creates.

Ongoing research on privacy has yielded some anonymization techniques. These methods can handle many cases, where they anonymize individual information, suppress edges, and reorganize nodes and/or edges such that all nodes become similar. Yet the knowledge of adversaries expands, too. Especially SNs, opposite to tabular datasets, have an inferring mechanism, caused by the connections between users. Hence anonymization and suppression becomes even harder. In addition to these facts, one must be able to make sense of the data after it has been anonymized, meaning that the utility must be kept over a threshold such that publishing the data will be useful.

In this thesis we propose an attack algorithm, called the probabilistic inference attack, that exploits this fact of over-sharing [23] or friend-sharing. The adversary would only need the distribution of the label, that he/she is

trying to gain control of. The algorithm iterates over the graph of the SN or its subparts and calculates the probability of each suppressed node of having the label or not.

Our tests show that our proposed algorithm is able to re-identify suppressed nodes with very low error rates. In most cases the algorithm was able to infer the correct labels for more than 90% of the nodes even when the suppression rate was as high as 80%. Our test consisted of big and small datasets with low and rich connectivity with respect to each other.

One part we have not yet concluded is, that as the number of labels increase, how we should modify the algorithm to run each one. In other words we must test all possible variations of coding the algorithm in order to find the version with the highest utility. We must also add that our test data had only suppressed labels on nodes. We tested our algorithm against low level of connectivity, but did not test on removed or suppressed edges.

Hence in the near future we will be concentrating on solving these issues. First of all, we will create a crawler to gather information from various SNs in order to broaden or testing space. Secondly, we will improve our algorithm to infer on multi-label SN datasets with high utility, i.e. based on memory, speed, error/accuracy. Finally, we will test the revised algorithm with the new datasets, which we will suppress and anonymize using different techniques. Despite testing against other anonymization methods, we believe our algorithm will succeed if the anonymized dataset's utility matches a standard value for publishing.

APPENDIX - Actual results of the average erroneous node count and node percentage tests

Table 15: Number of errors and the error rate in synthetic data size 25000

A	Number of avg. errors	Average of errors w.r.t. data size
0.05	$16.\overline{6}$	0.067%
0.125	50.5	0.202%
0.250	111.8	0.447%
0.375	201.5	0.806%
0.500	330.3	1.321%
0.625	572	2.28%
0.750	$1293.1\overline{6}$	5.17%
0.875	3882.8	15.53%

Table 16: Number of errors and the error rate in real data size 783

A	Number of avg. errors	Average of errors w.r.t.
0.05	4.15	0.53%
0.125	11.2	1.43%
0.250	$19.\overline{3}$	2.47%
0.375	29.5	3.76%
0.500	43.5	5.55%
0.625	54	6.9%
0.750	68	8.68%
0.875	81.33	10.38%

References

[1] Binomial distribution. http://en.wikipedia.org/wiki/Binomial_distribution.

[2] Data re-identification. http://itlaw.wikia.com/wiki/Data_re-identification.

[3] Facebook. http://www.facebook.com/facebook?sk=info.

[4] Facebook developers. http://developers.facebook.com/.

[5] Graph api. http://developers.facebook.com/docs/reference/api/.

[6] Graph (mathematics). http://en.wikipedia.org/wiki/Graph_(mathematics).

[7] Information sensitivity. http://en.wikipedia.org/wiki/Information_sensitivity.

[8] Please rob me: Raising awareness about over-sharing. http://pleaserobme.com/.

[9] Polynomial. http://en.wikipedia.org/wiki/Polynomial.

[10] Rest api. http://developers.facebook.com/docs/reference/rest/.

[11] Statistics. http://www.facebook.com/press/info.php?statistics.

[12] Table (information). http://en.wikipedia.org/wiki/Table_(information).

[13] Unique key. http://en.wikipedia.org/wiki/Unique_key.

[14] Web crawler. http://en.wikipedia.org/wiki/Web_crawler.

[15] What is following? http://support.twitter.com/entries/14019-what-is-following.

[16] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA, 2007. ACM.

[17] A. J. Blumberg and P. Eckersly. On Locational Privacy, and How to Avoid Losing it Forever. https://www.eff.org/wp/locational-privacy, Apr. 2009.

[18] R. Bowes. Return of the facebook snatchers. http://www.skullsecurity.org/blog/2010/return-of-the-facebook-snatchers.

[19] D. Boyd and N. B. Ellison. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, Oct. 2007.

[20] E. A. Çiçek. Ensuring location diversity in privacy preserving spatio-temporal data mining. Master's thesis, Sabancı University, 2009.

[21] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, 5:1–16, July 2003.

[22] E. Foundation. Eclipse. `http://www.eclipse.org`.

[23] B. B. Frank Groeneveld and B. van Amstel. Over-sharing and location awareness. `http://www.cdt.org/blogs/cdt/over-sharing-and-location-awareness`, 2010.

[24] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *IJCAI*, pages 1300–1309, 1999.

[25] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2 edition, Jan. 2006.

[26] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing Social Networks. Technical report, 2007.

[27] J. He, W. Chu, and Z. Liu. Inferring Privacy Information from Social Networks. In S. Mehrotra, D. D. Zeng, H. Chen, B. Thuraisingham, and F.-Y. Wang, editors, *Intelligence and Security Informatics*, volume 3975 of *Lecture Notes in Computer Science*, chapter 14, pages 154–165. Springer-Verlag, Berlin/Heidelberg, 2006.

[28] D. Heckerman. A Tutorial on Learning with Bayesian Networks. In D. Holmes and L. Jain, editors, *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*, chapter 3, pages 33–82. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[29] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, Sept. 1995.

[30] E. Kaplan. Privacy leaks in spatio-temporal trajectory publishing. Master's thesis, Sabancı University, 2009.

[31] E. Kaplan, T. Pedersen, E. Savaş, and Y. Saygın. Privacy risks in trajectory data publishing: Reconstructing private trajectories from continuous properties. In I. Lovrek, R. Howlett, and L. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 5178 of *Lecture Notes in Computer Science*, pages 642–649. Springer Berlin / Heidelberg, 2008.

[32] E. Kaplan, T. B. Pedersen, E. Savaş, and Y. Saygın. Discovering private trajectories using background information. *Data and Knowledge Engineering*, 69(7):723 – 736, 2010. Advanced Knowledge-based Systems.

[33] J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. Inferring private information using social network data. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 1145–1146, New York, NY, USA, 2009. ACM.

[34] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3+, Mar. 2007.

[35] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[36] M. A. Mehmet Nergiz and C. Clifton. Hiding the presence of individuals from shared databases. In *2007 ACM SIGMOD International Conference on Management of Data*, Beijing, China, 06 2007.

[37] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 21:1104–1117, 2009.

[38] M. A. Russell. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. O'Reilly Media, 1 edition, Feb. 2011.

[39] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, Oct. 2002.

[40] S. F. V. Ciriani, S. De Capitani di Vimercati and P. Samarati. k-anonymity. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, November/December 2001.

[41] Q. Wei and Y. Lu. Preservation of Privacy in Publishing Social Network Data. In *2008 International Symposium on Electronic Commerce and Security*, pages 421–425, Los Alamitos, CA, USA, Aug. 2008. IEEE.

[42] B. Zhou and J. Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *2008 IEEE 24th International Conference on Data Engineering*, pages 506–515. IEEE, Apr. 2008.