# COMS 6156 Final Project Report : Anonymization of Social Networks

Submitted by:

Team: intelligence (dc3383, sc4401, ik2442)

Dhruv Chamania, Srujan Chinta & Ishan Khandelwal

10th May 2019

# 1 INTRODUCTION

Social Network Analysis can be defined as the process of exploring and analyzing social structures through the use of networks and graph theory.

Networks are characterized in terms of nodes (actors or objects) and links (relationships or interactions) between the nodes. With the rapid growth of social networks, such as Facebook and LinkedIn, more and more researchers found that it is a great opportunity to obtain useful information from these social network data, such as the user behavior, community growth, disease spreading, etc. However, it is paramount that published social network data should not reveal private information of individuals.

Such kind of cases are extremely prevalent during current time where lots of companies give their data to different companies for analysis. The nodes in a social network can have several attributes associated with them. Some of these attributes contain very private information. These nodes are termed as “sensitive attributes”. Though it may appear that not publishing the sensitive attribute or making use of cryptograph measures would ensure the privacy of the sensitive attribute, it is not true. Similarity attacks utilize other published information to identify the individuals and their sensitive attribute. Background knowledge attacks utilize pre-existing knowledge to attack the privacy.

This is where anonymization of social networks comes in. Companies need to make sure that patterns are not recognizable, and all the sensitive attributes and relationship are safe.

A process of anonymization has generally the following flow.

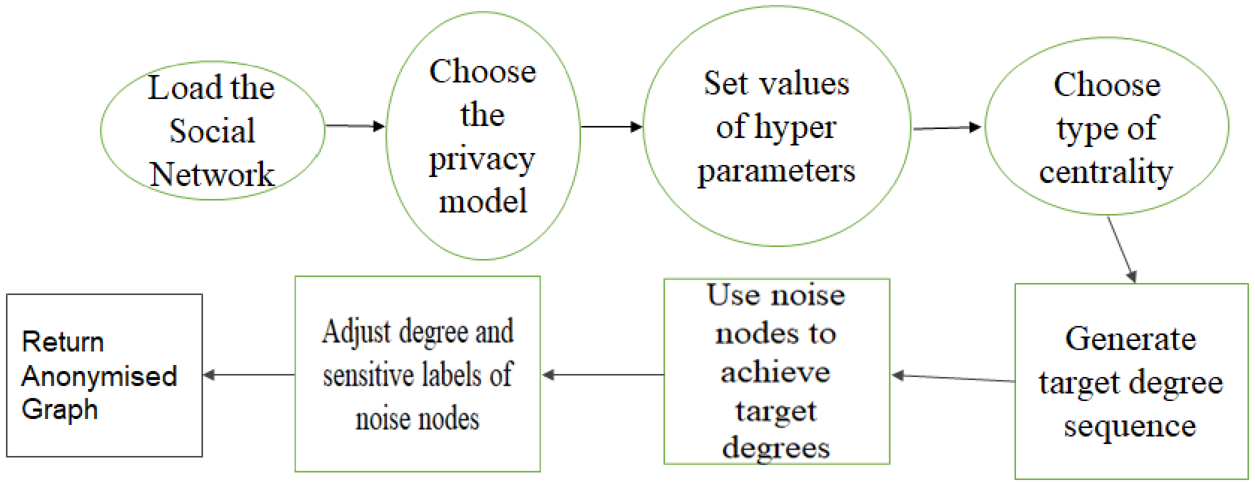

Load the Social Network
Choose the privacy model
Set values of hyper parameters
Choose type of centrality
Generate target degree sequence
Use noise nodes to achieve target degrees
Adjust degree and sensitive labels of noise nodes
Return Anonymised Graph

*Fig 1. Flow-Chart for anonymization using noise node addition algorithm.*

## 2 AIM

We wanted to compare two anonymization methods namely edge editing and noise node edition (Corresponding to label 1 in Fig 2). While doing that we also wanted to compare sub parameters for anonymization namely privacy measures (Corresponding to label 3 in Fig 2). and centrality values (Corresponding to label 4in Fig 2). that could be used. While making these comparisons, we wanted to also analyze these methods from software engineering perspective i.e. analyzing time, memory, effort taken to develop the code.

.

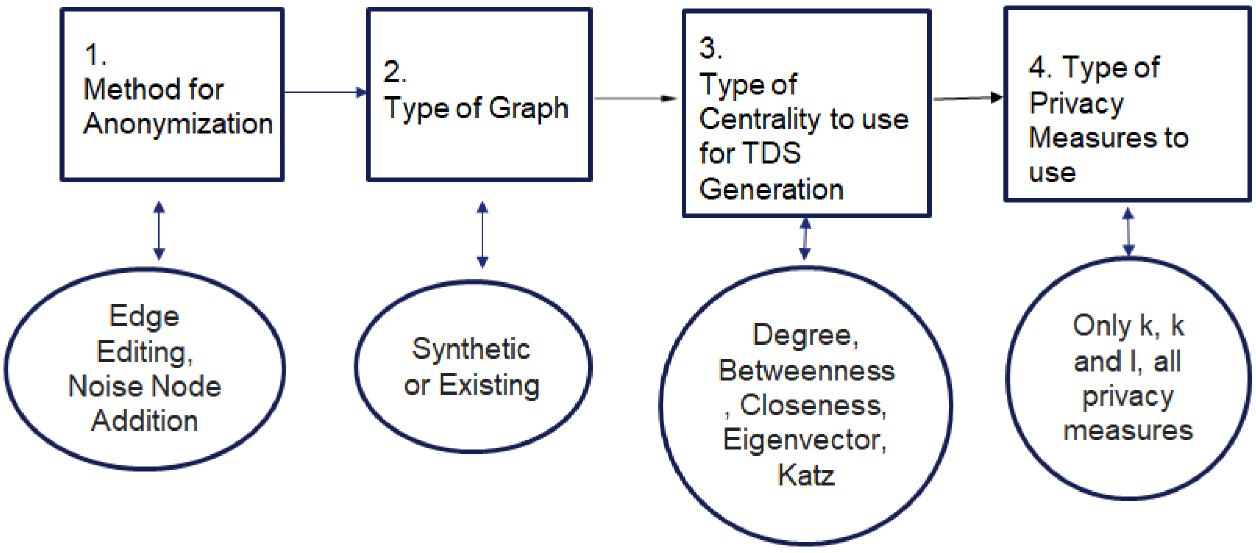

1. Method for Anonymization
2. Type of Graph
3. Type of Centrality to use for TDS Generation
4. Type of Privacy Measures to use
Edge Editing, Noise Node Addition
Synthetic or Existing
Degree, Betweenness , Closeness, Eigenvector, Katz
Only k, k and l, all privacy measures

*Fig 2. Parameters that we wanted to test on*

But there was no sample code base available that could be used to properly analyze different parameters and subfunctions for anonymization. We need a code base where all the different parameters and method seen in the graph above are all properly included.

This is where the novelty of our project comes in. We developed our own code base, which can be used as a base for creating anonymized social networks.

- This code base will be useful for people who need to create anonymized social networks for different data. Our codebase will give them many different options of hyperparameters to choose from they would want to choose from like centrality or privacy measures. *Any organization which aims to release graphical data which includes sensitive information will benefit by using our anonymization model.*
- This code base could act as a base for different kinds of research endeavors where people want to compare anonymization of social graphs with different methods. Researchers and developers can easily build upon our code base because of it being highly modularized.

- We have also included different tests facilities in our code like memory, time and average path length, centrality etc. These are helpful tools that could be used by researchers and software engineers.

Apart from the code base we have developed, the questions below are also research questions that are quite interesting:

1. Which method among noise node addition and edge editing performs better in terms of both algorithmic and ease of software development perspective?
2. Apart from the two methods above, amongst the rest of the parameters namely centrality values, types of privacy measures and values of privacy measures which are the best.
3. Finally, all these methods are for single sensitive labels. So, Can we extend the definition of the privacy measures to account for multiple sensitive attributes without compromising the structural integrity of the original social network while obtaining comparable results to single sensitive labels?

## 3 CODEBASE

Our whole project is located at the link below:

https://github.com/dhruvchamania/TSE_Final_Project

In wiki pages of our code base we have compiled a list of important links related to concepts talked about in the report. A software engineer should look into those in case he is new and not familiar to the concepts discussed in this paper. The wiki and also the README.md consists of rest of the information like software requirements, how to run, function and test description etc. Below we are briefly going over the dataset and testing approach for our code base.

### 3.1 Dataset

The dataset that our codebase uses are of Geography Markup Language (GML) format. To run our code base, it is important to use data only in GML format.

The 4 datasets that used for testing codebase are:

**Karate.gml :** A very small social network dataset consists of friendship between karate member club.

**Netscience.gml:** Social network Dataset of order $10^3$ which is made up of coauthorship network of scientists

**Netscience2.gml:** Social network Dataset of order $10^3$ that is a smaller subset of netscience dataset.

**as-july22july06.gml:** Social network Dataset of order $10^4$ (Restricted testing) is a snapshot of structure of internet for autonomous systems.

We decided to go with 4 datasets for primary testing of our codebase and for the research questions that are we are trying to find answer as it covers a sufficient range of data size for us to conduct our tests on.

## 3.2 Testing

All of the tests were conducted on a PC with 32GB ram and an i7 processor. The file docs/Complete_Summary consists all the data gathered from conducting the tests and a software engineer should download it to see our complete data. And the wiki consists of information about how our tests are structured. Below are some key aspects of it.

1. For each, a single method of anonymization and a single dataset, there were 15 permutation that we tested own. Namely 5 different centralities (Corresponding to label 3 in Fig 2) and 3 different privacy measures (Corresponding to label 4 in Fig 2).
2. We tested all of these methods for 4 different values of the hyper-parameters k anonymity and l diversity.
3. There were 3 datasets that we tested completely. (On the fourth one, we did restrictive testing)
4. So, for any one method, we have tested 180 different times (15x4x3)

And there were 4 different tests that we conducted for each of these permutations:

1. Amount of memory the process consumed.
2. Time taken for the anonymization
3. ***Error in path length ->Very important property of graph, tells us how much has the graph changed. According to network theory it is a key evaluating factor.***
4. Top 20 centrality. Centrality value for top two-degree nodes.

The test data that is compiled here

Of all the test that we conducted, 1&2 are more interesting to a software engineer and 3&4 are more interesting to a researcher. Apart from this, we also compared

- The development difficulty between noise and edge editing
- And also difference of number of nodes/edges for noise node/edge editing respectively.

All the time and memory data that are done on the single pc. They may vary depending on the kind of PC configuration you are using but our testing showed that even though the amount of time taken differs, the observations in the result section remain the same.

**Sample output:**

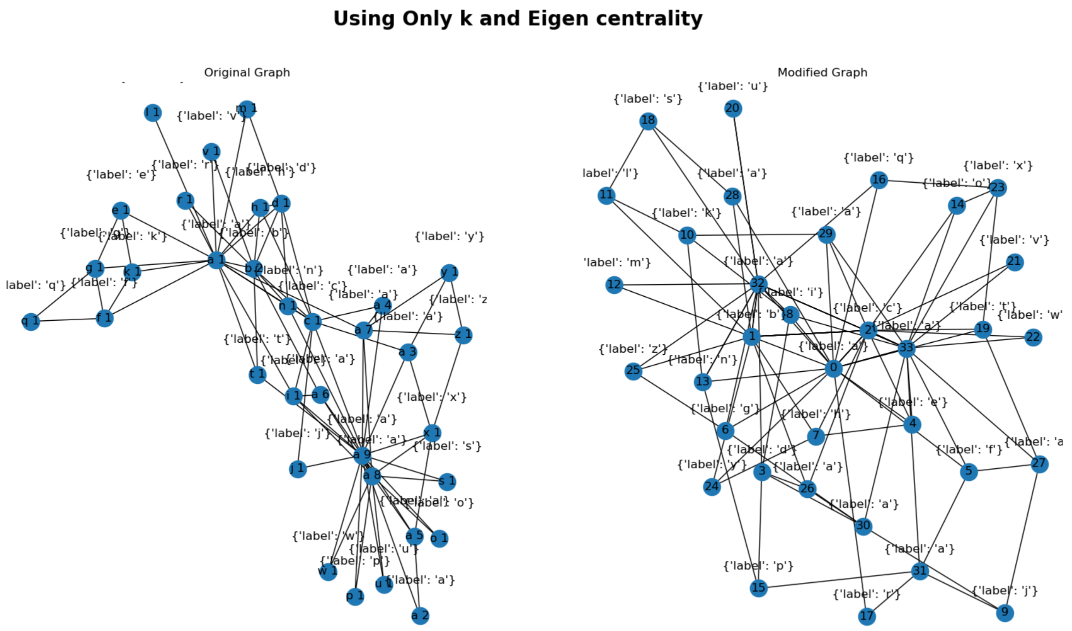


*Fig 3: Edge editing method using only k anonymity privacy measure and eigen centrality. Left original graph, right anonymized graph*

## 4 RESULTS

In this section, we will go over the results that we have obtained against the research questions that we raised in Section 2 and weather from our test results we are able to answer them. (All the data for this can be found in the "docs/Compete_Summary.csv" of our codebase.)

### 4.1 Which is better edge editing or noise node?

In terms of software development complexity, between the implementation of the two anonymization method, noise node was trickier. But our good modularized code helped us in developing these methods without too much difficulty and the difference between the two was not astronomical. Hence can we cannot differentiate the two methods in terms of software development complexity.

In terms of the performance metric, graphs for each of them are as follow, time (*Fig 4*), memory (*Fig 5*), error in path length (*Fig 6*) and top 20 centrality values (*Fig 7*). The values that are shown on these graphs are an average of 60 tests (as discussed in Section 3) that are carried out for each dataset. Important to note is the fact that the data size increase as we are going right in the graph.

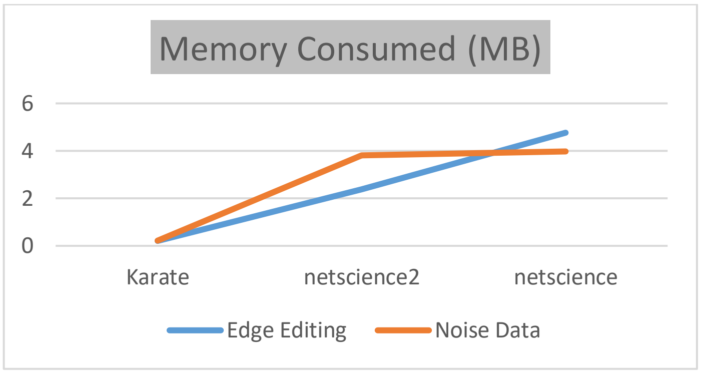


*Fig 3*

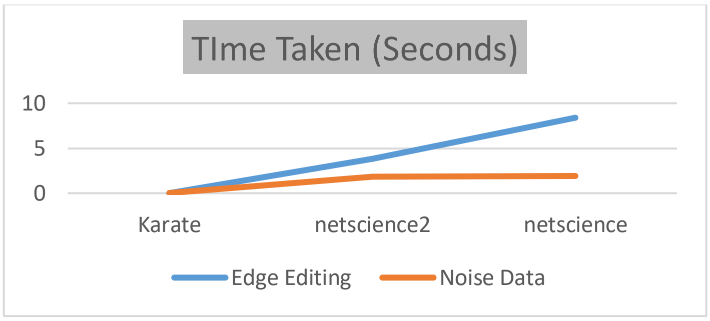


*Fig 4*

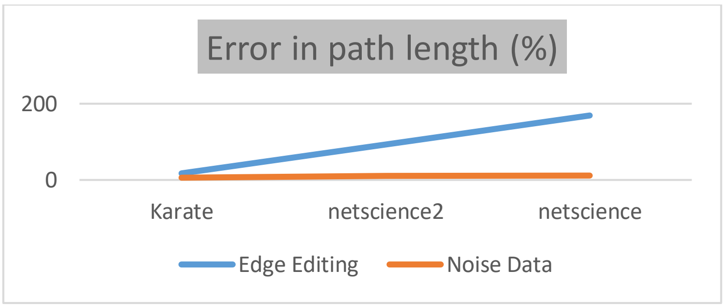


*Fig 5*

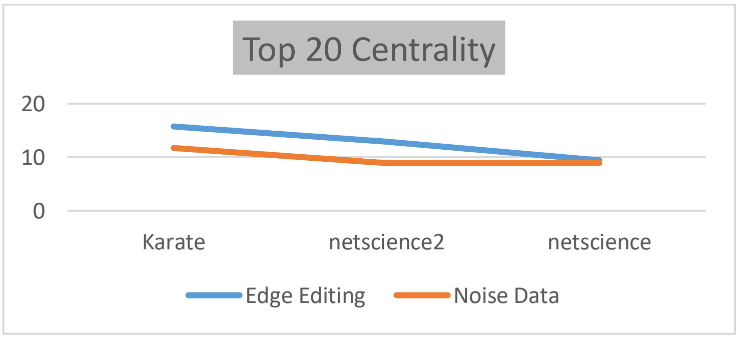


*Fig 6.*

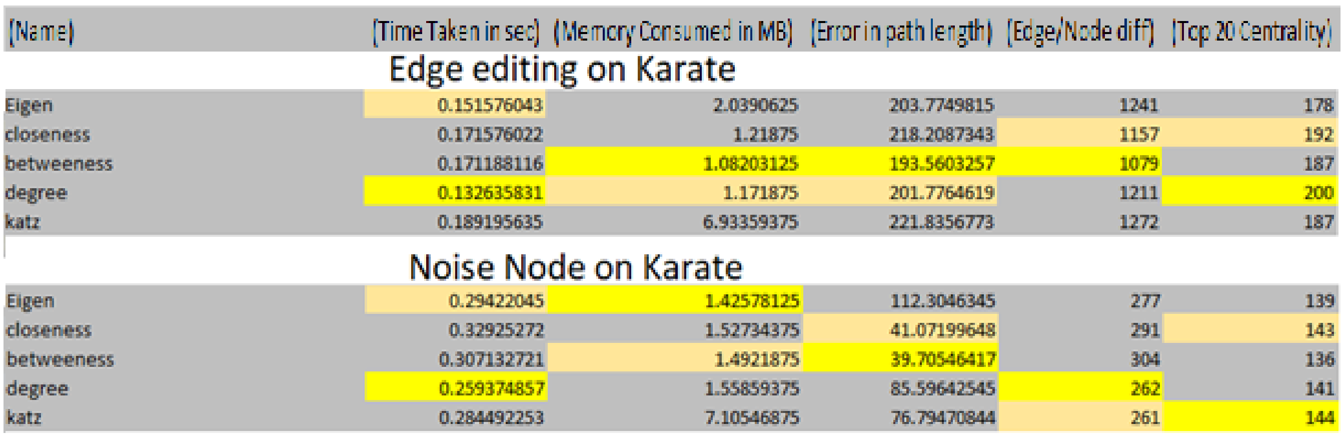
As seen from the graph above, the values for all the values gives slight edge to either one or the other but error in path length is very high for edge editing. Error in path length is a very important characteristic as discussed earlier and edge editing is performing very poor. Also, time starts to increase as the data size is increasing.

Seeing that noise is doing better across 60 tests for three different datasets**, we can conclude that Noise Node addition is better of the two methods.**

## 4.2 Can we say something regarding the rest of the parameters?

### 4.2.1. Centrality values

On an average degree centrality was performing the best followed by eigenvector centrality and betweenness centrality. But these were across the first three datasets of the range 10^3. But when we ran these tests over the internet database which was of a much higher order dimension wise, eigenvector centrality was the only one that was able to work properly where for the rest of centrality especially closeness and betweenness the time complexity went haywire.



| (Name) | (Time Taken in sec) | (Memory Consumed in MB) | (Error in path length) | (Edge/Node diff) | (Top 20 Centrality) |
|---|---|---|---|---|---|
| | **Edge editing on Karate** | | | | |
| Eigen | 0.151576043 | 2.0390625 | 203.7749815 | 1241 | 178 |
| closeness | 0.171576022 | 1.21875 | 218.2087343 | 1157 | 192 |
| betweeness | 0.171188116 | 1.08203125 | 193.5603257 | 1079 | 187 |
| degree | 0.132635831 | 1.171875 | 201.7764619 | 1211 | 200 |
| katz | 0.189195635 | 6.93359375 | 221.8356773 | 1272 | 187 |
| | **Noise Node on Karate** | | | | |
| Eigen | 0.29422045 | 1.42578125 | 112.3046345 | 277 | 139 |
| closeness | 0.32925272 | 1.52734375 | 41.07199648 | 291 | 143 |
| betweeness | 0.307132721 | 1.4921875 | 39.70546417 | 304 | 136 |
| degree | 0.259374857 | 1.55859375 | 85.59642545 | 262 | 141 |
| katz | 0.284492253 | 7.10546875 | 76.79470844 | 261 | 144 |

*Fig 7. Examples of different centrality values across different test parameters. Yellow depicts the best and orange second best*

We can conclude that **eigenvector centrality is the most ideal** although for smaller social networks **degree centrality is the best.**

### 4.2.2. Privacy Measures

| (Name) | (Time Taken in sec) | (Memory Consumed in MB) | (Error in path length) | (Edge difference) | (Top 20 Centrality) |
|---|---|---|---|---|---|
| K | 165.0515201 | 121.40625 | 3519.634222 | 1707 | 204 |
| KL | 168.5304992 | 81.484375 | 3574.848123 | 2408 | 187 |
| ALL | 169.9400496 | 82.99609375 | 3052.999809 | 3796 | 174 |

*Fig 8: Example of the privacy measures across 60 tests for noise node addition method*

As seen in *Fig 8* and throughout the tests we conducted, as we **made use of more privacy measures, the performance deteriorates**. This is expected as increasing more privacy measures means you are making your graph more complicated so that it anonymizes better and hence tests results are bound to drop. Hence to choose how many privacy measures does a user want is up to him knowing that more privacy measures mean worse off performance but he can use our tests to decide between the trade-off and choose privacy.

### 4.2.3. Different values of k & l

Analysing data for different values of k & l did not lead us to any significant conclusion. The different test parameters did not yield any discernible patterns.

## 4.3 Whether it is possible to extend the existing theory to two multiple sensitive attributes?

We were able to extend and create our own multiple sensitive attribute. In *Fig 9*. We can see that our graph is now anonymizing a graph with multiple labels.

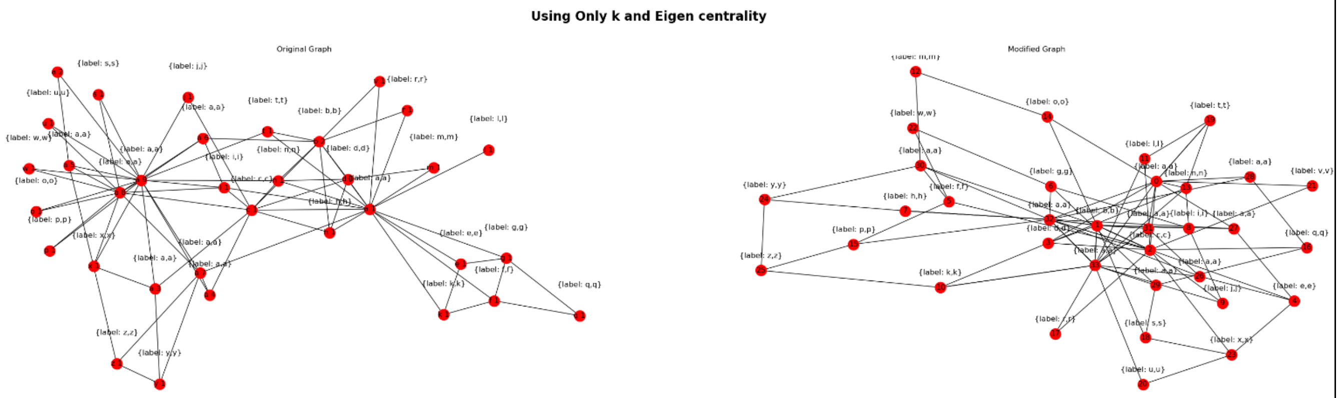
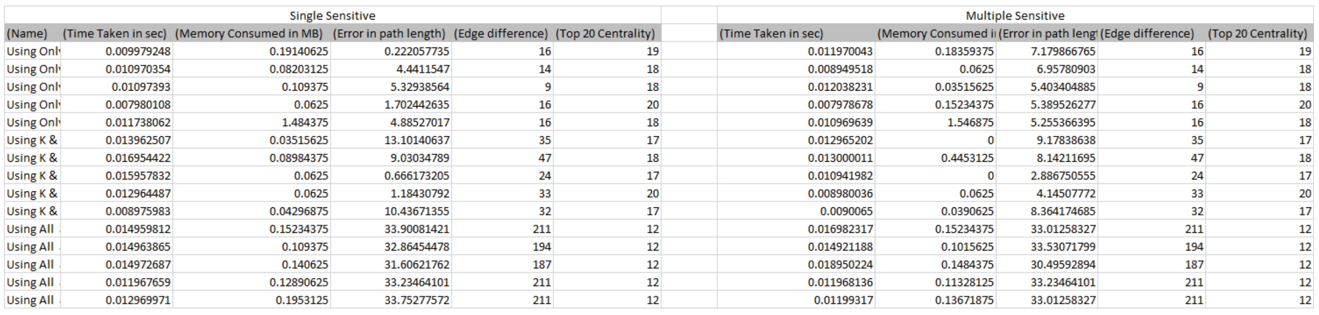


*Fig 9: Edge editing method for multiple labels using only k anonymity privacy measure and eigen centrality. Left original graph, right anonymized graph*

And also, in terms of test parameters, it has worked well as seen in the figure *Fig 10* where we can see that the error in path length is not a lot different. And the rest of the important attributes remain constant.



| | Single Sensitive | | | | | | Multiple Sensitive | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (Name) | (Time Taken in sec) | (Memory Consumed in MB) | (Error in path length) | (Edge difference) | (Top 20 Centrality) | | (Time Taken in sec) | (Memory Consumed i | (Error in path leng | (Edge difference) | (Top 20 Centrality) |
| Using Onl | 0.009979248 | 0.19140625 | 0.222057735 | 16 | 19 | | 0.011970043 | 0.18359375 | 7.179866765 | 16 | 19 |
| Using Onl | 0.010970354 | 0.08203125 | 4.4411547 | 14 | 18 | | 0.008949518 | 0.0625 | 6.95780903 | 14 | 18 |
| Using Onl | 0.01097393 | 0.109375 | 5.32938564 | 9 | 18 | | 0.012038231 | 0.03515625 | 5.403404885 | 9 | 18 |
| Using Onl | 0.007980108 | 0.0625 | 1.702442635 | 16 | 20 | | 0.007978678 | 0.15234375 | 5.389526277 | 16 | 20 |
| Using Onl | 0.011738062 | 1.484375 | 4.88527017 | 16 | 18 | | 0.010969639 | 1.546875 | 5.255366395 | 16 | 18 |
| Using K & | 0.013962507 | 0.03515625 | 13.10140637 | 35 | 17 | | 0.012965202 | 0 | 9.17838638 | 35 | 17 |
| Using K & | 0.016954422 | 0.08984375 | 9.03034789 | 47 | 18 | | 0.013000011 | 0.4453125 | 8.14211695 | 47 | 18 |
| Using K & | 0.015957832 | 0.0625 | 0.666173205 | 24 | 17 | | 0.010941982 | 0 | 2.886750555 | 24 | 17 |
| Using K & | 0.012964487 | 0.0625 | 1.18430792 | 33 | 20 | | 0.008980036 | 0.0625 | 4.14507772 | 33 | 20 |
| Using K & | 0.008975983 | 0.04296875 | 10.43671355 | 32 | 17 | | 0.0090065 | 0.0390625 | 8.364174685 | 32 | 17 |
| Using All | 0.014959812 | 0.15234375 | 33.90081421 | 211 | 12 | | 0.016982317 | 0.15234375 | 33.01258327 | 211 | 12 |
| Using All | 0.014963865 | 0.109375 | 32.86454478 | 194 | 12 | | 0.014921188 | 0.1015625 | 33.53071799 | 194 | 12 |
| Using All | 0.014972687 | 0.140625 | 31.60621762 | 187 | 12 | | 0.018950224 | 0.1484375 | 30.49592894 | 187 | 12 |
| Using All | 0.011967659 | 0.12890625 | 33.23464101 | 211 | 12 | | 0.011968136 | 0.11328125 | 33.23464101 | 211 | 12 |
| Using All | 0.012969971 | 0.1953125 | 33.75277572 | 211 | 12 | | 0.01199317 | 0.13671875 | 33.01258327 | 211 | 12 |

*Fig 10. Comparison between single sensitive attribute and multiple sensitive attribute.*

***<u>Hence because of this, we can say yes, it is possible for the existing methods to extend for multiple sensitive attributes.</u>***

# 5 CONTRIBUTION

## 5.1 Dhruv

- Analysis, Design and implementation of test methods.
- Designed the approach and develop code to capture various performance measurement data
- Pipelining and modularization of code (Including Bug Finding).

## 5.2 Srujan

- Implemented k-degree anonymity, l-diversity and recursive(c,l) diversity privacy measures.
- Wrote code for the edge editing method and the noise node addition method.
- Implemented the various centrality measures and created the target degree sequence generation module.

## 5.3 Ishan

- Wrote code for the implementation of privacy models on synthetic datasets.
- Wrote code for visualization of graphs.
- Debugging of code.

# 6 WHAT WE HAVE LEARNED

## 6.1 Dhruv

Till now I had only worked on research projects and associated code development. Working on this project is my first experience of building a solution from scratch. I looked into various software engineering accepts of the project and applied various techniques in this project. I particularly liked writing the test cases which helped us evaluate all the techniques by providing an in-depth analysis consisting of time-taken, memory consumed, error in path length as well as the edge difference between the original and the anonymized graph. While trying to find different ways to test our code, I came across quite interesting testing methods and literature about topics like unit testing, integration testing, etc. I also learned about engineering practices to make our code modularized and established proper pipeline/flow. Before this project, I had no prior knowledge of anonymization so I learned a lot about this interesting concept. Also, I was misinformed in believing that network security equates to only cryptography but now I have learned even through network relations, people can decode data.

### 6.2 Ishan

This project was a huge learning experience for me. Having previously done research on social networks (focused mainly on data mining from social networks data), this project gave me the opportunity to explore a previously unfamiliar topic in this field: anonymization of social networks. Prior to this project, I had little knowledge about the complexities involved in maintaining this privacy through anonymization of graphs. I had a false idea that just the hiding of nodes was adequate to protect the privacy of a graph. However, I was surprised to learn about the vast number of ways to exploit the privacy of a graph by leveraging on the properties of a graph structure. I thoroughly enjoyed learning about the 2 kinds of privacy models- Edge-editing, and Noise node addition. I found all 2 of them quite intuitive to understand.

### 6.3 Srujan

I was able to empirically estimate the effects of privacy measures on the structural integrity of social networks. This analysis gave me a great insight into the trade-off between privacy and the usefulness of anonymised information. This project also gave me the opportunity to explore various properties of social networks such as various centrality measures (and their unique interpretations), average path length, clustering co-efficient etc. I can now better appreciate the nuances associated with anonymising sensitive information. Furthermore, these properties that I gained knowledge about also hold good in tabular data (with certain changes). Overall, this project introduced me to the world of social network analysis and the challenges therein, and certainly enhanced my interest in this field.

## 7 FUTURE SCOPE

We used the python networkx open-source library extensively. Networkx provides a multitude of features for the manipulation and study of graphs. However, as of now, it doesn't provide any method to related to the privacy and anonymization graphs.

We have started communication and we hope to commit our code to the networkx open-source library in Github to provide method for the privacy measures and noise node and edge editing method.

Moreover, new privacy measures can be incorporated. We can also take a look at the state-of-the-art privacy measures in the field of databases and try to extend them to social network analysis.