PRIVACY PRESERVING IN ONLINE SOCIAL NETWORK DATA SHARING AND PUBLICATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Tianchong Gao

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

ACKNOWLEDGMENTS

Four years have passed since my first visit to Indiana. During the four years, happiness is accompanied by sadness. However, at this moment, the end moment, all the joy and pain are not so important any more. The only remaining thing is my gratitude. My gratitude to the four year's time period. My gratitude to many many people around me.

First I would like to raise special appreciation and thanks to my advisor, Dr. Feng Li, for his support of my studies. I am grateful for you:

- showing me the beauty of research;
- guiding me when I was wondering;
- discussing our interesting ideas together;
- taking care of my study, my work, and my life.

I am grateful for Dr. Xiaojun Lin, Dr. Stanley Yung-Ping Chien, Dr. Jianghai Hu, and Dr. Edward J. Delp, who kindly serve on my advisory committee. Dr. Lin helped a lot to manage my Ph.D. program. All my committee members show their patience, kindness, helpfulness, and professionalism during the four years. I would also thank Dr. Avi Kak for his guidance in my first year study. Thank you.

Research is somewhat a lonely journey, but life isn't. New Ph.D. students, Yuchen Xie and Taotao Jing, brought happiness to our lab. Sherrie and Matt helped me tons of times when I face troubles with the complicated processes. Professors in the department, Dr. Liu, Dr. Luo, Dr. Guo, and Dr. Ding, gave me a lot suggestions on both study and life.

Last but not least, my family is what sustained me thus far. Thanks to the love and encourage of my beloved parents Jin Gao and Jihong Xu. I would not keep it up without you. I would also like to express my deepest thanks to my aunt, Lin Gao.

When I first came to the US, she drove hours from Canada to Michigan and brought numerous life necessities for me. To my aunt: Be happy in heaven; your nephew becomes a man; everything will be good.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

G	graph
G^n	graph at epoch n
D	database
E	set of edges
V	set of nodes
A	set of attributes
K_n	set of simplicial complex in n-th dimension
$\mathcal{T}$	hierarchical random graph tree
$\mathcal{A}$	algorithm
$\|.\|$	absolute value, cardinality of the set
$\|\|.\|\|$	magnitude of the vector
$\langle . \rangle$	dK series
$\emptyset$	empty set
u-z	node
$e_{u,v}$	edge between node u and node v
p	probability
d_u	degree of node u
L, R	left, right subtree
M_a	adjacency matrix
M_d	distance matrix
S_p	topology similarity
S_a	attribute similarity
σ	simplicial complex
H_n	persistent homology barcode in n-th dimension

Δ	sensivity
ϵ	differential privacy parameter
δ	distance parameter
α, β	scaling parameter
$\vee$	wedge dK-3 series
$\bigtriangledown$	triangle dK-3 series
Lap	Laplace distribution
e, exp	exponential distribution
OS	output space
C	confidence
err	error of dK series

ABBREVIATIONS

OSN	online social network
HRG	hierarchical random graph
PHDP	persistent homology and differential privacy
ADS	all-distance sketch
NDL	neighbor degree list
GAN	generative adversarial network
CGAN	conditional generative adversarial network
GAE	graph auto-encoder
CDN	content delivery network

ABSTRACT

Gao, Tianchong. Ph.D., Purdue University, December 2019. Privacy Preserving in Online Social Network Data Sharing and Publication. Major Professors: Stanley Yung-Ping Chien and Xiaojun Lin.

Following the trend of online data sharing and publishing, researchers raise their concerns about the privacy problem. Online social networks (OSNs), for example, often contain sensitive information about individuals. Therefore, anonymizing network data before releasing it becomes an important issue. This dissertation studies the privacy preservation problem from the perspectives of both attackers and defenders.

To defenders, preserving the private information while keeping the utility of the published OSN is essential in data anonymization. At one extreme, the final data equals the original one, which contains all the useful information but has no privacy protection. At the other extreme, the final data is random, which has the best privacy protection but is useless to the third parties. Hence, the defenders aim to explore multiple potential methods to strike a desirable tradeoff between privacy and utility in the published data. This dissertation draws on the very fundamental problem, the definition of utility and privacy. It draws on the design of the privacy criterion, the graph abstraction model, the utility method, and the anonymization method to further address the balance between utility and privacy.

To attackers, extracting meaningful information from the collected data is essential in data de-anonymization. De-anonymization mechanisms utilize the similarities between attackers' prior knowledge and published data to catch the targets. This dissertation focuses on the problems that the published data is periodic, anonymized, and does not cover the target persons. There are two thrusts in studying the de-anonymization attacks: the design of seed mapping method and the innovation of generating-based attack method. To conclude, this dissertation studies the online data privacy problem from both defenders' and attackers' point of view and introduces privacy and utility enhancement mechanisms in different novel angles.

1. INTRODUCTION

Online Social Networks (OSNs) have exploded in popularity. The OSN providers, like Facebook and Twitter, own a vast amount of personal data and relationship information between their users. OSN service providers always have incentives to share data with third parties. Service providers publish the data for new friendship recommendations, targeted advertisement feeding, application evaluation, human social relationships analysis, etc.

However, leaking private information, e.g., users' interests, users' profiles, and the linking relationships between users, can cause great panic to OSN users and service providers. Cambridge Analytica gained access to approximately 87 million Facebook accounts [149]. Following the data scandal, Facebook apologized amid public outcry and fallen stock prices in 2018.

This dissertation mainly focuses on privacy preservation problems in OSN data sharing. Intuitively, the OSN data is modeled by a graph, where the nodes show the users and the edges show the relationships. Previously, researchers demonstrated that naive ID removal, which simply removes users' identities, was also vulnerable [76, 106]. Attackers can utilize the unchanged structural information to apply a de-anonymization attack. Hence, various anonymization techniques have been proposed to preserve privacy. These techniques only mask the identities but also perturb the graph structures. They include the k-anonymity based methods, i.e., making at least k users similar to each other, and differential privacy based methods, i.e., limiting the private information leakage.

While existing OSN anonymization schemes, especially differential privacy-based ones, are rich in preserving privacy, the regenerated graph lacks enough utility, which is the usefulness to the benign third parties for network analysis. Generally, this dissertation studies the following problems that may exist in data anonymization:

the angle to balance utility and privacy, the measurement of utility and privacy, and the unnecessary utility and privacy loss. Specifically, the main challenges of the anonymization schemes are:

1. Network data is susceptible to the changes in the graph structure. Although the global differential privacy techniques have a strict privacy guarantee, noise in the published graph affects the utility of the data.
2. In differential privacy-based schemes, abstraction models are employed to transform network data into numerical type. However, deploying one abstraction model can only capture some aspects of information, while the published graph loses the information in other aspects.
3. Existing differential-privacy schemes claim to preserve graph utility under certain graph metrics. However, each graph utility metric reveals the whole graph in specific aspects.
4. When the privacy level of the published graph is adjustable, the utility preservation of existing schemes is out of control.

Rising to these challenges, we propose several new angles to strike a smart balance between privacy and utility. For example, when setting the privacy level, we give the notion of local differential privacy when global differential privacy requires too much noise. When studying the graph abstraction models, we design a comprehensive model to combine existing models. When choosing utility metrics, we introduce a novel metric to measure graph utility. When designing the anonymization scheme, we choose a novel route which can adjust the utility level.

Besides OSN anonymization, OSN de-anonymization also has privacy issues but from the attacker's perspectives. De-anonymization helps the researchers to find weak points in anonymization design and provides valuable insights to OSN privacy preservation. Existing de-anonymization mechanisms mainly apply a mapping attack between adversary's background knowledge and the published data. After successfully mapping the unidentified users, adversaries gather information from the published data. The main challenges of the de-anonymization schemes are:

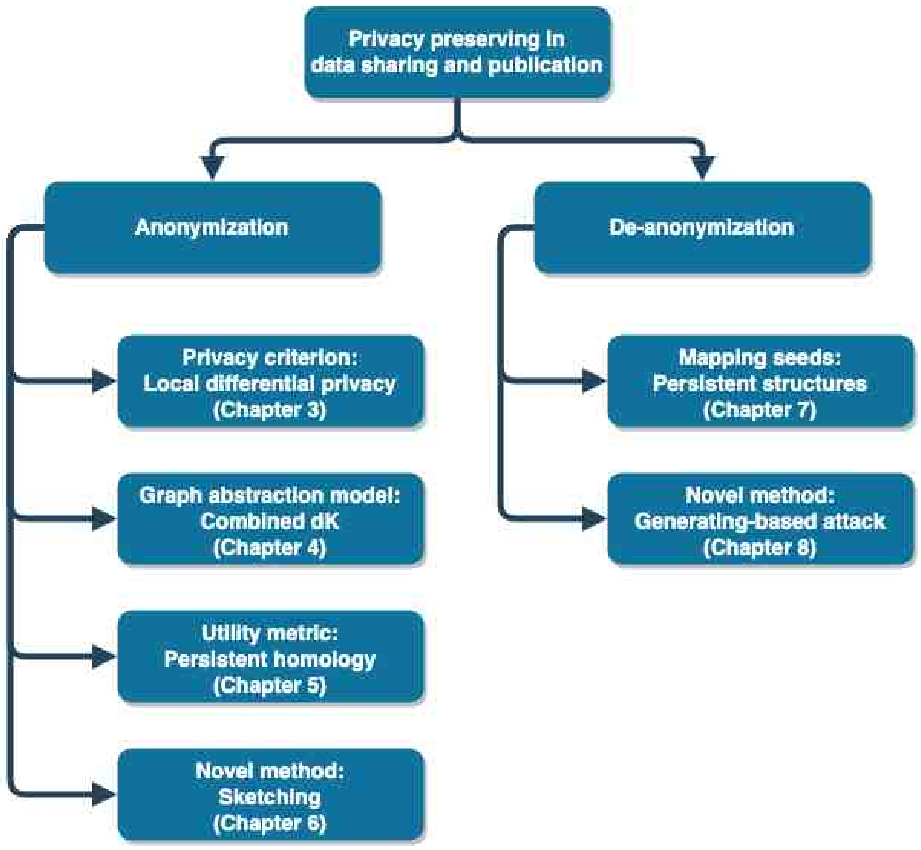

Privacy preserving in data sharing and publication
Anonymization
De-anonymization
Privacy criterion: Local differential privacy (Chapter 3)
Graph abstraction model: Combined dK (Chapter 4)
Utility metric: Persistent homology (Chapter 5)
Novel method: Sketching (Chapter 6)
Mapping seeds: Persistent structures (Chapter 7)
Novel method: Generating-based attack (Chapter 8)

Fig. 1.1.: Visual depiction of the dissertation organization.

1. Existing schemes do not take advantage of periodically published data. Most of them can only handle the static data or cut dynamic data into pieces of static data.
2. Based on existing de-anonymization schemes, attackers can hardly learn information about targets if published data is not related to these users. Existing mapping attack requires that adversary's background knowledge and published data involve the same group of users.

Rising to these challenges, we propose several designs to help attackers capture meaningful information from the published data. We introduce persistent structures to model the part in the dynamic OSN data. We use the generative adversarial network, a deep learning model, to apply a generating-based attack.

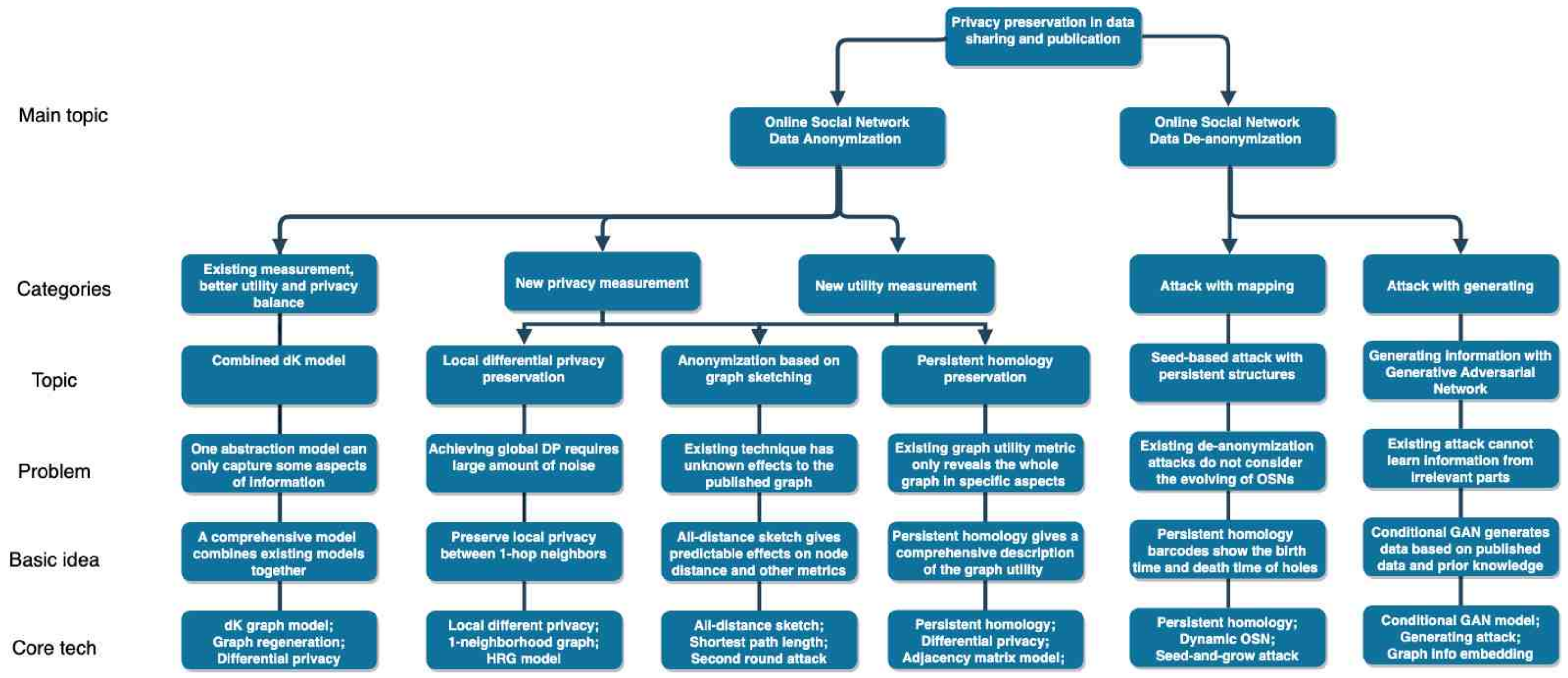

Main topic
Categories
Topic
Problem
Basic idea
Core tech
Privacy preservation in data sharing and publication
Online Social Network Data Anonymization
Online Social Network Data De-anonymization
Existing measurement, better utility and privacy balance
New privacy measurement
New utility measurement
Attack with mapping
Attack with generating
Combined dK model
Local differential privacy preservation
Anonymization based on graph sketching
Persistent homology preservation
Seed-based attack with persistent structures
Generating information with Generative Adversarial Network
One abstraction model can only capture some aspects of information
Achieving global DP requires large amount of noise
Existing technique has unknown effects to the published graph
Existing graph utility metric only reveals the whole graph in specific aspects
Existing de-anonymization attacks do not consider the evolving of OSNs
Existing attack cannot learn information from irrelevant parts
A comprehensive model combines existing models together
Preserve local privacy between 1-hop neighbors
All-distance sketch gives predictable effects on node distance and other metrics
Persistent homology gives a comprehensive description of the graph utility
Persistent homology barcodes show the birth time and death time of holes
Conditional GAN generates data based on published data and prior knowledge
dK graph model; Graph regeneration; Differential privacy
Local different privacy; 1-neighborhood graph; HRG model
All-distance sketch; Shortest path length; Second round attack
Persistent homology; Differential privacy; Adjacency matrix model;
Persistent homology; Dynamic OSN; Seed-and-grow attack
Conditional GAN model; Generating attack; Graph info embedding

Fig. 1.2.: Overview

Figure 1.1 shows the overall organization of this dissertation. This dissertation studies the privacy preservation problem in both anonymization and de-anonymization aspects. Several drawbacks in existing schemes, e.g., the privacy criterion of anonymization and the seed mapping algorithm in de-anonymization, are analyzed. This dissertation aims to design new schemes and improve existing schemes to avoid drawbacks. Figure 1.2 gives a detailed technical taxonomy. This dissertation contains the following chapters:

1. Chapter 2, "Related work" introduces the related researches in online social network anonymization and de-anonymization.
2. Chapter 3, "Anonymization with Privacy Criterion - Local Differential Privacy" gives the novel notion of group-based local differential privacy for achieving higher utility when the privacy level is the same as global differential privacy. Because hiding one node in the whole graph requires a large amount of noise, our main idea is to hide each node in a small subgraph and hide these subgraphs in groups.
3. Chapter 4, "Anonymization with Graph Abstraction Model - Combined dK" gives a comprehensive model combining dK-1, dK-2, and dK-3. Because existing graph abstraction models only extract some aspects of information from the graph data, our main idea is to use the dK-1 and dK-2 models, which are easy to reconstruct the graph, together with the dK-3 model, which contains more information.
4. Chapter 5, "Anonymization with Utility Metric - Persistent Homology" preserves persistent structures and differential privacy at the same time. Because existing utility metrics cannot reveal the whole graph in different dimensions, our main idea is to introduce the novel utility metric called persistent homology and preserve this information in differential-private graphs.
5. Chapter 6, "Anonymization with Novel Method - Sketching" proposes a novel route to anonymize graphs based on sketching. Because existing anonymization mechanisms cannot adjust the utility level, our main idea is to introduce a new

anonymization mechanism based on distance preserving sketch. In the published graph, both the utility, i.e., the distance information, and the privacy, i.e., the released information, is adjustable.

6. Chapter 7, "De-anonymization with Mapping Seeds - Persistent Structures" employs persistent homology to de-anonymize OSN users. Because existing de-anonymization schemes cannot take advantage of the OSN evolution information, our main idea is to use persistent homology to extract the holes in different OSN epochs and map these holes.
7. Chapter 8, "De-anonymization with Novel Method - Generating-based Attack" employs the conditional generative adversarial network model to generate information for the attackers. Because existing de-anonymization attacks cannot utilize information not related to target users, our main idea is to apply the deep learning model to inject this information into attackers' results.

1.1 Anonymization with Privacy Criterion - Local Differential Privacy

In Chapter 3, our anonymization scheme is based on the Hierarchical Random Graph (HRG) model [28]. The HRG model is a rooted binary tree with $|V|$ leaf nodes corresponding to $|V|$ vertices in the graph G. Each non-leaf node on the tree has a number on it that shows the probability of connection between its left part and right part. Xiao et al. applied this HRG model to achieve global ϵ-differential privacy over the entire dataset [153]. However, network data is sensitive to changes in the network structure. Although these global differential-privacy techniques are rich in preserving privacy, the regenerated graph lacks enough utility for network analysis.

The challenge in OSN anonymization is to find the genuine privacy demands and avoid adding unnecessary noise which damages utility. Analyzing the de-anonymization attack process can give us better guidance in designing anonymization schemes. Existing de-anonymization algorithms compute the structural similarities and attribute similarities of nodes. Some of these algorithms choose a group of nodes as mapping candidates of the target node [90, 120]. Some other algorithms group nodes into clusters and then do subgraph matching [27, 106]. These de-anonymization algorithms imply that anonymization does not need to hide one node with all other nodes. Moreover, the subgraph is an essential component in de-anonymization that we need to make subgraphs similar to each other.

In this chapter, our first step towards achieving such balance is to split the whole graph into multiple subgraphs. Graph segmentation has two main advantages: First, it helps to reduce the noise scale of differential privacy. The notion of local differential privacy preserves more graph utility than global differential privacy under the same privacy parameter ϵ. Second, it also helps to reduce the HRG output space size. Therefore, each HRG has higher posterior probability, and regenerating a perturbed graph from it loses less information. The subgraph model we use is the 1-neighborhood graph, which contains a central node and its 1-hop neighborhoods.

After separating the whole graph into subgraphs, the HRG model is deployed to extract the features with a differential-privacy approach. We introduce a grouping algorithm based on the similarity of HRG models to enhance anonymization power. Specifically, the HRGs with the overlap in their output space are grouped to form a representative HRG. We use this representative HRG to smooth other subgraphs inside the group. Since all sanitized subgraphs in a group are regenerated from one HRG, the adversary is not able to differentiate the target even with the help of prior knowledge.

Finally, we design the graph regeneration process. In order to replace the original 1-neighborhood graph with the perturbed one, the number of nodes in the new subgraph should not be fewer than that of the original graph. However, grouping makes it possible to merge subgraphs of different sizes. Generating the representative HRG from the largest subgraph will add many dummy nodes. Hence, we introduce two methods called ‘virtual node’ and ‘outlier distinction’ to solve this problem. Generally, the two methods avoid adding too many nodes when satisfying the grouping criteria, which balances privacy with graph utility.

1.2 Anonymization with Graph Abstraction Model - Combined dK

In Chapter 4, our anonymization scheme is based on the dK graph model Mahadevan et al.. The dK model is separated into different dimensions. The dK-N model captures the degree distribution of connected components of size N. For example, dK-1, also known as the node degree distribution, counts the number of nodes in each degree value. The dK-2 model, also called joint degree distribution, captures the number of edges in each combination of two-degree values. Sala et al. employed the dK-2 series as the graph abstraction model to achieve differential privacy [129]. However, deploying one abstraction model can only capture some aspects of information, while other utilities are lost in the published graph. For example, because the dK-2 graph model is the record of edges, it may not preserve information involving more than two nodes, e.g., the clustering coefficient.

Hence, choosing an abstraction model becomes an important issue. Mahadevan et al. proved that dK models in higher dimensions have more information than the ones in lower dimensions, e.g., the dK-3 model is more precise than the dK-2 model [95]. Our initial idea is to preserve differential privacy on the dK-3 model. In our study, we find that it is hard to reconstruct the graph with only the dK-3 series. After studying the different properties between the dK-1, dK-2, and dK-3 series. We find that low dimensional models, e.g., dK-1, are less sensitive to noise, and can efficiently regenerate a graph. High dimensional models can preserve more structural information.

In this chapter, we absorb the benefits of different models and design a new comprehensive model that combines three levels of dK graph models. To achieve differential privacy, we introduce noise on the dK-2 level, which causes less distortion than on the dK-3 level. Then we use the perturbed dK-2 series to get the corresponding dK-3 and dK-1 series. After that, we use three levels of dK abstractions together in our scheme to construct a new graph.

The noise impact is the major challenge in the graph regeneration process. Although the three models in our scheme are closely related, they may conflict with each other because of noise. Hence, we first use some dK information to regenerate an intermediate graph, then use the remaining information to rewire the edges. In particular, we propose two sub-schemes, called *consider all together* (CAT) and *low to high* (LTH), with different executing sequences in the dK series.

After getting the target dK series, the general purpose of graph regeneration is to minimize the error between it and the published graph in all three levels. In the rewiring part, we develop three dK rewiring algorithms to reduce the errors graphically. The rewiring algorithms also help us inject the remaining dK information to the graph. The algorithms analyze the differences to find potential rewiring pairs. Because one level of rewiring may have negative impacts on other dK levels, both intermediate graphs apply the rewiring from lower to higher except that the LTH graph needs no dK-1 rewiring.

1.3 Anonymization with Utility Metric - Persistent Homology

In Chapter 5, our anonymization scheme is based on persistent homology [58]. Persistent homology tracks the topological features of the whole graph at different distance resolutions in different dimensions. In OSNs, each persistent homology barcode is an interval showing a component or a hole in the corresponding dimension. The intervals begin with the distances the holes born; end with the distances the holes die. For example, the square structure in OSN is an H_1 bar $[1, 2)$ in persistent homology barcode.

Although existing anonymization schemes, e.g., dK-2 based one and HRG-based one, claim to preserve graph utility under some specified utility metrics, the actural utility of the published graphs is questionable for two reasons: First, the chosen metrics are limited by the graph abstraction models. Previous studies have shown that none of the schemes have energetic performance under all the metrics [47]. Second, existing metrics only describe the graph at a certain angle. For example, while the degree distribution and the clustering coefficient disjointedly reveal graph utility in two specific aspects, each aspect does not cover the other. Thus, lots of useful graph information gets lost or distorted during the graph anonymization process, primarily when the anonymization schemes are based on these types of graph metrics.

In this chapter, persistent homology is employed to analyze graph utility. Unlike the well-studied utility metrics, persistent homology gives a comprehensive summarization of the graph. Since persistent homology is a novel utility metric, the main challenge of our anonymization scheme is to extract the corresponding persistent homology information and preserve it in the published graph.

First, our scheme model the OSN by an adjacency matrix for two reasons: (1), the adjacency matrix contains the same topological information as the distance matrix. Because the persistent homology filtering phase tracks the persistent structures with different distances, the structures in the distance matrix can be easily mapped to the ones in an adjacency matrix. (2), the adjacency matrix has less sensitivity in edge adding or deleting than other graph abstraction models, i.e., it requires less noise under the same privacy level.

Second, to preserve the persistent homology in OSNs, we analyze the structural meaning of barcodes. We find that the OSN graph has the possibility of folding, which is different from existing studies of point cloud data [13, 116]. Initially, persistent homology defines H_1 bars as circular holes and H_2 bars as voids. However, folding complicates the analysis of high-dimensional holes but also opens the opportunity to extract the actual shapes of the persistent structures in OSNs. Particularly, high-dimensional voids are folded into unique kinds of holes. Therefore, preserving the polygons defined by the barcodes is preserving persistent homology.

Third, we design an anonymization algorithm that preserves the holes and satisfies differential privacy. The holes occupy a small part of the network; differential privacy is maintained through modifying the other parts. Notably, we divide the adjacency matrix into four kinds of sub-matrices, according to the corresponding subgraphs with or without holes. Then different regeneration algorithms are employed to each kind of matrix to satisfy differential privacy and preserve the holes at the same time.

1.4 Anonymization with Novel Method - Sketching

In Chapter 6, we embed All-Distance Sketch (ADS) in our OSN anonymization mechanism. ADS has two advantages:

First, ADS accurately preserves some structural information, e.g., distances, neighbors, and betweenness, with bounded error. Several OSN data applications, including analyzing the information transmission speed and building the rumor spreading model, have specific demands of the accurate information in the published graph. Thus, the ADS graph is appropriate to preserve the data.

Second, ADS eliminates insignificant edges, e.g., edges not on shortest paths and parallel edges between clusters, from the original graph. After edges removal, the adversary will have high uncertainty whether the original graph has some specific edges or not. Most de-anonymization attacks are seed-based [27, 119]. They use special attributes, e.g., high degree and profile similarity, to build mapping seeds and then extend the mapping attack [145]. Other de-anonymization attacks are often based on subgraph isomorphism [7, 128]. Since ADS graph dramatically changes the network structure, it is capable of defending against these attacks.

However, ADS is not designed for private data sharing. When the adversaries are intelligent, directly sharing ADS graph leaves two main challenges in preserving privacy:

First, because the ADS scheme does not add any edge to the published graph, the adversary knows that every edges in the ADS graph must be in the original graph. Hence, the performance of this anonymization scheme decreases when there is no false positive in adversaries' intelligent guesses on the links. In order to overcome this shortfall, we design an edge addition and deletion algorithm, in addition to the ADS.

Our analysis demonstrates that both the privacy and utility of our published graph are related to the total number of edges added/deleted. Hence, we can modify the tradeoff between privacy and utility with edge addition/deletion.

Second, even if the anonymization mechanism naively adds dummy edges, real edges have higher importance than dummy edges. Compared with dummy edges, real edges are more likely to be the edges along the shortest paths, which are the backbones in the network. Therefore, an intelligent attack strategy is to generate the ADS sample of the ADS graph. Edges in the ADS of ADS graph have a high probability of being contained in the original graph. To tackle this problem, we design the bottom-(l, k) sketch scheme based on the original bottom-k sketch. While bottom-k requires k nodes with the lowest ranks, bottom-(l, k) requires each node has at least l different paths to the source node. The newly added paths make it more challenging to find the real paths and enhance the privacy of the published data. Moreover, we design a new ADS graph generation process that achieves bottom-(l, k) sketch.

1.5 De-anonymization with Mapping Seeds - Persistent Structures

In Chapter 7, our de-anonymization scheme is also based on persistent homology. However, we apply persistent homology to dynamic OSNs. Persistent homology in this chapter tracks the topological features of the dynamic graph at different time resolutions in different dimensions. The barcode intervals begin with the time the holes born; end with the time the holes die. For example, the square structure exists from epoch 1 to epoch 2 in dynamic OSN is an H_1 bar $[1, 2)$ in persistent homology barcode.

Although existing de-anonymization attacks mainly focus on the static graphs of OSN data, OSNs are time-variant [90, 120]. Researchers also designed de-anonymization attacks on dynamic OSNs. Some schemes use the same methods that are used to de-anonymization attacks upon static data. Here, a time-series graph is considered as a combination of pieces of graphs [41]. Hence, the method to de-anonymize dynamic graphs is mere to sequentially de-anonymize static graphs. These schemes cannot use the time to conduce de-anonymization. Therefore, the de-anonymization attacks upon dynamic OSN data may face the same problems that are faced when trying to de-anonymize static OSN data.

In this chapter, we use persistent homology to give a multi-scale description of the time-series graphs. In particular, persistent homology filters persistent structures over time. Persistent homology barcodes show the birth time and death time of the holes. We examine the similarities between holes in two time-series graphs, instead of individually considering the similarities between nodes in each piece of the graph. If two holes match with each other, we use the nodes on the holes as seeds to further grow the node mapping, until two time-series graph are mapped.

1.6 De-anonymization with Novel Method - Generating-based Attack

In Chapter 8, we introduce the idea of a generating-based de-anonymization attack to replace existing mapping-based attacks. Specifically, we apply a deep neural network model called Generative Adversarial Network (GAN) to absorb the high-dimensional structure information and generate a new network to enhance the attacker's background knowledge. GAN designs a game theory scenario between the generator and the discriminator. In this game, the generator strives to generate fake examples similar to real examples, while the discriminator strives to discriminate between fake examples and real examples. After the game gets coverage, the generator can generate fake examples that are indistinguishable with the discriminator.

This chapter is based on the assumption that different parts of the OSN should have similar structural properties, e.g., degree distribution, clustering coefficient, and some high-dimensional properties. In real-world cases, OSN service providers or third parties sometimes directly publish a subgraph of the original OSN, but the target persons may not be in the published graph. Hence, we would like to deploy GAN to generate a subgraph that contains the target persons and is similar to the published graph. Finally, the newly generated edges may enhance the adversary's background knowledge, i.e., telling friendship information about the targets.

Although it is innovative to apply the GAN model to the graph domain, there leave three main challenges:

1. How to embed the adversary's background knowledge into our GAN model? The adversary always has some knowledge (albeit incomplete) about target users. This knowledge is the basic information in both a traditional mapping-based scheme and our generating-based scheme. In this chapter, we first apply Graph Auto-Encoder (GAE) to project the graph information into the feature domain. Then, we deploy the Conditional-GAN (CGAN) model to inject this information as conditional labels.
2. How to embed published data into our GAN model? The purpose of GAN is to generate a graph having properties similar to the published graph, but not exactly the same as any part of the graph in the published data. In this chapter, we apply the mini-batch method to defend against the model-collapse problem.
3. How to design the deep neural network architecture in both the generator and the discriminator? In order to collect the information of graph structure and attributes, we choose a specific classifier model, Graph Neural Network (GNN), in our GAN.

1.7 Contribution

In conclusion, this dissertation studies the OSN data privacy preservation problem. The major technical contributions can also be divided into the anonymization aspect and de-anonymization aspect.

To anonymization, this dissertation designs four novel anonymization schemes for OSN service providers to protect data privacy. Comparing with existing anonymization schemes, the proposed schemes achieve a different balance between privacy and utility from the following angles: privacy criterion, graph abstraction model, utility metric, and impact on utility. The proposed schemes are evaluated on the real-world OSN dataset. The evaluation results show that the proposed schemes preserve more graph utility when the data privacy levels are similar to the existing anonymization schemes.

To de-anonymization, this dissertation designs two novel de-anonymization schemes for the attackers to find private information. The proposed schemes focus on the scenarios that the OSN service providers periodically publish data, and the published data does not contain the targets. The experiments on real-world datasets demonstrate that the proposed schemes have better de-anonymization accuracy than existing schemes.

Publications related to this dissertation is listed as follows:

- **Gao, Tianchong**, and Feng Li, "Privacy-Preserving Sketching for Online Social Network Data Publication," *Proceedings of the 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).* IEEE, 2019.

- **Gao, Tianchong**, and Feng Li, "De-anonymization of Dynamic Online Social Networks via Persistent Structures," *Proceedings of the 2019 IEEE International Conference on Communications (ICC)*, pp.1-6, May 2019, Shanghai, China.

- **Gao, Tianchong**, and Feng Li, "PHDP: Preserving Persistent Homology in Differentially Private Graph Publications," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pp.1-9, April 2019, Paris, France.

- **Gao, Tianchong**, and Feng Li, "Sharing Social Networks Using a Novel Differentially Private Graph Model," *Proceedings of the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp.1-4, Jan 2019, Shanghai, China.

- **Gao, Tianchong**, Feng Li, Yu Chen, and XuKai Zou, "Local Differential Privately Anonymizing Online Social Networks Under HRG-Based Model," *IEEE Transactions on Computational Social Systems*, vol. 5, num. 4, pp. 1009-1020, IEEE, 2018.

- **Gao, Tianchong**, and Feng Li, "Studying the utility preservation in social network anonymization via persistent homology," *Computers & Security*, vol. 77, num. 1, pp. 49-64, Elsevier, 2018.

- **Gao, Tianchong**, Wei Peng, Devkishen Sisodia, Tanay Kumar Saha, Feng Li, and Mohammad Al Hasan, "Android Malware Detection via Graphlet Sampling," *IEEE Transactions on Mobile Computing*, vol. 1, num. 1, pp. 1-15, IEEE, 2018.

- **Gao, Tianchong**, and Feng Li. "Preserving Graph Utility in Anonymized Social Networks? A Study on the Persistent Homology." *Proceedings of the 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 348-352. Jan 2017, Silicon Valley, CA.

- **Gao, Tianchong**, Feng Li, Yu Chen, and XuKai Zou, "Preserving local differential privacy in online social networks," *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications*, pp. 393-405, Jun 2017, Gulin, China.

- Peng, Wei, **Tianchong Gao**, Devkishen Sisodia, Tanay Kumar Saha, Feng Li, and Mohammad Al Hasan, "ACTS: Extracting android App topological signature through graphlet sampling." *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS)*, pp. 37-45, Oct 2016, Philadelphia, PA.

2. RELATED WORK

2.1 Online social network anonymization

This dissertation aims to preserve the private information in online data sharing and publication. The main topic of the dissertation focuses on publishing Online Social Networks (OSNs) while preserving individual's security and keeping information of the network. To preserve privacy, removing the identity of each user is a straight forward procedure before sharing the data [106]. To the adversaries, they hardly take advantage of the released data when they cannot link the attributes/profiles with the owners. To the third parties, removing the identities has little impact to the statistics of the data. Naive ID removal gained widely commercial usage because of its simplicity [76]. However, naive ID removal is vulnerable to inference attacks, which means the adversaries infer the true identity with their background knowledge [100]. When the OSN data is defined as a graph, naive ID remove does not perturb the structure of the graph. The released data suffered from structure information de-anonymization attacks [111, 142].

Hence, existing OSN data anonymization techniques not only removed the identities and modified the profiles, but also perturb the graph structures. Several privacy criteria from database privacy preservation were introduced to provide guidance on OSN anonymization. Two famous criteria are called k-anonymity and differential privacy. k-anonymity requires that there are at least k elements in each category, then it is hard for the attacker to differentiate these k elements in the inference attack [138]. k-anonmity has many privacy-preservation applications. For example, k-anonymity was embedded in the credit incentive system, or the query answering system to preserve location data privacy [91, 144].

In OSN anonymization, researchers defined several graph structural semantics, e.g., a cluster, a clique, and a node-hierarchy, as the categories to achieve k-anonymity [26, 131, 166]. These researchers designed their structure perturbation algorithms to get graph automorphism or isomorphism with the minimum modification to original graphs. Unfortunately, most of these k-anonymity techniques have strict limitation on adversarial background knowledge. After choosing the specific structure semantics, k-anonymity may be overcome by other structure semantics [76].

Differential privacy is another kind of privacy preservation criterion [**?**]. It is designed to protect the privacy between neighboring databases that differ by only one element [42]. It means that the adversary cannot determine whether one of the elements changed based on the releasing result. In our model of OSNs, the adversary is not able to tell whether or not two users are linked in the original network.

Definition 1 (NEIGHBOR DATABASE). *Given a database D_1, its neighbor database D_2 differs from D_1 in at most one element.*

In our research, the neighbor database/graph refers to an OSN with one edge added or deleted.

Definition 2 (SENSITIVITY). *The sensitivity ($\triangle f$) of a function f is the maximum distance of any two neighbor databases in the ℓ_1 norm.*

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\| \tag{2.1}$$

Definition 3 (ϵ-DIFFERENTIAL PRIVACY). *A randomized algorithm $\mathcal{A}$ achieves ϵ-differential privacy if for all neighbor datasets D_1 and D_2 and all $S \subseteq Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(D_1) \in S] \leq e^{\epsilon} \times \Pr[\mathcal{A}(D_2) \in S] \tag{2.2}$$

Equation (2.2) calculates the probability that two neighbor databases have the same result under the same algorithm. Based on the definition, researchers designed the Laplace mechanism to achieve ϵ-differential privacy when the entries have real values. It adds Laplace noise with respect to the sensitivity $\triangle f$ and the desired security parameters ϵ to the result. In particular, the noise is drawn from a Laplace distribution with the density function $p(x|\lambda) = \frac{1}{2\lambda} e^{\frac{-|x|}{\lambda}}$, where $\lambda = \frac{\triangle f}{\epsilon}$.

Theorem 1 (LAPLACE MECHANISM). *For a function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the randomized algorithm $\mathcal{A}$,*

$$\mathcal{A}(G) = f(G) + Lap(\frac{\triangle f}{\epsilon}) \tag{2.3}$$

achieves ϵ-differential privacy [99].

Researchers also designed the exponential mechanism to achieve ϵ-differential privacy when the query's result is an output space instead of a real value [99].

Theorem 2 (EXPONENTIAL MECHANISM). *For a function $f : (G, OS) \rightarrow \mathbb{R}$, the randomized algorithm $\mathcal{A}$ that samples an output O from OS with the probability proportional to* $\exp\left(\frac{\epsilon \cdot f(G,OS)}{2\triangle f}\right)$ *achieves ϵ-differential privacy.*

The exponential mechanism resamples the original output space OS with a new probability sequence. In particular, it assigns exponential probabilities with respect to the sensitivity ($\triangle f$) and the desired security parameters ϵ such that the final output space is smoothed [153].

Nowadays, differential privacy has been widely adopted in privacy preservation for research purposes and commercial purposes, e.g., Apple and Google [36, 139]. Differential privacy theoretically guarantees that the probability of the adversaries to differentiate any piece of information from the released data is bounded. Differential privacy has been applied to protect the electricity usage information [162], to estimate the cardinality of set operations [135], to answer a collection of Structured Query Language (SQL) queries [78].

Similarly to k-anonymity, differential privacy was originally proposed for numerical-type data in databases. The perturbation mechanisms, e.g., the Laplace mechanism, the exponential mechanism, and the random response mechanism, are only designed to add noise to numerical-type data. Hence, researchers designed different graph abstraction models, e.g., the dK graph model [60], the Hierarchical Random Graph (HRG) model [29], and the adjacency matrix model [24], to transform OSN from graph-type data into numerical-type data.

The choice of graph abstraction model restricts the information preserved in the final data. For example, in the degree sequence model (dK-1) or the joint degree model (dK-2), the relationship information involved with more than three nodes is abandon. Hence, we designed a comprehensive model which contains the existing dK-1 and dK-2 model as well as the high dimensional dK-3 model [49].

Although differential privacy provides strict privacy guarantee, graph utility dramatically loses because the criteria aims to hide any piece of data in the whole dataset. The noise is proportional to the size of the dataset and it damages the final output. We combined differential privacy with k-anonymity to design a novel kind of privacy criterion, which called group-based local differential privacy [55]. This novel criterion ensures differential privacy in a local area and achieves k-anonymity among these areas.

A different definition of local differential privacy is also introduced in other researches [80, 81]. Kairouz et al. defined the local as the individual who anonymize his/her data before disclose to the untrusted data curator. Google and other companies adopted this definition to collect personal data [79]. Recently, researchers also apply this definition to anonymize OSNs [121]. Under this definition, the privacy is more strict than the common differential-privacy definition but at the cost of introducing more noise than regular differential-privacy mechanisms. In our work, the local differential privacy is defined based on a trusted curator. Although the curator also anonymizes subgraphs one by one, it should be aware of the global structure in subgraph connection. While the other definition requires OSNs fully locally anonymize the data, our definition holds a global view about the network and locally deals with the network. The results prove that our scheme is an enhancement to common differential-privacy schemes that it reduces unnecessary noise. The two different definitions of local differential privacy have different purpose in advancing privacy or utility.

When analyzing the graph utility for the published data, different anonymization mechanisms may have different advantages. For example, the dK-2 model is good at degree distribution preservation while the HRG model does well in the clustering coefficient preservation. However, our experiments show that none of existing anonymization mechanisms preserve good utility under all utility metrics, and there is no graph utility metric which can comprehensively describe utility [48]. We introduced persistent homology as the summary metric for graph utility. We also designed the anonymization mechanism to preserve differential privacy as well as persistent homology on the adjacency matrix model [50].

Persistent homology is a description of topology [165]. It has many applications, e.g., analyzing persistent aircraft networks [116], calculating the distance between networks [71], and scheduling robot paths in uncertain environments [13]. Persistent homology is novel in security analysis. Speranzon and Bopardikar achieved k-anonymity based on the zigzag persistent homology [21, 134]. Ghrist proposed the barcode to demonstrate persistent homology [58]. It was applied to analyze the structure of the complex network [70] and random complexes [1]. The persistent landscape, which is the abstraction of the barcodes, was also deployed to analyze the topology data [17]. Compared to the landscapes, barcodes present the persistent structures more directly.

The anonymization mechanisms based on k-anonymity, differential privacy, and other privacy criteria all set a specific privacy-level, e.g., k, as their target. However, the impact of these mechanisms to utility is unbounded. Then we proposed a novel anonymization mechanism which has bounded impact to both privacy and utility. Our anonymization mechanism is based on All-Distance Sketch (ADS). This mechanism preserves node distance information extremely well, under a comparable privacy-level with differential privacy based mechanisms.

The most basic sketch is called the MinHash sketch, which randomly summarizes a subset of k items from the original set [16]. Researchers designed three variations of MinHash sketch, named bottom-k, k-mins, and k-partition [30, 32, 34]. Specifically, bottom-k sketch samples k items with the lowest hash values; k-mins sketch samples one item each iteration with the lowest hash value and repeats the iteration k times (in each iteration, the hash values are different); k-partition sketch divides the original set into k subsets and samples one item from each subset. Based on MinHash sketch, researchers define the all-distance sketch to sample the data with graph structures [31]. The main idea of ADS is to keep nodes with the lowest hash values within a specific distance to the central node.

Storing network data into ADS, which is in the format of set of node-distance pairs, saves several orders of space [39]. However, publish this format of data is not appropriate to the third parties who want to analyze the graph utility of OSN. Sketch Retrieval Shortcuts (SRS) is introduced to publish a graph which summarizes ADSs of different nodes [3]. Generally, SRS combines ADS graphs with edge merging. SRS graph is not designed for privacy preserving data publication that it has no false positive and it is vulnerable to attacks.

Although existing privacy and utility measurement works well with previous OSN data application, machine learning methods have been widely applied to graph structure data which impacts both benign third parties and malicious users. Various machine learning methods have been introduced to analyze the graph structure data. Goyal and Ferrara divided them into four categories, e.g., factorization methods, random walk methods, deep learning methods, and other miscellaneous methods [63]. Factorization methods applied spectrum analysis methods to factorize the graph matrices, e.g., the adjacency matrix and the Laplacian matrix [2, 11]. Random walk methods, e.g., DeepWalk and node2vec, used the nodes along a random walk as the feature of the source node [64, 115].

Recently, deep learning methods, which gain huge success in image area and natural language area, are applied to graph data. These methods include the graph convolution network [84, 107], graph attention network [140], gated graph sequence network [92], and graph auto-encoder [83]. These deep learning models aim to learn a vector representation for each node, in which the graph structural information is embedded. For example, the information of source node's 1-hop neighborhoods is embedded in the source node's vector representation after we apply one graph convolution layer.

After learning the vector representation, several downstream learning tasks could be done. These learning tasks are in two categories, node classification tasks and graph classification tasks. Xu et al. showed the difference as the existence of information aggregation in the classification task [155]. Their work also summarized various types of aggregation methods, e.g., sum, average, and max. Sum preserves more information than the other two. Their ideas about the difference between individual learning tasks and group learning tasks inspired our work.

Innovation of the graph learning methods brought the development of real-world applications, e.g., OSN data analysis, as well as the growth of adversaries' interests. Researchers showed the possibility of employing the gradient decent method, which is well-studied in attacking learning of image data, to the discrete graph data [37]. Researchers, behaving as attackers, introduced several attack methods to obtain wrong node classification result [37, 168], change node embedding [14], and damage the learning model [167].

One may notice the similarity between the adversarial attack task to graph data and our privacy preservation task. Both tasks aim to obtain wrong classification results of nodes. While the privacy preservation task has the utility preservation requirement, the adversarial attack task also has the unnoticeable demand. These requirements both limits the total amount of perturbation. However, existing limitation considering in adversarial attacks are a bit outdated. Previous researchers still define unnoticeable as small amount of change to statistics, e.g., limit changes of number of edges, and limit changes to degree distribution [167, 168]. When the adversaries utilized machine learning tools to apply attacks, benign users have unreasonable limited access to use machine learning tools to detect the attacks. This limitation is a bit outdated and not suitable with the development of graph data analysis.

The learning tasks, including node classification and graph classification, are extremely suitable with the OSN data. For example, third parties can apply these machine learning models to group nodes or subgraphs into several categories. Unfortunately, previous researchers did not take the machine learning results preservation in their utility measurement. In the future, we aim to design novel anonymization techniques with updated privacy and utility measurement based on learning.

2.2 Online social network de-anonymization

The attack upon the OSN data, i.e., the de-anonymization of published OSN data, mainly focuses on identify the target users in the released graphs [19, 61]. The adversaries can build an auxiliary graph with their background knowledge. Then the task of finding the target users is transformed into a graph mapping problem [106]. If the adversaries successfully map the nodes from their auxiliary graph into the nodes in the released graph, they can take advantage of the information in the released OSN, e.g., the relationships in the graph and the salary amounts in the profile.

Some existing de-anonymization mechanisms examine both the structure similarity and the attribute similarity of nodes from the two graphs [90, 120]. These de-anonymization attack can be divided into two categories, the seed-based attack and the seed-free attack. In the seed-based attack, attackers first choose high similarity nodes and map them together [4, 77, 147]. Then the attackers design several seed-and-grow algorithms to further expand the mapping [7, 113]. In the seed-free attack, attackers map nodes from the global view of matching probability [74, 110]. For example, the Bayesian model is applied to get the pairwise matching probabilities of nodes in the two graphs [110]. Previous researchers theoretically and experimentally compared the two categories of de-anonymization attacks. Seed-based attacks were believed to have better performance with the same prior knowledge.

In the seed-based attacks, the most important part is the seed-chosen stage. The structure change, which is introduced by both errors in adversaries' background knowledge and the noise injected by anonymization mechanisms, greatly affects the performance of seed chosen [76]. OSN data is time variant although existing de-anonymization attacks mainly focus on the static graphs. For example, Facebook periodically releases their up-to-date OSN data, and the adversary sequentially add his/her new knowledge to the auxiliary graph. De-anonymizing dynamic OSNs should extract the time variant information and employ this kind of information in de-anonymization. Otherwise, the de-anonymization attacks upon dynamic OSN data may face the same problems that are faced when trying to de-anonymize static OSN data.

Existing de-anonymization attack to dynamic OSNs naively combine slices of graphs [41]. A time-series graph is considered as a combination of pieces of graphs. Then the overall probability of mapping two nodes is the product of mapping probabilities in all time-series graphs [41]. Some other work only considers the similarity of path building time when mapping two nodes together [94]. Although these attacks embed some temporal features in de-anonymization, there is not enough temporal information to describe the evolution of OSNs, especially when the OSN graphs are complex. Persistent homology provides a novel angle to analyze the evolution of OSNs. Persistent homology barcodes show the birth time and death time of homology structures, i.e., holes. These structures are utilized as seeds in our de-anonymization attacks [51].

Another shortage of existing de-aonymization attack is that the mapping attack only focuses on the overlap part between the published data and the attackers' background knowledge. However, the published graph may partially cover the target persons or it may not cover them at all. When the attackers seek a one-to-one mapping, losing one user in the published graph will greatly impact the attack performance. Then we introduced the novel generating attack to replace existing mapping attack. The deep neural network structure Generative Adversarial Network (GAN) can learn the high-dimensional structures and generate fake samples which are similar to input [62]. Specifically, GAN designs a game between the two parties, the generator and the discriminator. The generator aims to generate fake samples which can fool the discriminator, while the discriminator aims to discriminate the fake samples with real samples. After this game get equilibrium, our generator can generate OSNs which are very similar to the real ones. Attackers can utilize the generator to produce graph containing target users.

The idea of GAN is based on adversarial machine learning, in which the adversary searches the best angle to add noise to fool the traditional machine learning classifier. GAN extends this conception, in that it adds a virtual adversary in the learning process [103]. The virtual adversary generates confusing samples, which are leveraged to improve the performance and robustness of machine learning models. GAN is widely applied in semi-supervised learning since part of the samples are self-generated and automatically labelled [130].

Moreover, GAN can also be utilized to generate new samples that are similar to inputs. Mirza and Osindero introduced Conditional-GAN, which generates samples under the guidance of conditional information [101]. CGAN was applied to transfer text description to images, extract clothes from dressed-person photos, reconstruct objects from edge maps, colorize images, and transfer day-view photos into night-view ones [72, 158, 160]. When handling graph structure data, e.g., OSNs, knowledge graphs, and recommendation graphs, GAN is applied to learn the graph representation, calculate network embedding, and mimic real-world graphs [9, 15, 143]. However, among the existing studies of GAN on graph structure data focused on high-level structure learning and analysis, few of them have applications in privacy preservation.

GAN is also adopted in privacy attack/defense research due to its capacity for samples generation. Hitaj et al. deployed GAN to attack the online collaborative deep learning system [69]. Unlike the traditional design of GAN, with a virtual adversary, the researchers act like the adversary and force the target person to progressively leak sensitive information in the two-person game. GAN was also deployed in inferring membership, attacking the text captcha system, and so forth [67, 157].

2.3 Privacy preserving online data sharing

Besides social network, several other kinds of online data also formalize as networks, e.g., the cryptographic currency network, the content delivery network. Sharing these kinds of data has similarities and differences with sharing OSN data. For example, in the cryptographic currency network, the privacy concern is similar with the OSN data, i.e., the transaction history should keep private to other individuals. However, the utility concern is different. There is no centralized third party caring about the overall statistics, while the statistics of a specific node may be useful. Studying these kinds of online data gives us insights in OSN data privacy preservation.

2.3.1 Cryptographic currencies

Cryptographic currencies reached a market capitalization of approximately 170 billion dollars in October 2017 [35]. The best known digital currency, Bitcoin (BTC), had a price of 0.005 dollars when it launched in 2009. Eight years later, each BTC equals 5000 dollars, which is 1 million higher than the original value. The explosion in BTC price demonstrates huge success in cryptographic currency. However, some existing cryptographic currencies face privacy leakage problems.

Based on previous digital currency system including eCash and b-money, Nakamoto introduced Bitcoin, which is the most successful cryptographic currency in the world [22, 23, 38]. Ethereum is another important work in cryptographic currency [152].

It includes contracts as well as transactions in the blockchain. While the contracts allow the users to build decentralized applications besides cryptographic currency, these contracts also introduce vulnerabilities such as the DAO attack [6].

Nakamoto claimed that Bitcoin can achieve privacy by keeping public keys anonymous, but researchers studied anonymity of the public blockchain data [68, 108, 126]. Some researchers drew the Bitcoin transaction graph and analyzed the stationary parameters in the graph [108]. Some researchers combined public keys, which are the inputs in the same transaction, and viewed them as the same person [126]. Then some external information, like context discovery and flow analysis, is combined with the graph to de-anonymize the identity.

Because of privacy and latency concerns, some off-chain payment network, e.g., the lightning network and other payment channel networks, are built on top of existing cryptographic currencies [82, 97, 117]. Malavolta et al. studied security and privacy problems in credit networks [96]. They built the pairwise cryptographic credit network, which encourages we should combine it with the blockchain.

2.3.2 Content delivery network

Content Delivery Networks (CDNs) are widely used in data sharing. CDNs distribute high-performance service to end-users according to their spatial position [18]. Some end-users, who directly download the data through a cellular network, can behave as the data servers. CDN then has the ability of mobile data offloading, making the trade-off between the low-cost short-range communications and the high-quality but expensive cellular network. It is first proved that Wi-Fi could be used to build the CDN [8]. The feasibility of communication with bluetooth is discussed [66]. The edge caching technique and the new 5G technique contain the possibility of mobile data offloading in device-to-device communications [10, 104]. Although some of their model also use the Poisson process to model the download requests, these techniques lack a design to guide the behavior of the end-users from the point of view of the global network. In [43], helper caches, i.e., seeds, are totally randomly chosen. Recently, researchers analyzed the topology of the network and proposed specific seeding algorithms to build CDNs [112].

Some current researches are about the caching problems in CDNs. Berger et al. studied the algorithm to choose the hot object to download [12]. Their work is orthogonal to the proposed scheme. When their work is about choosing the right contents to download, our work is about choosing the right users as the servers. Retal et al. designed a platform to provide Content Delivery Network as a Service, which is another good addition to our work [127].

Some other researchers employed content delivery cloudlets to improve the network performance [133]. However, the users should wear a GPS sensor and the system is assumed to be perfectly aware of the moving path, which is unrealistic in real mobile environments. Wang et al. proposed a probabilistic model about the mobility of users [146]. This model analyzed spatial properties and temporal properties. In [154], the authors employed the probabilistic model to embed the social relationships in CDN design, but their scheme is restricted by the particular social network. Nevertheless, previous studies give us insights to design caching schemes based on the probabilistic mobility model.

The study of the CDN data sharing is previously published as a conference paper in IEEE International Conference on Communications (ICC), 2019 [52].

2.3.3 Android application

Android systems are widely used in mobile & wireless environment. However, the Android application data sometimes not shared on official store [150]. The third party applications installed from alternative software repositories may contain malicious code, which cause security and privacy threats to users. We study the Android malware detection problem with the topological signature of applications based on the function call graphs [56, 114].

Our study follows a line of recent works [5, 57, 98, 156, 163**?**] that apply advances in machine learning and data mining for Android malware detection. Some of them were based on semantic information, which includes the signatures, API calls, opcode, and Java methods. DroidAPIMiner focused on API level information within the bytecode since APIs convey substantial semantics about the apps behavior [**?**]. More specifically, DroidAPIMiner extracted the information about critical API calls, their package level information, as well as their parameters and use these features as the input of classification. Droid Analytics designs a signature based analytic system [163]. This system can automatically generate the signatures based on the input Android application's semantic meaning at the opcode level. Unlike previous signature-based approaches, which are vulnerable with bytecode-level transformation attacks, Droid Analytics can defense against repackaging, code obfuscation, and dynamic payloads [125]. Drebin was a combination of previous semantic based detection methods [5]. It extracted string features from multiple Android-specific sources, e.g., intent/permission requests, API calls, network addresses. Although these semantic features directly reflect the application's behavior, novel code encryption and obfuscation method made these methods hard to extract the useful information [46]. In our study, our idea is exploring the application feature space to find some special features, which may be indirect with application's behavior, but they should be hard to be obfuscated.

One major kind of indirect feature space is the structure information. Researchers first builded a FCG to show the relationships between functions. Then, Martinelli et al. compared the subgraphs in the input FCGs with known benign or malicious applications' FCGs, which formulates the malware detection problem as a subgraph mining problem [98]. Zhang et al. introduced weight to FCGs and their FCGs contained both Java methods and APIs [161]. They selected critical APIs and set different weights to nodes when these nodes' APIs have different importance. After that, a similarity score is given between two FCGs to measure the distance when converting one FCG to another, by adding/deleting edges and nodes. In MaMaDroid, Onwuzurike et al. also added API information in FCGs [109]. They used a Markov chain to extract the structural information in FCGs. Although these structure-based detection method focused on the indirect features, all these features, e.g., the big subgraphs, the distance between graphs, and the linear linking relationships, are easy to be obfuscated. For example, adversaries can simply add some edges, i.e., dummy call relationships, to make the malicious subgraph looks benign. In our study, we choose the frequency of graphlets because it is harder to build desired graphlets without affecting existing graphlets. The term of graphlet was first propose by Pržulj et al. [118]. Two recent advances on graph mining, GRAFT [123] and GUISE [124], inspire our use of GFD as a robust and efficient topological signature for apps.

Besides semantic information and structure information, researchers also use other features to enhance static classification performance. FeatureSmith did not directly give the feature space. Instead, it applied Natural Language Processing (NLP) analysis to automatically collect features from other security papers [164]. However, the performance of FeatureSmith relied on other detection methods. DroidSieve used semantic features as well as resource centric features, e.g., certificates and their time, nomenclature, inconsistent representations, incognito applications, and native codes. Although DroidSieve gained success with the comprehensive feature space, it would be vulnerable if the attackers are aware about the feature space and obfuscate every feature.

The study of the Android malware detection is previously published as a conference paper in IEEE International Conference on Communications and Network Security (CNS), 2016 [114]. Then it is extended as a journal article in IEEE Transactions on Mobile Computing (TMC), 2018 [56].

3. ANONYMIZATION WITH PRIVACY CRITERION - LOCAL DIFFERENTIAL PRIVACY

In this chapter, we begin with studying the privacy and utility preservation in existing anonymization mechanisms. Our analysis shows that existing anonymization mechanisms choose the global differential privacy, which is so strict that significantly damage the utility preservation. Hence, this chapter defines the notion of group-based local differential privacy. In particular, by resolving the network into 1-neighborhood graphs and applying HRG-based methods, our scheme preserves differential privacy and reduces the noise scale on the local graphs. By deploying the grouping algorithm, our scheme abandons the attempt to anonymize every relationship to be ordinary, but we focus on the similarities in HRG models. In the final released graph, each individual user in one group is not distinguishable, which greatly enhances the OSN privacy.

The major technical contributions of this chapter are the following:

1. We define the notion of local differential privacy, which could preserve more information when the privacy level is the same as global differential privacy.
2. We group the nodes with similar local features. By carefully designing two heuristic methods, we show the grouping algorithm could enhance the privacy level without loss of too much information.
3. We design a uniform framework to publish perturbed networks, satisfying group-based local ϵ-differential privacy.

This chapter is previously published as a conference paper in International Conference on Wireless Algorithms, Systems, and Applications (WASA), 2017 [54]. The extended version of this chapter is published as a journal article in IEEE Transactions on Computational Social Systems, 2018 [55].

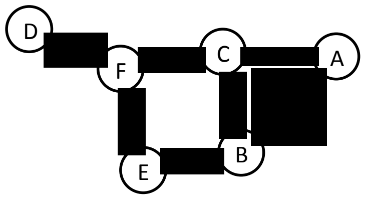
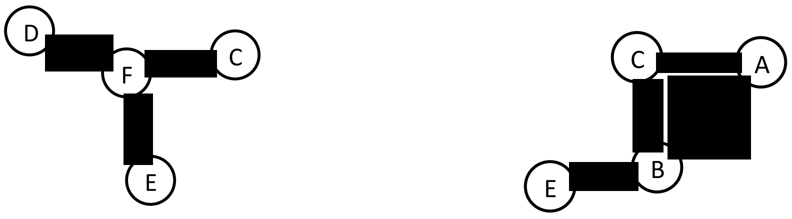


Fig. 3.1.: Toy example of the OSN graph, nodes are users and edges are linking relationships



(a)1-neighborhood graph G(F) (b)1-neighborhood graph G(B)

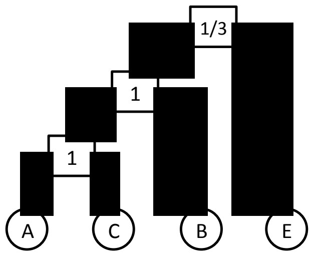
Fig. 3.2.: 1-neighborhood graphs getting from graph in Figure 3.1

3.1 Preliminaries

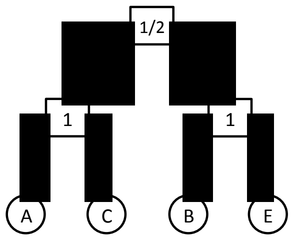
In this chapter, an online social network graph is modeled as an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. $|V|$ is the cardinality of the set V. Figure 3.1 shows a toy example of the OSN graph.

3.1.1 1-Neighborhood Graph

For each node v in V, we define its 1-neighborhood graph to contain all the neighbors of v and the node v itself [93]. The 1-neighborhood graph of v is denoted as $G(v) = (V(v), E(v))$, where $V(v) = v \cup \{u|e_{v,u} \in E\}$ and $E(v) = \{e_{w,u}|w, u \in$



(a) HRG $\mathcal{T}_{B1}$



(b) HRG $\mathcal{T}_{B2}$

Fig. 3.3.: HRGs generated from G(B)

$V(v) \wedge e_{w,u} \in E\}$. The node v is marked as the central node of the 1-neighborhood graph. Figure 3.2 gives two 1-neighborhood graphs of the original graph in Figure 3.2(a). The two subgraphs have the central nodes F and B.

3.1.2 Hierarchical random graph model

Because the connection probability between two vertices depends on their degrees, the HRG model is captured by statistical collection [28]. Specifically, an HRG model here is an HRG $\mathcal{T}$, which is a rooted binary tree with $|V|$ leaf nodes corresponding to $|V|$ vertices in the graph G. Each node on the tree except the leaf node has a number on it that shows the probability of connection between its left part and right part. Assume r is one of the interior nodes of the HRG $\mathcal{T}$; then the probability is denoted as p_r. For example, Figure 3.3(a) is an HRG of the graph in Figure 3.2(b). The four leaf nodes on the tree correspond to the nodes in the graph. And the connection probability p_r of the subtrees {A,C,B} and {E} is $\frac{1}{3}$.

Let L_r and R_r denote the left and right subtrees rooted at r respectively. n_{Lr} and n_{Rr} are the numbers of leaf nodes in L_r and R_r. Let E_r be the total number of edges between the two groups of nodes L_r and R_r. Then, the posterior probability for the subtrees rooted at r is $p_r{=}E_r/(n_{Lr}n_{Rr})$. The posterior probability of the whole HRG model $\mathcal{T}$ to represent G is given by,

$$p(\mathcal{T}) = \prod_{r\in\mathcal{T}} p_r^{E_r}(1-p_r)^{n_{Lr}n_{Rr}-E_r} \tag{3.1}$$

Figure 3.3 gives an example of two possible HRGs of B's 1-hop neighborhood graph in Figure 1(c). The p_r in each root node is first calculated. For instance, in the HRG $\mathcal{T}_{B2}$, the root node of subtrees {A, C} and {B, E} has a probability 1/2. Because there are two edges between the two sets of nodes, we have E$_r$=2, p_r=2/(2*2)=1/2. Then we get the posterior probability of the two HRGs. $p(\mathcal{T}_{B1}) = (1/3)(2/3)^2 \approx 0.148$ while $p(\mathcal{T}_{B2}) = (1/2)^2(1/2)^2 \approx 0.006$. $p(\mathcal{T}_{B1})$ is greater than $p(\mathcal{T}_{B2})$, so $\mathcal{T}_{B1}$ has more probability to represent G(B). In addition, since the size of 1-hop neighborhood graphs are often small, there are few candidate HRGs. Actually, if the sequential change of leaf nodes is ignored, G(B) just has two possible structures of HRG shown in Figure 2, and $\mathcal{T}_{B1}$ is the more plausible one.

3.2 Scheme

Given a OSN graph, our goal is to publish an anonymized graph that preserves the structural utility as much as possible, while satisfying the privacy criteria. The overall diagram of the scheme is shown in Figure 3.4. There are four steps, as follows:

1. Finding the approximate maximum independent set and getting the 1-neighborhood graph of each node in the set.
2. Extracting the HRGs to each node's subgraph under the criteria of differential privacy.
3. Grouping the HRGs and sampling one representative for each group.
4. Regenerating the 1-neighborhood graph and pasting the sanitized one to the whole graph.

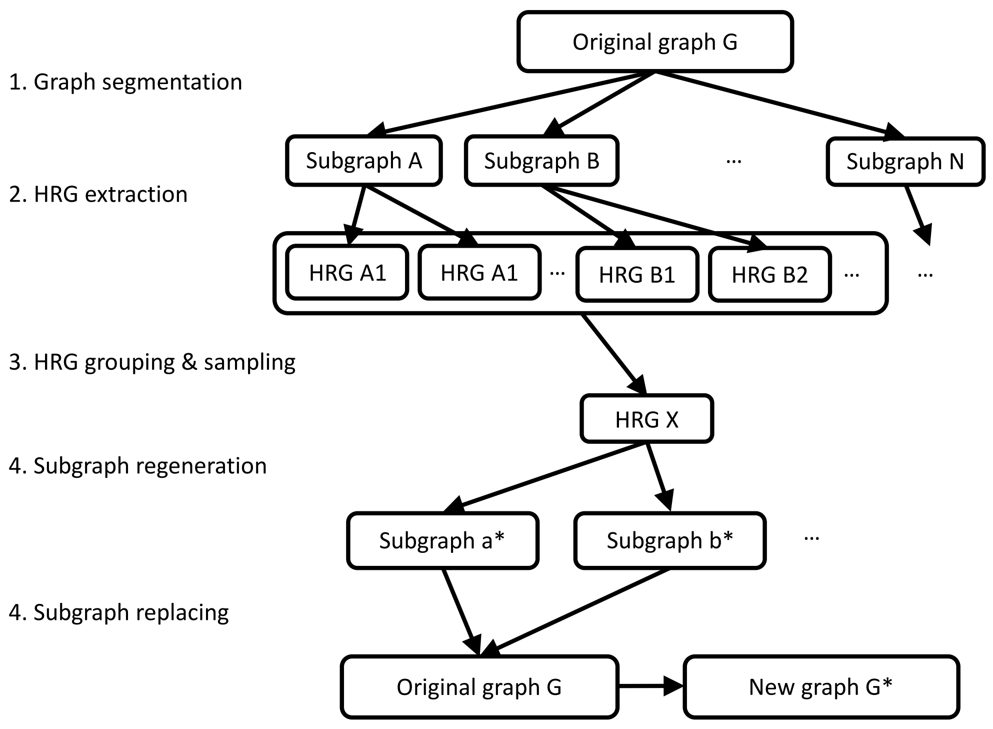


Fig. 3.4.: Scheme diagram

3.2.1 Group based local differential privacy

To preserve link privacy, previous research advocated differential privacy, where the output changes at a small probability (less than e^{ϵ}) with the modification of one of its tuple [129, 153]. It is a rigorous privacy guarantee and it may create a significant negative impact on utility because the amount of necessary noise is proportional to the complete graph size, which is a huge number in online social network analysis. Researchers proposed different techniques to reduce the noise. Sala et al., for example, sorted and clustered the query results so that less noise was needed in some clusters [129].

Instead of hiding every link in the network with the same probability, we reduce the scale of the network to 1-neighborhood graphs. Also, the grouping algorithm is adopted in our scheme to construct a confidential group. The users are hidden with the users having similar structural information, instead of being hidden with all users in the network.

In this chapter, we define the concept of group-based local ϵ-differential privacy, which can preserve privacy with less information loss.

Definition 4 (GROUP-BASED LOCAL ϵ-DIFFERENTIAL PRIVACY). *For a group of at least k nodes, a randomized algorithm $\mathcal{A}$ extracts local features. $\mathcal{A}$ achieves group-based local ϵ-differential privacy if for all neighbor graphs, D_1 and D_2, with one edge adding/deleting, the resultant probability* $\Pr[\mathcal{A}(D) \in S]$ *satisfies Equation (2.2).*

In the following sections, our anonymized graph satisfies group-based local ϵ-differential privacy.

3.2.2 Maximum independent set

Since we split the original network G into multiple 1-neighborhood graphs and then perturb them, we carefully choose a set of subgraphs that could be sanitized together without mutual influence. Initially, this requires that the two subgraphs' central nodes are neither adjacent nor have common neighbors.

However, if the central nodes are not adjacent and the subgraph sizes are not smaller than the original, we find a solution to assign the outer nodes and then the perturbed subgraph can replace the original. Here, outer nodes means the nodes in $V(v)$ excluding the central node v. The definition of the HRG model indicates that the leaf nodes in the HRG are always equal to the total size of the graph, which means that the perturbation work does not change the number $|V(v)|$. Hence, the straightforward approach is to search the non-adjacent nodes, which is also called searching the maximum independent set of graph G.

Algorithm 1 Find independent sets

```
Input: $G$: the original graph
Output: $Set_1$: the approximate maximum independent set
 1: $G_1 \leftarrow G$
 2: $|V| \leftarrow G$'s total number of nodes
 3: Let S be an empty stack
 4: while size(S) $< |V|$ do    ▶ the stack S is not full
 5:     $x \leftarrow$ the node with lowest degree in $G_1$
 6:     S.push(x)
 7:     $d \leftarrow$ the maximum number of neighbors
 8:     $G_1 \leftarrow G_1 - x$    ▶ delete x's 1-neighborhood graph from the graph $G_1$
 9: end while
10: while size(S) $> 0$ do    ▶ the stack is not empty
11:     $x \leftarrow$ S.pop()    ▶ let x be the node painted now
12:     $G_1 = G_1 + x$    ▶ add x to the graph $G_1$ according to graph $G$
13:     $c \leftarrow$ color label    ▶ paint the node x using the lowest label that x's neighbors
    haven't used
14: end while
15: $\{Set_1, Set_2, ..., Set_d\} \leftarrow G(c_n)$    ▶ divide the nodes according to the color label
16: return $Set_1$
```

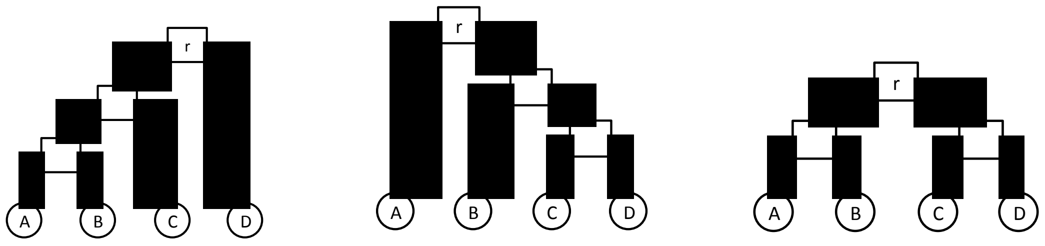
The problem of finding the maximum independent set is a NP-hard optimization problem. There are some greedy algorithms to find the approximation results. However, as differential privacy is defined on edges in this chapter, the algorithm should guarantee that all edges are contained in at least one 1-neighborhood graph. Hence, the output set should be the maximum independent set as well as the dominating set of the graph. Although there are several greedy algorithms for finding independent set or dominating set, few of them combine the two purposes together.

Here we propose Algorithm 1 to get the approximation result. Algorithm 1 is inspired by the graph coloring algorithm. The adjacent nodes could not have the same color, which is similar to the limitation of the independent set. Because Algorithm 1 requires the color label to be as low as possible, every node in G is painted as label 1 or is a neighbor of label 1 nodes, which means that the Set_1 is also a dominating set. Therefore, searching the independent set maintains the privacy guarantee of our algorithm.

3.2.3 HRG extraction

After collecting the maximum independent set, we get the 1-neighborhood graphs whose central nodes are in Set_1. Then the HRGs of these 1-neighborhood graphs are extracted. The Hierarchical Random Graph (HRG) model is deployed to capture the local features because it is easy to integrate local ϵ-differential privacy into the HRG and a new graph could be regenerated from the sanitized HRG. Also the HRG model tends to group a cluster of nodes in the same branch on the tree, which preserves more clustering information compared with other models. In this section, we first introduce the work to extract HRG model, then derive the amount of noise necessary to achieve a given local ϵ-privacy level.

The number of possible HRGs is $|\mathcal{T}|$=(2$|V|$-3)!! for a network with $|V|$ vertices, where !! is the semi-factorial symbol. Compared with the global extraction, the segmentation work largely reduces the size of graph so as the amount of computation. Furthermore, each single HRG has higher posterior probability to represent the graph because the HRG output space OS is reduced. It means that regenerating a graph from a HRG loses less utility compared with the global technique. However, extracting the entire output space OS is still expensive especially for large subgraphs. In our scheme, we introduce a Markov chain Monte Carlo (MCMC) process to control the time complexity and give an approximate result.



(a) Original (b) Sequential change (c) Structural change

Fig. 3.5.: Two neighbor configuration samples of r's subtree

In a profile $\{\mathcal{T}^1 ... \mathcal{T}^m\}$ with the desired size m, the $\mathcal{T}^1$ stores the HRG with the highest posterior probability $p(\mathcal{T}^1)$, $\mathcal{T}^2$ is the second highest, etc. Specifically, Algorithm 2 first chooses an initial HRG $\mathcal{T}_0$. Assume $\mathcal{T}_{i-1}^1$ is the most possible HRG of the last step, then in the new step the MCMC process randomly chooses a root node r in $\mathcal{T}_{i-1}^1$ and then configure a neighbor HRG of $\mathcal{T}_{i-1}^1$ called $\mathcal{T}'$. There are $A_4^1 * 2 = 48$ candidate neighbors of a four-leaf-node subtree and two of the neighbor examples is shown in Figure 3.5. The HRGs in the profile are replaced by $\mathcal{T}'$ at the acceptance ratio $\frac{p(\mathcal{T}')}{p(\mathcal{T}_{i-1})}$. When equilibrium of $p(\mathcal{T}^1)$ is reached, the set of m possible HRGs are stored.

According to Theorem 3, if the expected result is to achieve ϵ-differential privacy, there should be another sample process after we draw the original output space. In Algorithm 2, the MCMC process picks the HRG profile and simulate the exponential mechanism at the same time. The exponential mechanism requires to resample the output space OS with the probability $\exp\left(\frac{\epsilon \cdot f(G,OS)}{2\triangle f}\right)$. So the acceptance ratio is changed from $\frac{p(\mathcal{T}')}{p(\mathcal{T}_{i-1})}$ to $\frac{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}')\right)}{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}_{i-1})\right)}$.

Here, we still need to analyze the local sensitivity $\triangle$f to finish the acceptance ratio equation. We consider the ϵ-differential privacy only in the link privacy area, which means the neighbor of a graph is a graph with just one edge changes according to Definition 4. We assume the edge is missing without loss of generality. So the sensitivity could be denoted as:

$$\triangle f = \max\left(p(\mathcal{T}(E_r)) - p(\mathcal{T}(E_r - 1))\right) \tag{3.2}$$

In order to simplify the analysis, we calculate the log-based sensitivity.

$$\log(\triangle f) = \max\left(n_{Lr}n_{Rr}\left(h\left(\frac{E_r}{n_{Lr}n_{Rr}}\right) - h\left(\frac{E_r - 1}{n_{Lr}n_{Rr}}\right)\right)\right) \tag{3.3}$$

where h is the entropy function and $h(p) = -plog(p) - (1-p)\log(1-p)$.

After analyzing the relationship between the $\log(\triangle f)$ and $n_{Lr} \cdot n_{Rr}$, we find that $\log(\triangle f)$ monotonically increases when $n_{Lr} \cdot n_{Rr}$ increase. Because $\triangle f$ shows the maximum distance between two neighborhood databases, it gets the value when n_{Lr} and n_{Rr} have the same value equal to half of the total vertices number $\frac{|V|}{2}$. And we have,

$$\begin{aligned} \triangle f &= \frac{|V|^2}{4} * \left(1 + \frac{1}{\frac{|V|^2}{4} - 1}\right)^{\frac{|V|^2}{4} - 1} \\ \log(\triangle f) &= \log\left(\frac{|V|^2}{4}\right) + \left(\frac{|V|^2}{4} - 1\right)\log\left(1 + \frac{1}{\frac{|V|^2}{4} - 1}\right) \end{aligned} \tag{3.4}$$

We can make,

$$\begin{aligned} y &= \left(\frac{|V|^2}{4} - 1\right) \\ \log\left(1 + \frac{1}{y}\right)^y &= \left(\frac{|V|^2}{4} - 1\right)\log\left(1 + \frac{1}{\frac{|V|^2}{4} - 1}\right) \end{aligned} \tag{3.5}$$

Because $y \in \mathrm{R}^+$ when there are more than two nodes, we can further zoom the $\log(\triangle f)$ as follows:

$$\begin{aligned} \log\left(1 + \frac{1}{y}\right)^y &\leqslant \log e \\ \log(\triangle f) &\leqslant \log\left(\frac{|V|^2}{4}\right) + 1 \end{aligned} \tag{3.6}$$

Hence, if we use $\log\left(\frac{|V|^2}{4}\right) + 1$ as the log-based sensitivity, the differential-privacy criteria is satisfied. The amplitude of noise increases as $|V|$ increases. Because $|V|$ is the total number of nodes in the graph, Resolving the graph greatly reduce the size

Algorithm 2 Extract differential private HRG profile

```
Input: G(v): the subgraph, m: profile size,
       ϵ: privacy parameter
Output: HRG profile $\{\mathcal{T}^1 ... \mathcal{T}^m\}$
 1: $\triangle f \leftarrow f(G)$   ▶ calculate the local sensitivity according to the size of graph $G(v)$
 2: choose a random starting HRG $\mathcal{T}_0$
 3: $\{\mathcal{T}^1 ... \mathcal{T}^m\} \leftarrow \mathcal{T}_0$
 4: while step number i < maximum iteration time do
 5:     randomly pick an internal node r
 6:     pick a neighbor HRG $\mathcal{T}'$ of $\mathcal{T}^1_{i-1}$ by randomly
        choosing a configuration of r's subtrees
 7:     $\mathcal{T}^1_i \leftarrow \mathcal{T}'$ with the probability
        $\min\left(1, \frac{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}')\right)}{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}^1_{i-1})\right)}\right)$
 8:     ...
 9:     $\mathcal{T}^m_i \leftarrow \mathcal{T}'$ with the probability
        $\min\left(1, \frac{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}')\right)}{\exp\left(\frac{\epsilon}{2\triangle f} p(\mathcal{T}^m_{i-1})\right)}\right)$
10:     if equilibrium of $p(\mathcal{T}^1_i)$ is researched then
11:         break
12:     end if
13: end while
```

of 1-neighborhood graph from $|V|$ to $|V(v)|$. Prior studies have demonstrated that in large network graphs, the maximum value of $|V(v)|$ is upper bounded by $\mathcal{O}(\sqrt{|V|})$ [87]. Furthermore, our algorithm adds sufficient noise to different HRGs according to different $|V(v)|$ but not the maximum value. So if the desired privacy criteria ϵ is the same, there is more utility preserved under the local ϵ-differential privacy compared with the global ϵ-differential privacy.

3.2.4 HRG grouping and sampling

Instead of hiding every user/relationship globally, we only consider the 1-neighborhood graph with local features in the previous sections. Here our scheme adopts a grouping algorithm to enhance the privacy power, which means that although the user could not hide behind the whole graph, it hides in a group with other users having similar structural information. The general procedure here is to group the similar HRGs together and make them indistinguishable.

Intuitively, the HRGs extracted from the same 1-neighborhood graph should be grouped together. Based on this stating point, the procedure of HRGs grouping can also be viewed as the procedure of node grouping. Since the number of leaf nodes in an HRG is equal to the number of nodes in the original graph, only the subgraphs with the same size may have overlap in their output HRG space OS. Hence, for a given graph $G = (V, E)$, we group nodes $\{v\} \in V$ according to the metric $|V(v)|$, number of nodes in its 1-hop neighborhood graph.

Although the group formulation procedure groups the subgraphs with the same sizes together, not all groups have a size greater than or equal to our desired size k. Therefore, we merge the small groups if they have the most similar $|V(v)|$ to make sure each group has an appropriate size which is at least k. Then, the sampling space OS is grouped together, each group contains at least $k * m$ candidate HRGs.

To achieve the group-based local ϵ-differential privacy, each group chooses a representative HRG from the group's output space OS. There is a naive method that samples one HRG according to its probability $\exp\left(\frac{\epsilon}{2\triangle f}p(\mathcal{T})\right)$ in the profile. In the groups of HRGs with exact same number of leaf nodes, the naive method works well. However, several groups have different $|V(v)|$ because they were merged. A small graph cannot be used to replace a large one because there is not a solution to assign the outer nodes. Hence the naive method requires a representative HRG with at least $|V| - 1$ leaf nodes. It's not hard to imagine that the naive method uses larger sanitized graphs to replace smaller ones and add a huge group of dummy nodes. It damages the utility of final graph. In order to preserve the node number, we design the following two different methods.

Method 1 (VIRTUAL NODE). In the begining, we sort the node numbers of subgraphs to $\{|V_1|, |V_2|, ..., |V_{med}|, ..., |V_k|\}$. Assuming $|V_1|$ is the largest number and $|V_{med}|$ means the median number. Then the group representative HRG is chosen from the HRGs with $|V_{med}|$ leaf nodes, because we want to use the $|V_{med}|$ to average other subgraphs' size in the group.

If the subgraph's size is larger than $|V_{med}|$, taking $|V_1|$ as the example, we make the following changes. First, we get the central node of the 1-neighborhood graph, denoted as u. Second, we add $\left\lceil \frac{|V_1|}{|V_{med}|} \right\rceil$-1 virtual nodes into the original graph G, u's $|V_1|$ edges are partition into these virtual nodes and u averagely. Third, since u's and virtual nodes' subgraphs are all smaller than $|V_{med}|$, their sanitized graph could be generated from HRG with $|V_{med}|$ leaf nodes. Finally, we combine all the virtual nodes to u.

Method 2 (OUTLIER DISTINCTION). Compared with Method 1, there is no special change to the subgraphs but the outliers with more special information are caught and treated differently. Specifically, we calculate the standard deviation of the size sequence $\{|V_1|, |V_2|, ..., |V_k|\}$ and set a threshold *std* to the standard deviation. If the group's standard deviation is greater than *std*, we break the group and use each subgraph's own HRGs to get the sanitized graph.

In this method, we use an absolute value, standard deviation, to simulate the amount of dummy nodes needed in the naive method. Hence, the perturbation which adds too much dummy nodes is prevented. Although not every subgraph is grouped, if we choose an appropriate *std*, it does not affect the privacy guarantee too much. Because the group breaking only happens when the standard deviation is greater than the threshold, in most of the real-world social networks, these groups has very high degree nodes and their degrees are different. According to the previous research in [75], these high-degree outliers carrying more structural information are more vulnerable to structural de-anonymization attacks. So instead of being considered in the same way, different users have different importance in grouping and sampling. In this method, these outliers are not grouped with other users.

In the HRG extraction algorithm in Section 3.2.3, our scheme introduces noise proportional to ϵ to make each node's subgraph similar to all its possible neighbors. Not like the particular group-mates in the grouping algorithm, differential privacy uses a manner to create neighbors, or we can call it building synthetic group-mates. In the group sampling algorithm in this section, our scheme also finds $k-1$ particular group-mates for each subgraph, and makes these group-mates extremely similar to each other. Hence, an attacker is not able to identify the target node from a confidential group of at least k members even with the help of releasing graph and prior knowledge of 1-neighborhood relationship.

3.2.5 Subgraph regeneration and connection

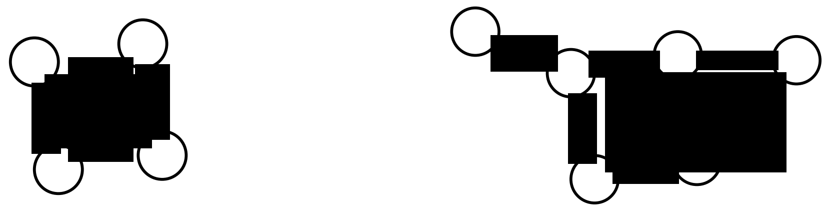
In the last part of our scheme, the subgraphs are restored from HRGs and we want to publish the entire perturbed graph $\tilde{G}$. Firstly, the sanitized 1-neighborhood graph is generated according to the group representative HRG. It is shown in the Subgraph Regeneration procedure in Algorithm 3. For each internal node r, the algorithm randomly generates E_r edges between the two node groups L_r and R_r.

Secondly, the sanitized subgraph replaces the original 1-neighborhood graph. Specifically, for each node v, the sanitized graph randomly chooses $|V(v)|$-1 nodes as v's neighbor, and we call them v's outer nodes. No auxiliary information like the degree sequence is used to choose these outer nodes to maintain the privacy guarantee. The perturbed graph could be easily pasted on the original graph G when the neighbor nodes' label is changed to its corresponding label in the original graph. However, the connecting algorithm forces to deal with the subgraphs having at least $|V(v)|$-1 nodes. A small graph is not appropriate to replace a large subgraph because it does not have enough outer nodes. Hence in Section 3.2.4, we add a restriction in the

Algorithm 3 Subgraph regeneration and connection

```
Input: $G$: the original graph,
        $\{\mathcal{T}^1 ... \mathcal{T}^{\lfloor \frac{|V|}{k} \rfloor}\}$: representative HRGs for each group
Output: A perturbed graph $\tilde{G}$
 1: for each node $v \in V$ do
 2:     $\mathcal{T} \leftarrow$ v's representative HRG
 3:     procedure Subgraph Regeneration($v$,$\mathcal{T}$)
 4:         for each internal node r $\in \mathcal{T}$ do
 5:             $E_r \leftarrow p_r * n_{Lr} * n_{Rr}$    ▶ $p_r$ is recorded in $\mathcal{T}$
 6:             find the two groups L$_r$ and R$_r$
 7:             randomly place E$_r$ edges between nodes from
                L$_r$ and nodes from R$_r$
 8:         end for
 9:     end procedure
10:     random choose $|V(v)| - 1$ nodes in $\tilde{G}(v)$    ▶ $v$
        has $|V(v)| - 1$ neighbors in $G$
11:     $\tilde{G} \leftarrow G + \tilde{G}(v)$    ▶ paste the perturbed subgraph
        according to the neighbors
12:     $\tilde{G} \leftarrow \tilde{G} - G(v)$    ▶ cut v's original 1-neighborhood
        graph
13: end for
14: return $\tilde{G}$    ▶ For every node in the independent set, its 1-neighborhood graph
    has been replaced
```

group sampling algorithm that the HRGs having less than $|V(v)|_{max}$-1 leaf nodes do not contribute to the group sampling result, where $|V(v)|_{max}$ means the maximum subgraph size in the group.



(a) Sanitized subgraph from $\mathcal{T}_{B2}$ (b) Graph $\tilde{G}$ with perturbed G(B)

Fig. 3.6.: One possible change on G(B)

Table 3.1.: Network dataset statistics

Dataset	# of nodes	# of edges	Max subgraph size
Facebook	4039	88234	1045
Enron	33692	183831	1383
ca-HepPh	12008	118521	491
BA graph	10000	49975	418

Figure 3.6 shows an example of subgraph regeneration and connection. The original subgraph is G(B) in Figure 1(c), and the sanitized subgraph is based on $\mathcal{T}_{B2}$ in Figure 2(b). The HRG $\mathcal{T}_{B2}$ requires to have two pairs of linked nodes, and they are randomly connected with two edges. Figure 4(a) is one possible sanitized subgraph rather than the original G(B). Then the Algorithm 3 randomly chooses three nodes in the sanitized subgraph to be the node A, C, and E in the original graph. Using the three nodes, the sanitized subgraph is pasted on the original graph. Finally, for simplicity and anonymization purpose, the sanitized subgraphs do not contain the labels, as well as the final releasing graph $\tilde{G}$.

3.3 Evaluation

In this section, we evaluate our anonymization scheme over three real-world datasets, namely Facebook, Enron and ca-HepPh [88], and a synthetic network, Barabási-Albert (BA) graph. Facebook is a famous social network containing users and their relationships. Enron email communication network covers around half million emails in Enron. ca-HepPh is collaboration networks which cover scientific collaboration among authors in the area of high energy physics. BA graph is a generated random scale-free network, while OSNs are thought to be approximately scale-free. Due to space constraint, results of some datasets are omitted and readers can find them in [54]. The statistics of these datasets are given in Table 5.2. It shows that the original graph is 4 to 30 times bigger than the biggest 1-neighborhood graph. All experiments have been done on a desktop workstation (8-core Intel Core i7-3820 CPU at 3.60GHz with 12GB RAM).

3.3.1 Experimental settings

Table 3.2 shows the important parameters in our experiment. ϵ is a privacy parameter to measure the ability of hiding existence edges. The smaller ϵ is, the better the privacy protection is [153]. In this chapter, we set a strict criteria where ϵ=0.1 to test the utility performance of the local privacy scheme. We set the minimum group size to 10, corresponding to the size of networks, which contain 400-1300 users. Each subgraph has 3 candidate HRGs, so the representative HRG is sampled from the profile with at least 30 HRGs. In the 'outlier distinction' method, the threshold *std* is initially set to 5 and the Facebook, Enron, ca-HepPh datasets results show that there are only 0.50%, 0.03%, 0.18% nodes are not grouped and their degrees are all greater than 45. So these nodes can be treated as outliers appropriately. We also test the impact of different ϵ and *std*.

Table 3.2.: Parameters

Differential privacy parameter ϵ	0.1
Group minimum size k	10
HRG profile size of one subgraph m	3
Standard deviation threshold std	5

For comparison purposes, we implement one state-of-the-art technique, the basic global differential-privacy algorithm with HRG models in [153] under the same privacy criteria. Our evaluation is based on the python implementation of the work in [28]. In the following figures, previous global differential-privacy HRG scheme's result is marked as 'reference', two of our methods are marked as their name 'virtual node' and 'outlier distinction'.

3.3.2 Evaluation result

To show the scalability of our scheme, we test the running time of the algorithms. To show the utility of the released networks, we compare three most robust measures of network topology, the degree distribution, the shortest-path length and the clustering results.

Running time. We test the efficiency of our two sub-schemes and the reference scheme with different input graph size. Specifically, we cut the Enron email graph into subgraphs with desired number of nodes from 1k to 15k. Then the algorithms are evaluated on these subgraphs and the running time is shown in Figure 3.7. The running time of the reference scheme linearly increases with the size of the graph. Every 1k nodes addition results in an increase of about 120s in running time. The running time of our two methods are shorter especially when the graph size is huge. The 'virtual node' and the 'outlier distinction' methods only take about 230s to deal with a graph having 10k nodes.

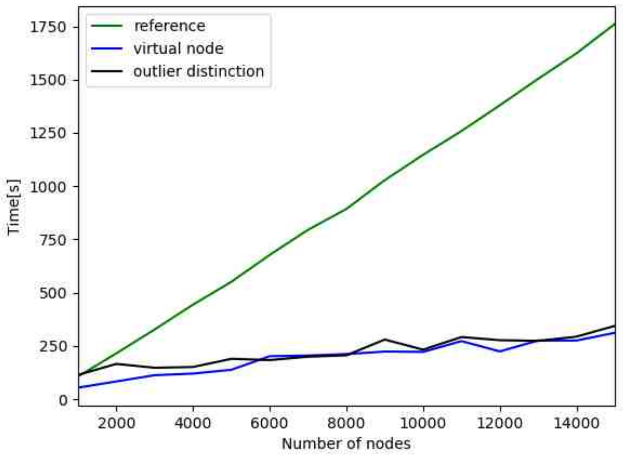


Fig. 3.7.: Running time with different sizes of graphs

Although a graph with $|V|$ nodes have (2$|V|$-3)!! possible HRGs, the MCMC procedure gives the upper bound of the HRG space size to all the three methods. The difference in running time mainly come from the difference of the complexity of building HRGs. In the reference method, the global algorithm has no splitting so that the number of nodes $|V|$ of the graph is also the number of leaf nodes of the HRG. The result of the reference method demonstrates that the complexity is approximately linearly correlated with the number of nodes.

In our two methods, splitting helps to reduce the HRG size from $|V|$ to $V(v)$, and $V(v)$ has an upper bound $\mathcal{O}(\sqrt{|V|})$ [87]. The result of the 'outlier distinction' method shows that the running time is in proportional to $\sqrt{|V|}$. The 'virtual node' method further divides the big subgraphs. And this division significantly reduce the running time when the graph is small. The results prove that our two methods are scaleable to real OSNs with millions of nodes and they are more efficient than the reference method.

Degree distribution. Degree distribution is a utility metric to show the similarity of two graphs. For example, if the third-parties want to count the number of users with 5-10 friends in the OSN, graphs with a similar degree distribution can give more

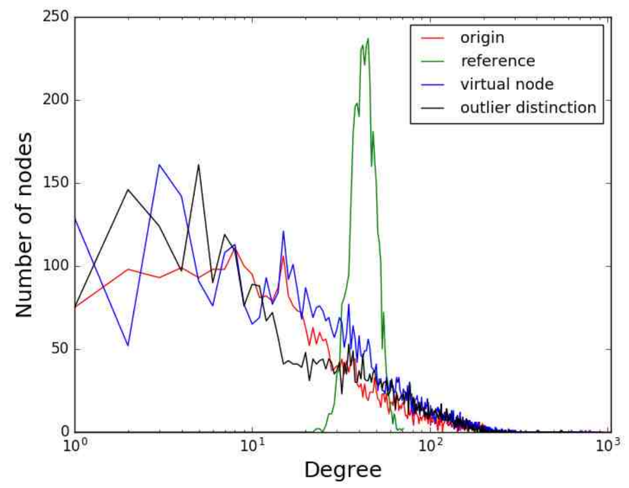


Fig. 3.8.: Degree distribution of Facebook, $\epsilon = 0.1$

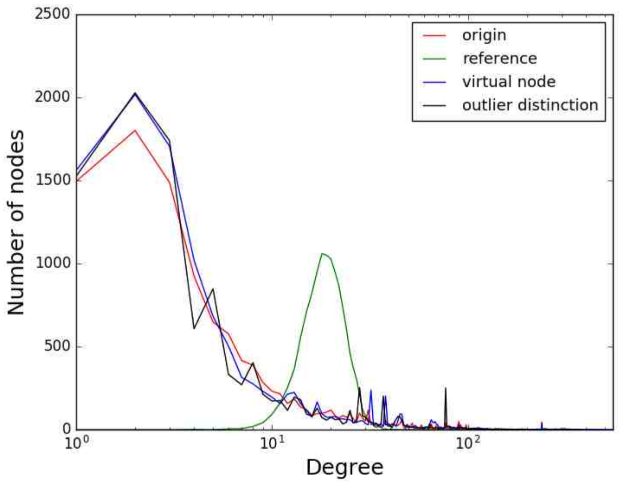


Fig. 3.9.: Degree distribution of ca-HepPh, $\epsilon = 0.1$

precise information. Figure 3.8 and 3.9 show the degree distribution of the Facebook dataset and ca-HepPh. For better presentation, we use a base-10 log scale for the X axis because these schemes have different results in the low degree space.

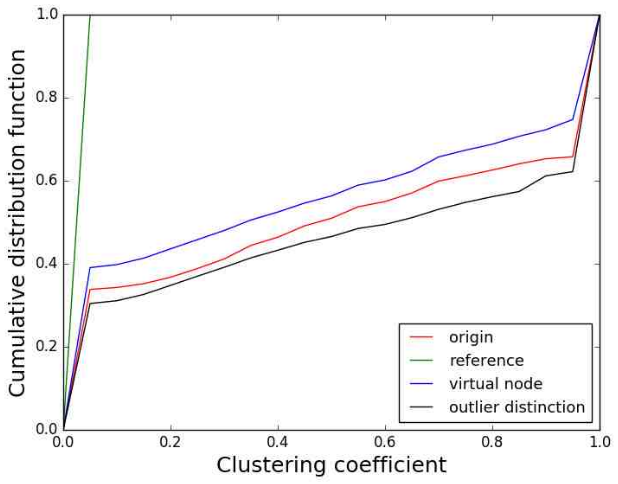


Fig. 3.10.: Clustering coefficient distribution of Enron, $\epsilon = 0.1$

Our two methods follow the trend that there are less nodes when degree becomes higher as the origin. However, the global differential-privacy scheme has a new trend that the degree centralizes in a small range. There is no node degree lower than 23 in the reference anonymized result while other results have a lot of low degree nodes. The strict differential-privacy criteria makes the nodes to be similar with each other. So in the global algorithm result, the degree distribution is centralized in a small area. The results under local differential privacy just have slight change because they only take the local neighborhoods into consideration and the node is easy to be an ordinary one. The global differential-privacy result has worse utility because it needs to hide every node under a global version.

Clustering. Clustering coefficient is a measure of how nodes in a graph tend to cluster together. Similar clustering coefficient distribution means the graph is a good simulation of the clustering behavior in the original graph. While other models like dk2 [129] may break the features of cluster, HRG model is believed to protect some clustering information because it is a procedure of grouping close nodes together to build the HRG.

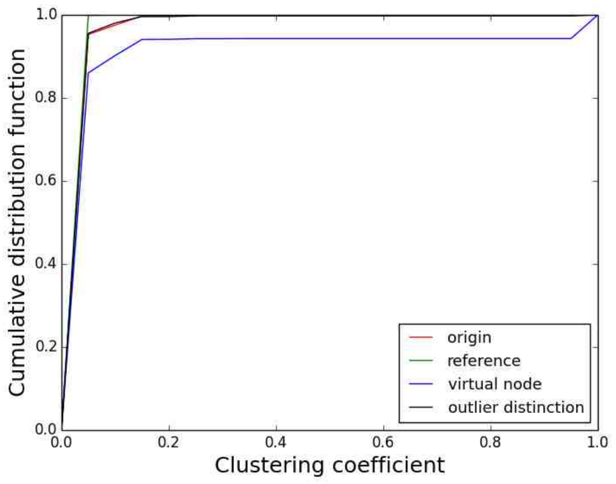
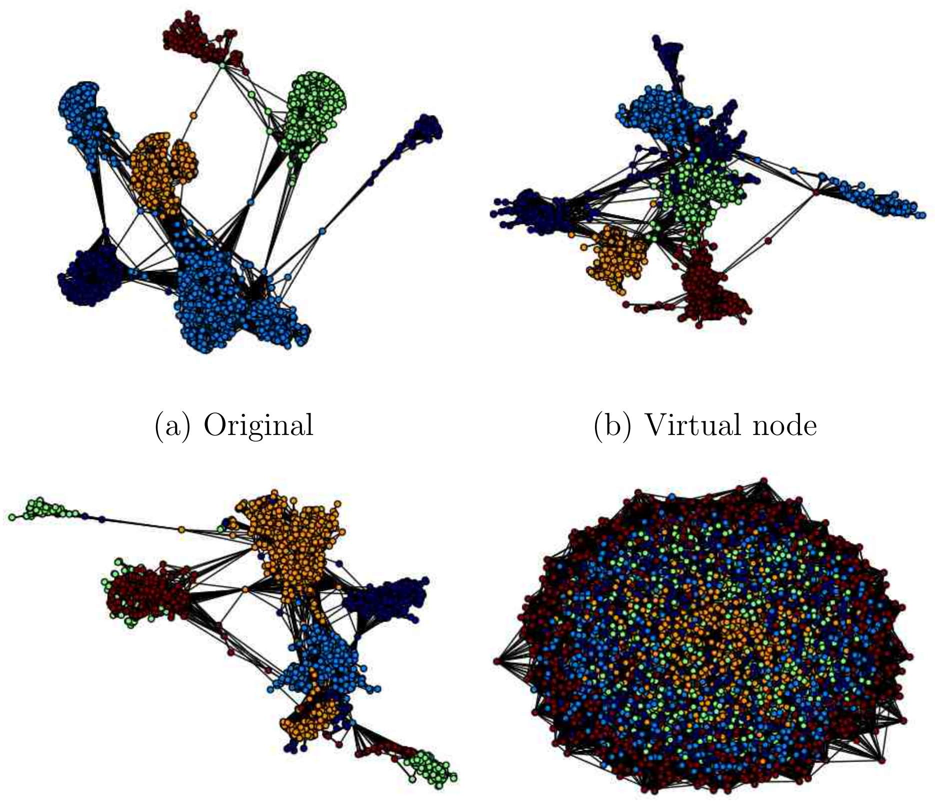


Fig. 3.11.: Clustering coefficient distribution of BA graph,$\epsilon = 0.1$.

Figure 3.10, and 3.11 show the clustering coefficient distribution of Enron and BA graph. The global differential-privacy scheme also reduces the clustering coefficient to a very low level. In the Enron dataset result, the highest clustering coefficient under the global differential-privacy scheme is 0.17, the original dataset and two local differential-privacy methods have 34.2%, 25.3% and 37.8% nodes with clustering coefficient higher than 0.95. These nodes are the critical users in the Enron dataset and our local differential-privacy scheme could preserve some of them.

In order to analyze the difference between the schemes in preserving clustering results, we apply the K-means clustering method to the Facebook dataset and the result is shown in Figure 3.12. The global differential-privacy scheme mixes the clusters all together. On the contrary, our local privacy methods could preserve the clustering information. Because changes happen in 1-hop neighborhood graphs in our scheme, and the cluster size is always bigger than the size of subgraphs, then most of the changes are inside the clusters and does not affect the clustering property. Under the global differential-privacy scheme, nodes may randomly pick the same number of neighbors. While there is no special rules to choose neighbors, there is no clusters and outliers.



(a) Original (b) Virtual node

(c) Outlier elimination (d) Reference

Fig. 3.12.: K-means clustering of Facebook dataset. Four graphs are the original graph and three anonymized graph. K = 5 that all graphs have five clusters. Nodes in the same cluster are in the same color

Shortest-path length. The shortest-path length measures the average length from one node to every other node. Similar shortest path length distribution means similar information transmission time in the two graphs. In order to avoid disconnected part, we choose to measure the maximum connected subgraphs, Figure 3.13, 3.14, and 3.15 show the distribution of average shortest length. The global differential-privacy scheme makes every path to nearly the same length while the local differential-privacy scheme could preserve the information.

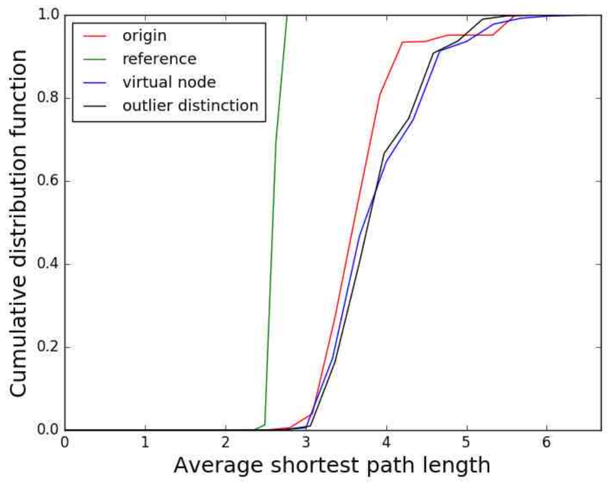


Fig. 3.13.: Shortest-path length distribution of Facebook, $\epsilon = 0.1$

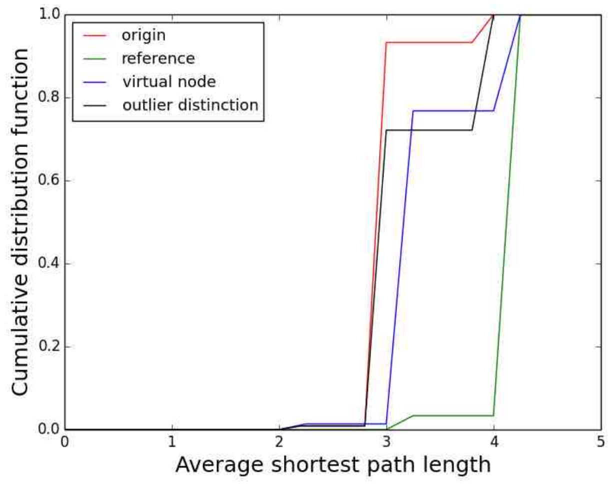


Fig. 3.14.: Shortest-path length distribution of BA graph, $\epsilon = 0.1$

Influence maximization. Influence maximization [25] is an application metric to measure the percentage of users which are influenced by the information diffusion. In the evaluation, the greedy algorithm described in [73] is selected to choose the seed users. Then the independent cascade model is applied to propagate information while the propagation probability is 0.02 for each dataset. Figure 3.16 and 3.17 show

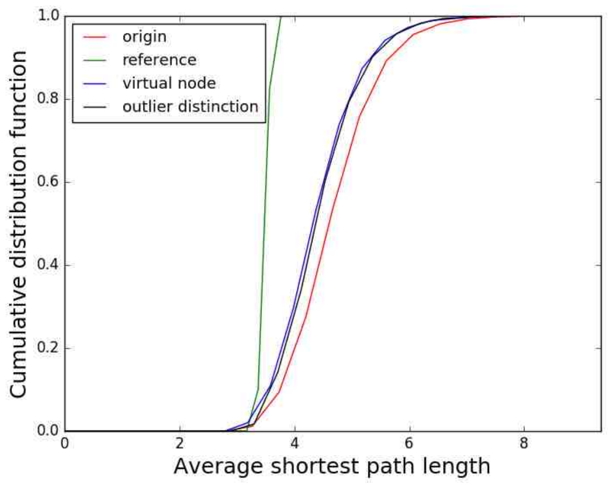


Fig. 3.15.: Shortest-path length distribution of ca-HepPh, $\epsilon = 0.1$

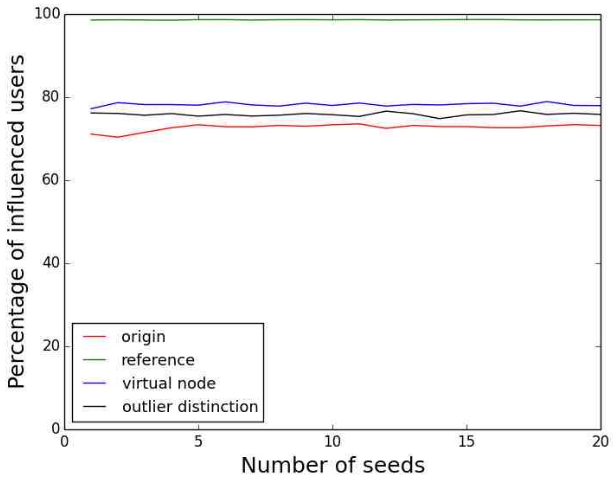


Fig. 3.16.: Percentage of influenced users in Facebook, $\epsilon = 0.1$

the percentage of influenced users with different number of seeds. The anonymized graphs of the global differential-privacy scheme are too easy (Facebook) or too difficult (Enron) to propagate information. The local differential-privacy anonymized graphs have similar properties in influence maximization with the original network.

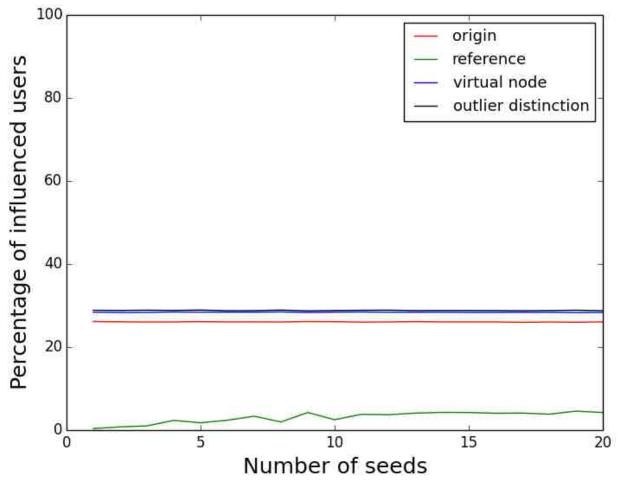


Fig. 3.17.: Percentage of influenced users in Enron, $\epsilon = 0.1$

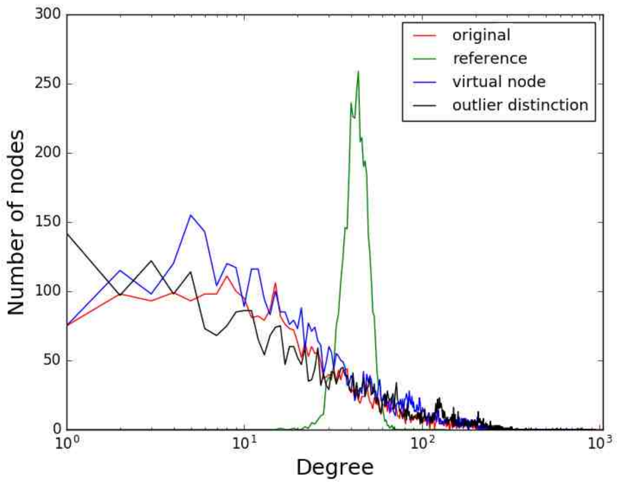


Fig. 3.18.: Degree distribution of Facebook, $\epsilon = 1$

3.3.3 Impact of parameters

In previous researches, the privacy parameter ϵ is 1 in [153] and 5 to 100 in [129], more loose than our setting. Theoretically, higher ϵ means loose privacy that the adversaries may have more confidence in guessing the existence of edges. However,

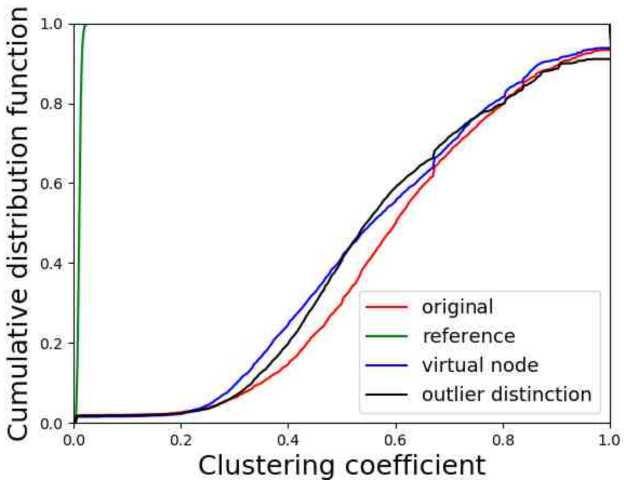


Fig. 3.19.: Clustering coefficient distribution of Facebook, $\epsilon = 1$

higher ϵ may also lead to more utility of the published graph because less noise is introduced. Here we evaluate the utility of the published graph when $\epsilon = 1$. Figure 3.18 and 3.19 show the results.

Compared with the result in Figure 3.8, the degree distribution performs closer to the original distribution when the ϵ is 10 times than before, especially using the local differential-privacy algorithm. When $\epsilon = 0.1$, the degree error, which is a sum of absolute difference of the number of nodes in all degree levels, are 6192, 1818 and 1515 matching with the reference result, 'virtual node' and 'outlier distinction' methods results compared with the original graph. When $\epsilon = 1$, the degree error are 6146, 1421, 1460, respectively.

The local differential-privacy scheme outperforms the global one under the clustering coefficient when $\delta = 1$. The overall average clustering coefficient are 0.60, 0.01, 0.57 and 0.58 corresponding to the original graph, the result of 'virtual node' method and 'outlier distinction' method. And the standard deviation of clustering coefficient is 0.21, 0.003, 0.22 and 0.22. The overall average shortest-path length are 3.69, 2.60, 3.93, 3.68. The standard deviation of shortest-path length is 0.56, 0.04, 0.56, 0.58.

Although local differential-privacy methods achieves better utility with looser privacy criteria, the improvement of global method is not remarkable. It means that the main reason of information loss may be the low probability of each single HRG, so that the regeneration graph has low probability to be similar to the original one. And using local features instead of global ones can significantly reduce the damage of HRG extraction.

We also test different *std* in the 'outlier distinction' method. The higher *std*, the more outliers are grouped. Taking the result of ca-HepPh dataset as the example, the degree error is 3836 when $std = 5$ in Figure 3.9. There are 0.18% nodes which are not grouped. When $std = 3$, the degree error is 3344. There are 0.27% nodes considered as outlier. When $std = 1$, the degree error is 3045. There are 0.36% nodes considered as outlier. It shows that changing *std* modifies the balance between privacy and utility.

3.3.4 Evaluation conclusion

Based on the above three measures, two of the heuristic methods perform well in all three datasets. However, the 'virtual node' method could give a more strict privacy guarantee while the result of 'outlier elimination' method is related with the parameter *std*. Compared with the global differential-privacy algorithm, our local differential-privacy scheme is more robust under the same strict privacy criteria. Although global scheme takes more edges into consideration, in reality, attackers always focus on the direct neighbors as our scheme does. And the evaluation result shows that it is worthy reducing the network scale because it can decrease the noise level, increase single HRG's probability and reduce the computation work.

3.4 Conclusion

In this chapter, we started from the utility preservation problem in existing global differential-privacy criterion. We identified the group-based local differential-privacy criterion and proposed a uniform framework based on HRG models to generate a perturbed social network under that criteria. We adopted a more realistic model, 1-neighborhood graph, to capture the local features and reduce the total amount of noise. Our scheme also contained a grouping algorithm to enhance the privacy level. The experimental study verified that our scheme has less damage to graph utility compared with previous global privacy schemes.

When analyzing the privacy and utility preservation of the proposed scheme, we found that the graph abstraction model playing an important role. As discussed in this chapter, low posterior probability of a HRG tree means regenerating the graph may loss huge amount of information. While this chapter mainly focuses on the HRG model, next chapter will study another graph abstraction model, the dK model. We may continue reduce the unnecessary utility loss in the dK model in Chapter 4.

4. ANONYMIZATION WITH GRAPH ABSTRACTION MODEL - COMBINED DK

As discussed in the last chapter, graph abstraction model is essential in privacy and utility preservation. In this chapter, our analysis shows that choosing inappropriate graph abstraction model may cause unnecessary utility loss. This chapter focuses on the dK model. Comparing with the HRG model discusses before, the dK model preserves degree information extremely well. However, existing anonymization mechanisms, which based on the dK-1 model or the dK-2 model, cannot preserve the clustering information [40, 60].

Therefore, we design and analyze a comprehensive differentially private graph model that combines the dK-1, dK-2, and dK-3 series together. The dK-1 series stores the degree frequency, the dK-2 series adds the joint degree frequency, and the dK-3 series contains the linking information between edges. In our scheme, low dimensional data makes the regeneration process more executable and effective, while high dimensional data preserves additional utility of the graph. As the higher dimensional model is more sensitive to the noise, we carefully design the executing sequence and add three levels of rewiring algorithms to further preserve the structural information. The final releasing graph increases the graph utility under differential privacy.

The major technical contributions of this chapter are the following:

1. We are the first to use the dK-3 model in graph anonymization, which helps to preserve more utility than existing dK models.
2. We combine the dK-3 model with both dK-1 and dK-2 models in sampling and graph regeneration, which mitigates the high sensitivity and complexity in the dK-3 model and makes the design practical.
3. We design two different routes, CAT and LTH, to generate the graph efficiently and effectively, even under the noise impact.

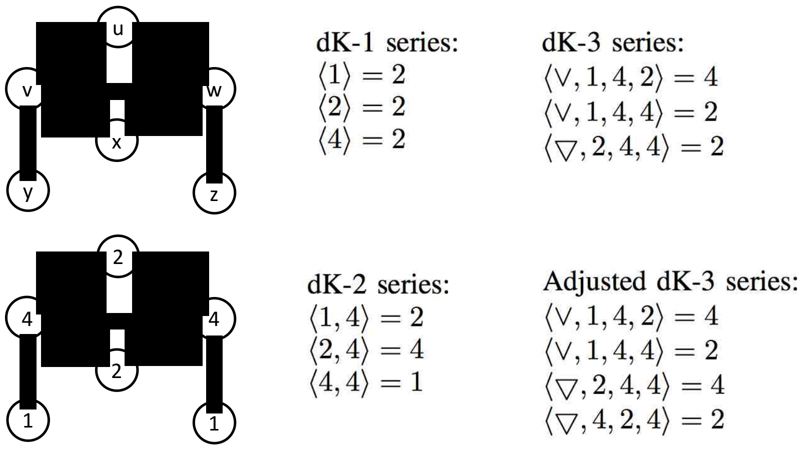

u
v
w
x
y
z
dK-1 series:
$\langle 1\rangle = 2$
$\langle 2\rangle = 2$
$\langle 4\rangle = 2$
dK-3 series:
$\langle \vee, 1, 4, 2\rangle = 4$
$\langle \vee, 1, 4, 4\rangle = 2$
$\langle \triangledown, 2, 4, 4\rangle = 2$
2
4
4
2
1
1
dK-2 series:
$\langle 1, 4\rangle = 2$
$\langle 2, 4\rangle = 4$
$\langle 4, 4\rangle = 1$
Adjusted dK-3 series:
$\langle \vee, 1, 4, 2\rangle = 4$
$\langle \vee, 1, 4, 4\rangle = 2$
$\langle \triangledown, 2, 4, 4\rangle = 4$
$\langle \triangledown, 4, 2, 4\rangle = 2$

Fig. 4.1.: An example of the dK model

4. We use three levels of rewiring algorithms to comprehensively utilize three kinds of dK information in the published graph.
5. We reveal the insights and challenges of using different levels of dK abstraction models jointly to enhance the utility under differential privacy.

This chapter is previously published as a conference paper in IEEE Annual Consumer Communications & Networking Conference (CCNC), 2019 [49].

4.1 Preliminaries

In this chapter, an OSN graph is modeled as an undirected graph $G = (V, E)$. d_v is the degree of the vertex v. $e_{u,v}$ means an edge between nodes u and v.

Since differential privacy is applied on the query result, the dK graph model is chosen to transform an input graph into a set of structural statistics. Although many models can give graph statistical information, the dK graph model is better than most of them because the dK series could be used to construct a new graph having structural similarities with the original graph.

The dK-N model captures the degree distribution of connected components of size N in a target graph [95]. Figure 4.1 gives an example of the dK model. The dK-1 model, also known as the node degree distribution, counts the number of nodes in each degree value. The dK-2 model, also called joint degree distribution, captures the number of edges in each combination of two degree values. In this chapter, we define the dimension of dK information as the subgraph size (N). Hence, the dK-1 series has lower dimension than dK-2. The dK-3 model captures the number of 3-node subgraphs with different combinations of node degrees. Specifically, there are two kinds of 3-node subgraphs with different structures, wedges and triangles.

The wedge dK-3 entry: The dK-3 entry $\langle \vee, d_u, d_v, d_w \rangle = k$ means that there are k 3-node wedges which have the node degree values equal to d_u, d_v, and d_w, and each of the two subgraphs have at least one different node. In order to prevent double counting, d_u should be less than or equal to d_w. Assume the combination of nodes u, v and w forms such a subgraph, then w should not be the neighbor of u. The node set of the subgraph should be $V = \{u\} \cup \{v | e_{u,v} \in E\} \cup \{w | e_{v,w} \in E \wedge e_{u,w} \notin E\}$.

The triangle dK-3 entry: The dK-3 entry $\langle \bigtriangledown, d_u, d_v, d_w \rangle = k$ means that there are k triangles with node degree d_u , d_v, and d_w. To prevent double counting, we have $d_u \leqslant d_v \leqslant d_w$. The node set of the subgraph should be $V = \{u\} \cup \{v | e_{u,v} \in E\} \cup \{w | e_{v,w} \in E \wedge e_{u,w} \in E\}$.

The error between two dK-3 series is defined as the sum of all absolute differences in each corresponding dK-3 entry.

$$err_3 = \sum_{\text{dK-3 } entry} |k_i - k'_i|. \tag{4.1}$$

Similarly, err_1 and err_2 measure the errors in the dK-1 and dK-2 series. And our work focuses on minimizing the error between the dK series in the published graph and the target dK series calculated under differential privacy.

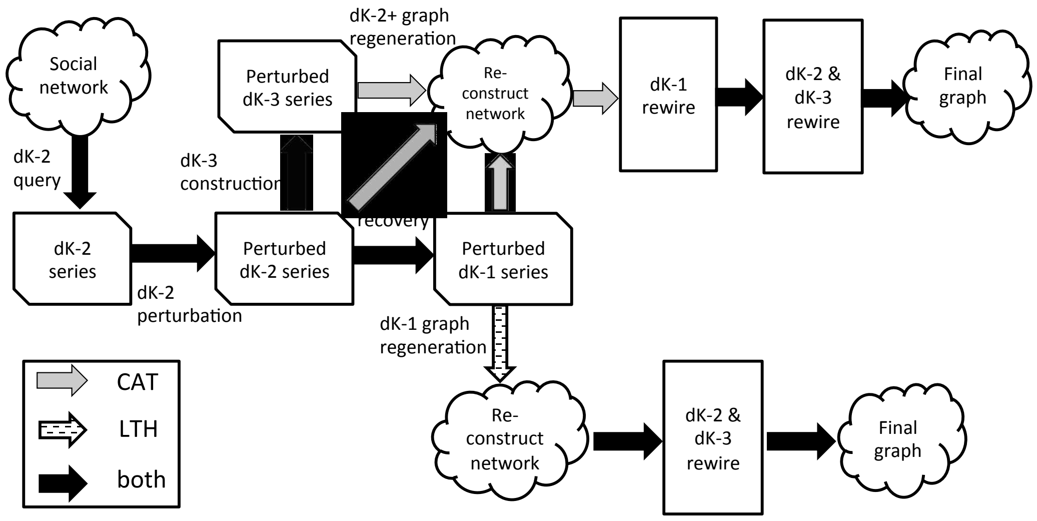


Fig. 4.2.: Scheme overview

4.2 Scheme

Given an OSN, our goal is to publish an anonymized network that preserves the structural utility as much as possible while satisfying ϵ-differential privacy. The general idea is to add sufficient noise to the dK model and reconstruct a graph G based on the perturbed dK series.

As mentioned in previous research, a model of higher dimension is more precise, but it is difficult to directly reconstruct the graph from the dK-3 series [95, 129]. Moreover, our analysis in Section 4.3.1 shows that the dK-3 model is more sensitive than the dK-2 model. Hence, it is not a good idea to start with adding noise to the dK-3 series. Another option is to add noise to the dK-1 series. However, the dK-2 and dK-3 series also need the corresponding perturbation in order to maintain consistency. Because the dK-1 series has no information of edges, it is hard to do those perturbations. In the scheme, we inject noise into the dK-2 series.

After injecting noise, our purpose in the graph regeneration process is to publish a graph with similar dK series as the perturbed results (in all three levels). There are two main routes in graph regeneration, starting with the dK-1 series or starting with the dK-2 series. We design two sub-schemes, shown in Figure 4.2, called CAT and LTH, and the mutual steps are marked in 'both'. LTH uses the dK-1 series to reconstruct an intermediate graph, then we do two steps of rewiring on it. In CAT, we find that when we use the dK-2 series to place edges, there is some freeness in the sequence of placing edges, in which we can invoke the dK-3 series. Hence we call it *consider all together*, which has all three levels of information. As a result, these two sub-schemes are especially good at reducing the dK-1 or dK-2 error. After the regeneration part, both sub-schemes have an active rewiring procedure to mitigate their errors, e.g., the dK-2 and dK-3 series have not been used by LTH.

In the following sections, we discuss these components, which are also shown in Figure 4.2:

1. *dK-2 perturbation:* perturb the dK-2 series under differential privacy,
2. *dK-3 construction:* build the dK-3 model with perturbed dK-2 series,
3. *dK-1 recovery:* recover the dK-1 information,
4. *Graph regeneration:* reconstruct the perturbed graph with different combinations of dK series,
5. *Target rewiring:* rewire some of the edges according to the dK series.

4.2.1 dK-2 perturbation

We find that achieving dK-3 differential privacy needs much more information distortion, which largely reduces the benefits of the dK-3 model after the analysis in Section 4.3.1. What's more, as the dK-2 series is the record of edges, we can make it indistinguishable to achieve edge differential privacy. Hence, we choose to inject noise at the dK-2 level. In particular, after counting the dK-2 entries, we add sufficient Laplace noise to achieve differential privacy. According to Equation 2.3, the noise level is determined by the sensitivity $\triangle f$ and the privacy parameter ϵ.

The sensitivity shows the impact of adding or deleting an edge in the model. For a given entry $\langle d_x, d_y \rangle = k$, the sensitivity is $2 \cdot d_x + 2 \cdot d_y + 1$ (see Section 4.3.1). The perturbed dK-2 entry is $\langle d_x, d_y \rangle = k + Lap(\frac{\triangle f}{\epsilon})$.

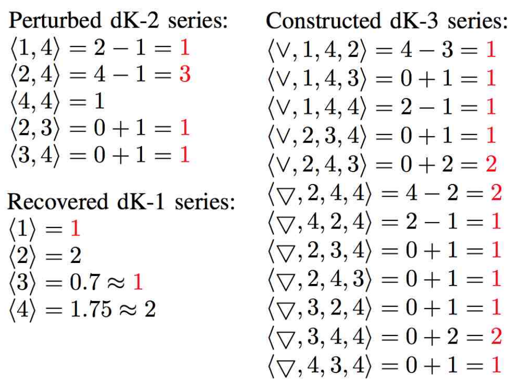
Example. Figure 4.1 shows a running example, which is also used in the following sections. Figure 4.3 has the perturbed dK-2 series. If the value of an entry changes, it is marked in red. We can find that some dK-2 series, like $\langle 2, 3 \rangle$, although not present in the original example (have a value of 0), are created. Because of the differential privacy request, any entries in the range between $\langle 1, 1 \rangle$ and $\langle d_{max}, d_{max} \rangle$ are modified.

4.2.2 dK-3 construction

Given the dK-2 model, we construct the dK-3 model to preserve edge linking information. Particularly, if one dK-2 entry is perturbed, its corresponding dK-3 entries are also perturbed, which leaks no edge information beyond differential privacy. Hence, we examine the influence of dK-2 perturbation on the dK-3 model in the example of one edge $e_{u,v}$, then do the modification.

First, there is a simple case in which all three-node pairs in the graph are wedges. There are $d_u - 1$ edges connected with the node u. Then the edge produces $d_u - 1$ dK-3 entries in the form of $\langle \vee, d_x, d_u, d_v \rangle$ or $\langle \vee, d_v, d_u, d_x \rangle$. Similarly, it also produces $d_v - 1$ dK-3 entries in the form of $\langle \vee, d_y, d_v, d_u \rangle$ or $\langle \vee, d_u, d_v, d_y \rangle$. Hence, there are totally $d_u + d_v - 2$ dK-3 wedges entries produced by the edge $e_{u,v}$.

Second, we improve the case that the graph has some triangles. Adding an edge $e_{u,v}$ between node u and v, if they have a common neighbor x, the original entry $\langle \vee, d_u, d_x, d_v \rangle$ will be changed to $\langle \bigtriangledown, d_u, d_x, d_v \rangle$. However, if they do not have a common neighbor, there will be some new entries added, like the case before. Therefore, the total number of dK-3 entries containing the edge $e_{u,v}$ is also affected by the number of triangles.



Perturbed dK-2 series:

$\langle 1,4\rangle = 2-1 = 1$

$\langle 2,4\rangle = 4-1 = 3$

$\langle 4,4\rangle = 1$

$\langle 2,3\rangle = 0+1 = 1$

$\langle 3,4\rangle = 0+1 = 1$

Recovered dK-1 series:

$\langle 1\rangle = 1$

$\langle 2\rangle = 2$

$\langle 3\rangle = 0.7 \approx 1$

$\langle 4\rangle = 1.75 \approx 2$

Constructed dK-3 series:

$\langle \vee,1,4,2\rangle = 4-3 = 1$

$\langle \vee,1,4,3\rangle = 0+1 = 1$

$\langle \vee,1,4,4\rangle = 2-1 = 1$

$\langle \vee,2,3,4\rangle = 0+1 = 1$

$\langle \vee,2,4,3\rangle = 0+2 = 2$

$\langle \bigtriangledown,2,4,4\rangle = 4-2 = 2$

$\langle \bigtriangledown,4,2,4\rangle = 2-1 = 1$

$\langle \bigtriangledown,2,3,4\rangle = 0+1 = 1$

$\langle \bigtriangledown,2,4,3\rangle = 0+1 = 1$

$\langle \bigtriangledown,3,2,4\rangle = 0+1 = 1$

$\langle \bigtriangledown,3,4,4\rangle = 0+2 = 2$

$\langle \bigtriangledown,4,3,4\rangle = 0+1 = 1$

Fig. 4.3.: Perturbed dK series

Adjusted dK-3 model. We find that if we deploy some specific counting method for triangles, the wedges and triangles can be treated equally. Thus, the adjusted dK-3 model is proposed to simplify the calculation of the dK-3 series. The adjusted model is completely based on the basic dK-3 series. Using the adjusted model will not increase or decrease the ability of the dK-3 series to present or reconstruct the graph. The new model does not change the wedge entry $\langle \vee, d_u, d_v, d_w\rangle$. But if there is a triangle entry $\langle \bigtriangledown, d_u, d_v, d_w\rangle = k$, it will be replaced by three entries, $\langle \bigtriangledown, d_u, d_v, d_w\rangle = k$, $\langle \bigtriangledown, d_v, d_w, d_u\rangle = k$, and $\langle \bigtriangledown, d_w, d_u, d_v\rangle = k$. In the following sections, all dK-3 series are sampled in the adjusted dK-3 model. After deploying the adjusted dK-3 model, deleting or adding an edge $e_{u,v}$ always changes $d_u + d_v - 2$ dK-3 entries. In the following sections, a wildcard character $*$ is used to match $\vee$ and $\bigtriangledown$. The dK-3 entry is like $\langle *, d_u, d_v, d_w\rangle$.

In the above section, the dK-2 series is perturbed for privacy. Each unit of increment or decrement in dK-2 entries could be viewed as one edge adding or deleting. Then we do corresponding modifications on the dK-3 series. Specifically, for increasing or decreasing, there are three possible changes in dK-3 entries.

The first possible change is called replacement. If $\langle d_u, d_v \rangle$ decreased by one and $\langle d_u, d_w \rangle$ increased by one, the graph replaces the edge $e_{u,v}$ by $e_{u,w}$. So we pick $min(d_w, d_v) + d_u - 2$ dK-3 entries and use the number d_w to replace d_v in the dK-3 entries.

The second possible change is subtracting. For each unit of decrement in $\langle d_u, d_v \rangle$, the graph deletes the edge $e_{u,v}$. So we reduce the dK-3 entries containing $\langle d_u, d_v \rangle$ by the total value of $d_u + d_v - 2$.

The third possible change is adding. For each unit of increment in $\langle d_u, d_v \rangle$, the graph adds an edge $e_{u,v}$. The formation part is a little special because there is no original record of the neighbors of u or v. So we randomly pick a structure, wedge or triangle, and a degree number, d_x, in the range of $[1, d_{max}]$. Then we add the total value of $d_u + d_v - 2$ to the dK-3 entries containing $\langle d_u, d_v, d_x \rangle$.

Example. In the example of Figure 4.1, the dK-2 entry $\langle 4, 4 \rangle$ has a total of $4 + 4 - 2 = 6$ corresponding dK-3 entries $\langle \vee, 1, 4, 4 \rangle$ and $\langle \bigtriangledown, 2, 4, 4 \rangle$ in the adjusted model. In Figure 4.3, the corresponding dK-3 series is constructed. Taking the dK-2 entry $\langle 1, 4 \rangle$ as an example, because the dK-2 perturbation causes 1 unit of decrease, the corresponding dK-3 series has $1 + 4 - 2 = 3$ units of decrease. In the constructed dK-3 series, the first three modifications are '-3', '+1', and '-1', while the total amount of decrease is 3.

4.2.3 dK-1 recovery

The dK-1 series is also important in the generation of the graph. Unlike the dK-3 series, it can be recovered directly from the dK-2 series. It is calculated by the following equation.

$$\langle d_v \rangle = \frac{\sum_{\text{dK-2 } entry} \langle d_u, d_v \rangle + \sum_{\text{dK-2 } entry} \langle d_v, d_u \rangle}{d_v}. \tag{4.2}$$

The recovery process shows that the high dimensional data, e.g., dK-2, contains all the information of the low dimensional data, e.g., dK-1.

Example. In the example of Figure 4.3, as the number '4' total appears $1 + 3 + 1 * 2 + 1 = 7$ times in the dK-2 series, the dK-1 series should be $\langle 4 \rangle = 1.75 \approx 2$. Although the perturbed dK-2 values are integers, the recovery dK-1 values may not be integers. Here we can only round the value to integers because these values show the number of nodes and we have no information besides the dK-2 series. And the round-off error causes the two levels of dK series, dK-1 and dK-2, mismatch. In the rewiring section, we discuss the mismatch problem.

4.2.4 Graph regeneration

Given the target dK-2, dK-3, and dK-1 series, we need to regenerate the corresponding graph. Focusing on a different level of dK series, we propose two subschemes, namely CAT and LTH, with different regeneration algorithms.

The LTH scheme starts from the dK-1 series because of the idea that dK-1 series is the base of the graph. If the degree of a node has an error, there will be large distortion on the corresponding dK-2 and dK-3 series. Hence, LTH just needs the dK-1 information and to generate a graph with the least err_1. It leaves the task of mitigating err_2 and err_3 to the rewiring procedure.

By contrast, the CAT scheme considers the dK-2 and dK-3 series in regeneration because rewiring cannot guarantee to achieve the lowest err_2 and err_3. This scheme aims to reduce err_2 the most, while preserving some dK-3 information as well.

In both schemes, we call a node 'saturated' if it has as many neighbors as its label (dK-1 information), and call it 'unsaturated' otherwise. If the value of a dK entry in the graph reaches the target value, we call it 'full'.

LTH Algorithm 4 firstly sorts the degree sequence into a non-increasing order, which means $d_1 \geqslant d_2 \geqslant ... \geqslant d_{|V|}$. Each number in the sequence also represents the target degree value of a corresponding node. Then, beginning from the first node with degree d_1, the algorithm links the node with d_1 nodes. These nodes are chosen from the set of nodes that are unconnected with the first node, and they have the

Algorithm 4 dK-1 graph regeneration (LTH)

```
Input: dK-1
Output: $G_1(V_1, E_1)$: the perturbed graph
 1: $V_1 \leftarrow$ dK-1    ▶ add nodes with degree labels
 2: $\{d_1, d_2, ..., d_{|V|}\} \leftarrow$ dK-1
 3: for $i = 1, i \leqslant |V|, i++$ do
 4:     pick a node $u$ with degree $d_i$
 5:     while $u$ is unsaturated do
 6:         if all nodes are connected with $u$ then break
 7:            ▶ the dK-1 is non-graphical
 8:         pick $v$ with the highest degree among all unsaturated nodes unconnected with $u$
 9:         $E_1$ adds edge $e_{u,v}$
10:     end while
11: end for
12: return $G_1$
```

highest degree values in the set. According to [40], a graph can be reconstructed with the exact dK-1 information if and only if every node v is connected to all d_v nodes in the leftmost part of the degree sequence (having the highest degree values).

CAT In each iteration, Algorithm 5 picks one dK-3 entry and tries to add one edge to the graph. If this algorithm can find two nodes, having corresponding degrees in the dK-3 entry, it can pass the edge check. Here, the edge check means there are two unsaturated nodes with the correct degree, the two nodes are not connected, and the corresponding dK-2 entry is not full. When an edge is added in the graph, its corresponding dK-2 and dK-3 entries are updated. The regeneration process stops when there are no node pairs that can pass the edge check. Also, in the edge check process, it may happen that the only pair of unsaturated nodes are already connected. Simply connecting them together forms multi-edges in the graph, which is forbidden in OSNs. Algorithm 6 switches one neighbor from a saturated node to an unsaturated node with the same label.

Algorithm 5 dK-2+ graph regeneration (CAT)

```
Input: dK-1, dK-2, dK-3
Output: $G_1(V_1, E_1)$: the perturbed graph
1: $V_1 \leftarrow$ dK-1    ▶ add nodes with degree labels
2: dK-2$' \leftarrow 0$, dK-3$' \leftarrow 0$    ▶ initialize the dK-2 and dK-3
3: while exists dK-2 entry not full do
4:     ————————beginning phase————————
5:     randomly pick $\langle *, d_u, d_v, d_w \rangle$ not full in dK-3$'$
6:     if $\langle d_u, d_v \rangle$ not full in dK2$'$ then
7:         if exists $u$ and $v$ unconnected and unsaturated
8:             if $* = \vee$, add edge $e_{u,v}$
9:             if $* = \triangledown$, add edge $e_{u,v}, e_{u,w}$
10:            update dK-2 and dK-3 entries
11:        else if exists $u$ and $v$ connected and unsaturated
12:            ▶ adding edge causes multi-edges
13:            NeighborSwitch$(u, v)$
14:        else    mark $\langle d_u, d_v \rangle$ full, continue
15:            ▶ $\langle d_u, d_v \rangle$ cannot form an edge
16:    else    continue
17:    end if
18:    do Step 6-17, between $v$ and $w$
19:    ———————continuing phase————————
20:    pick $\langle *, d_v, d_w, d_x \rangle$, do Step 6-17 between $w$ and $x$
21:    ...
22: end while
23: return $G_1$
```

There are two phases in which the Algorithm 5 chooses dK-3 entries and adds edges. In the beginning phase, if the node pairs could pass the edge check, we randomly picks a dK-3 entry and adds two or three edges to the graph accordingly. In the continuing phase, we use the last chosen dK-3 entry, denoted as $\langle *, d_u, d_v, d_w \rangle$, to

Algorithm 6 NeighborSwitch(u,v)

```
1: find unsaturated node $v'$ with degree $d_v$, $e_{u,v'} \notin E_1$
2: assume $z$ is a neighbor of $v'$, $e_{z,v'} \in E_1$ and $e_{z,v} \notin E_1$
3: $E_1$ removes edge $e_{z,v'}$, adds edge $e_{z,v}$ and $e_{u,v'}$
4: increase $\langle d_u, d_v \rangle$ in dK-2'
```

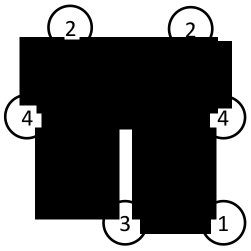


Fig. 4.4.: LTH results

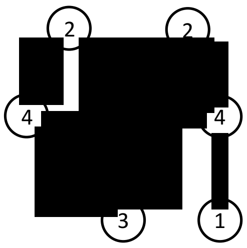


Fig. 4.5.: CAT result

find a new dK-3 series $\langle *, d_v, d_w, d_x \rangle$. Assuming the node w could pass the edge check with another node x, the algorithm links w, which is used in the last step with the new node x. It stops if the newly picked node cannot pass the edge check with any other node, then it jumps to the beginning phase again.

Algorithm 5 makes distinctions between wedges and triangles. It builds triangles if the three users link with each other originally, and forces no edge between u and w if the dK-3 entry is in the form of $\langle \vee, d_u, d_v, d_w \rangle$. The dK-3 information used in Algorithm 5 could preserve more structural information on the triangles and wedges, which is helpful in reconstructing a network with similar clustering information.

Example. Figure 4.4 and Figure 4.5 give the example of regenerated graphs from the perturbed dK series in Figure 4.3. When the numbers on nodes represent the request degree, the LTH result satisfies the dK-1 series. However, it has no dK-2 information, e.g., $\langle 2, 4 \rangle = 4$ in the graph, but the desired value is 3. Compared with the given dK series in Figure 4.3, we have $err_1 = 0$, $err_2 = 4$, $err_3 = 18$.

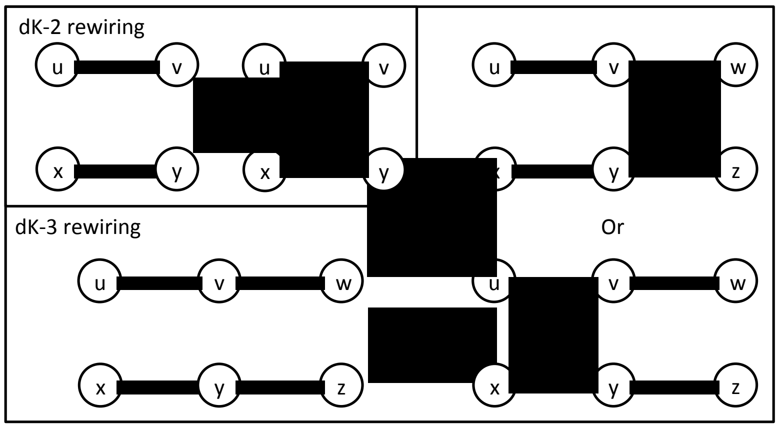


Fig. 4.6.: Examples of rewiring

By contrast, the CAT result seems to satisfy the dK-2 requirement perfectly. However, one node with mark '3' and one with '4' do not have the required degree, and all the dK-2 series are exhausted. err_1 happens because of mismatch, and has a impact on err_2 and err_3. Compared with the given dK series, we have $err_1 = 2$, $err_2 = 4$, and $err_3 = 13$. Comparing the two results of the example, each sub-scheme has an advantage in preserving information.

4.2.5 Target rewiring

As mentioned in the last section, there is no dK-2 and dK-3 information preserved in the LTH intermediate graphs. LTH needs to compare the graph with the target dK-2 and dK-3 series and apply rewiring. Intuitively, the CAT intermediate graph only needs to apply the dK-3 rewiring because it does not consider the dK-3 entries in their entirety. However, after analyzing the noise impact in Section 4.3.2, we find that the result that satisfies the dK-2 series may have non-trivial error in dK-1 information. As a result, the CAT scheme needs dK-1, dK-2, and dK-3 rewiring, from

Algorithm 7 Target rewiring

```
Input: dK-1, dK-2, dK-3, $G_1$
Output: $G_2(V_2, E_2)$: a new graph
 1: $G_2$=$G_1$
 2: ——————————dK-1 rewiring——————————
 3: $u$, $v$ unsaturated and unconnected, $E_2$ adds edge $e_{u,v}$
 4: $u$, $v$ unsaturated and connected, NeighborSwitch$(u, v)$
 5: $u$ needs two or more edges, NeighborSwitch$(u, u)$
 6: ——————————dK-2 rewiring——————————
 7: dK-2$'$ $\leftarrow G_2$    ▶ count the dK-2 in dK-1 rewired graph
 8: while there exists dK-2 rewiring pairs do
 9:     $E_2$ removes $e_{u,v}, e_{x,y}$, adds $e_{u,y}, e_{v,x}$
10: end while
11: ——————————dK-3 rewiring——————————
12: dK-3$^0$ $\leftarrow G_2$    ▶ count the dK-3 in dK-2 rewired graph
13: $err_3^0$ $\leftarrow$dK-3$^0-$ dK-3    ▶ store the initial error
14: $i = 0$    ▶ the step number
15: while there exists dK-3 rewiring pairs do
16:     $E_2^{i+1}$ removes $e_{v,w}, e_{y,z}$, adds $e_{v,z}, e_{y,w}$
17:     get new dK-3$^{i+1}$ and $err_3^{i+1}$
18:     if $err_3^{i+1} \geqslant err_3^i$, reject the rewiring, $G_2^{i+1} = G_2^i$
19:     $i = i + 1$
20:     do the rewiring check, Step 11-14, between $e_{u,v}, e_{x,y}$
21: end while
22: return $G_2$
```

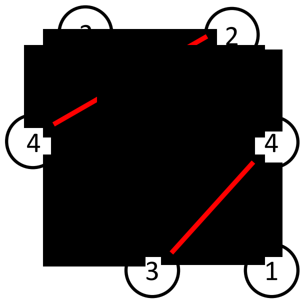
lower to higher. Here we propose three levels of dK rewiring algorithms: each level of the rewiring preserves the lower dimensional information, but may change the higher dimensional information.

dK-1 rewiring. Given G_1 as the input, we build edges between pairs of unsaturated nodes. Building each edge reduces err_1 by two. There are two special cases in the dK-1 rewiring shown in Algorithm 7. First, there are just two unsaturated nodes and they are already linked. A neighbor switch process should be applied on these two nodes. Second, there is just one node unsaturated, but it needs at least two edges. Then the neighbor switch should also be applied on this node. Here the neighbor switch process in dK-1 rewiring is slightly different from the one in Algorithm 6. It has no limitation on the degree of v'; it just needs v' and u to be unconnected.

dK-2 rewiring. In this step, err_2 is reduced while keeping the result from the first step. Figure 4.6 shows the dK-2 rewiring process described in Algorithm 7. The dK-2 series in the intermediate graph is compared with the target dK-2. The algorithm applies the rewiring procedure if the prerequisites are satisfied. We define the dK-2 rewiring prerequisites such that $\langle d_v, d_w \rangle$ and $\langle d_y, d_z \rangle$ are higher than the target, and $\langle d_v, d_z \rangle$ and $\langle d_y, d_w \rangle$ are lower than the target. When at least three out of four prerequisites are satisfied, we admit a rewiring pair to reduce err_2 by at least two.

dK-3 rewiring. err_3 is reduced with a similar procedure. Figure 4.6 shows two kinds of rewiring on the same six nodes. It is notable that the two different solutions lead to the same direct dK-3 changes, which is denoted as the direct impact of dK-3 rewiring. However, the rewiring process may also have indirect impact on dK-3, e.g., some entries involving nodes u and v are also changed. Hence, in each iteration, Algorithm 7 calculates the dK-3 series of the new graph called dK-3^i and finds the dK-3 rewiring pairs. We admit a step of dK-3 rewiring only if the dK-3 error is decreased.

Numerically, in the example of Figure 4.6, the rewiring changes the dK-3 series directly but keeps the dK-2 unchanged if and only if $d_u \neq d_x$, $d_v = d_y$ and $d_w \neq d_z$. Hence, we define the dK-3 rewiring prerequisites as $\langle *, d_u, d_v, d_w \rangle$ and $\langle *, d_x, d_v, d_z \rangle$ are higher than the target, and $\langle *, d_u, d_v, d_z \rangle$ and $\langle *, d_x, d_v, d_w \rangle$ are lower than the target. We also admit the pair when at least three requirements are satisfied. Here



dK-1 series before rewiring:
$\langle 1\rangle = 1$
$\langle 2\rangle = 3$
$\langle 3\rangle = 1$
$\langle 4\rangle = 1$
dK-1 series after rewiring:
$\langle 1\rangle = 1$
$\langle 2\rangle = 2$
$\langle 3\rangle = 1$
$\langle 4\rangle = 2$

Fig. 4.7.: dK-1 rewiring

the rewiring can directly reduce err_3 by at least two. Structurally, the two types of dK-3 series have additional prerequisites on the existence of edges. For example, if the value of the entry $\langle\bigtriangledown, d_u, d_v, d_z\rangle$ is lower than the target value, there should be one edge between u and z before rewiring, then rewiring builds a triangle automatically.

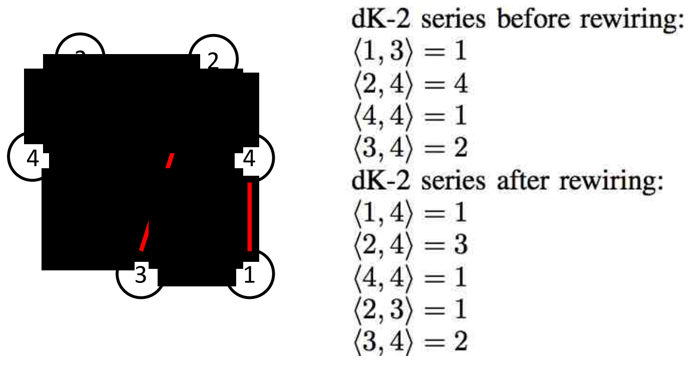
Example. Figure 4.7 shows the example of dK-1 rewiring when the original graph is in Figure 4.5. When the original graph has two unsaturated nodes but the two nodes are linked, a neighbor switch process involving the right node with mark '2' can help all nodes satisfy the dK-1 series.

Figure 4.8 shows the example of dK-2 rewiring when the original graph is in Figure 4.4. In the simple example, Figure 4.7 and Figure 4.8, are the same graph which shows that the CAT result after dK-1 rewiring can get the same graph as the LTH result after dK-2 rewiring. Both graphs have $err_2 = 1$ and $err_3 = 2$, which shows that the rewiring algorithms can significantly reduce the error in the dK series.

4.3 Analysis

4.3.1 Sensitivity analysis

The sensitivity shows the impact of adding or deleting an edge in the dK-2 model.



dK-2 series before rewiring:
$\langle 1,3\rangle = 1$
$\langle 2,4\rangle = 4$
$\langle 4,4\rangle = 1$
$\langle 3,4\rangle = 2$
dK-2 series after rewiring:
$\langle 1,4\rangle = 1$
$\langle 2,4\rangle = 3$
$\langle 4,4\rangle = 1$
$\langle 2,3\rangle = 1$
$\langle 3,4\rangle = 2$

Fig. 4.8.: dK-2 rewiring

Theorem 3. *Given an entry $\langle d_x, d_y\rangle$ in the dK-2 model, the sensitivity $\triangle f$ is upper bounded by $2 \cdot d_x + 2 \cdot d_y + 1$.*

Proof: Let $e_{x,y}$ be a new edge added to the graph G between nodes x and y. There is one new dK-2 series $\langle d_x, d_y\rangle$ getting incremented by 1. Also, the degrees of x and y increase from d_x and d_y to $d_x + 1$ and $d_y + 1$, respectively. In the original dK-2 model, there are d_x series related with the node x. They are in the form of $\langle d_u, d_x\rangle$ and $\langle d_x, d_u\rangle$. They are deleted and new series, $\langle d_u, d_x+1\rangle$ and $\langle d_x+1, d_u\rangle$, are added. Hence, totally $2 \cdot d_x + 2 \cdot d_y + 1$ dK-2 series are changed when adding the edge. □

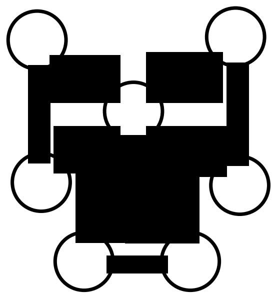
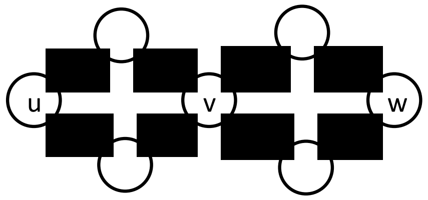
Similarly, given the dK-3 series $\langle *, d_x, d_v, d_z\rangle$, the sensitivity of the adjusted model is $2 \cdot (d_y + d_z) \cdot d_{max} + (d_y + d_z)$, where d_{max} is the max degree value in the graph.

4.3.2 Performance analysis

In this section, we analyze the noise impact on dK graph models and then show the ability of our schemes to reduce dK error under noise.

If the dK series is graphical, which means it can build a graph, it must obey the following rules:

1. The values of dK entries being non-negative integers.

(a) target graph (b) regeneration result

Fig. 4.9.: Example of the dK-1 rewiring

2. The dK-1 information, if in non-increasing degree sequence form, following the Erdös-Gallai theorem [44],

$$\sum_{i=0}^{j} d_i \leqslant j(j+1) + \sum_{i=j+1}^{|V|-1} min\{j+1, d_i\}. \tag{4.3}$$

3. The dK-2 entries having $\langle d_x, d_y \rangle \leqslant \langle d_x \rangle \cdot \langle d_y \rangle$,
$\langle d_x, d_y \rangle \leqslant d_x \cdot \langle d_x \rangle$, $\langle d_x, d_y \rangle \leqslant d_y \cdot \langle d_y \rangle$
, if $d_x = d_y$, and $\langle d_x, d_x \rangle \leqslant \langle d_x \rangle^2 - \langle d_x \rangle$.
4. The dK-3 entries having $\langle *, d_x, d_y, d_z \rangle \leqslant \langle d_x, d_y \rangle \cdot \langle d_y, d_z \rangle$,
$\langle *, d_x, d_y, d_z \rangle \leqslant d_y \cdot \langle d_x, d_y \rangle$, $\langle *, d_x, d_y, d_z \rangle \leqslant d_y \cdot \langle d_y, d_z \rangle$
, if $d_x = d_z$, and $\langle *, d_x, d_y, d_x \rangle \leqslant \langle d_x, d_y \rangle^2 - \langle d_x, d_y \rangle$

In most of the real cases, the perturbed dK-2 and dK-3 series are non-graphical, so we need to fix the dK-2 and dK-1 values to non-negative integers. However, the approximation makes the dK series conflict with one another. For instance, if the degree value 3 appears 4 times in the dK-2 series, we can just make $\langle 3 \rangle = \frac{4}{3} \approx 1$. Hence, we introduce the rewiring algorithms and apply the approximation graphically from lower to higher.

When considering the ability of reducing err_1, our LTH scheme has such a property.

Property 1. *The dK-1 regeneration result (LTH) has less err_1 than the dK-2+ regeneration result (CAT), although CAT has the dK-1 rewiring algorithm.*

Proof: Because the dK-1 regeneration algorithm is directed by the Erdös-Gallai theorem [44], it always builds a graph when the dK-1 information is graphical. If the dK-1 information is not graphical, the dK-1 regeneration algorithm adds possible links to high-degree nodes as much as possible, which does not form a forbidden link enlarging err_1 [40].

We also show the shortage of the CAT scheme in an example. Assuming the target dK-1 information is $\langle 6 \rangle = 1$, $\langle 2 \rangle = 6$, Figure 4(a) is a correct result of the dK-1 regeneration algorithm. Although Figure 4(b) violates the dK-2 series of the graph, it is still a possible intermediate graph published by the dK-2+ regeneration algorithm. Then the dK-1 rewiring algorithm cannot add neighbors to unsaturated node v. Although the nodes u and w are the possible candidates, the neighbor switch process is impossible because all neighbors of them are linked with v, which is the only unsaturated node in the graph. Hence, the dK-1 rewiring cannot get the correct graph, while the LTH scheme can. □

As shown in the example, the ability of the dK-1 rewiring algorithm is limited. Also, our analysis shows that the dK-2 and dK-3 rewiring algorithms have the possibility of trapping in a local area when searching for the global minimum. The detailed analysis is omitted for space. Here the term local minimum is defined as there is no neighbor graph (with one edge changing) having lower error than the rewiring result. The rewiring pairs reduce the error by two or four in dK-2 and dK-3 rewiring algorithms. Considering some particular pairs may trap the error in the local area.

In conclusion, all three kinds of rewiring algorithms have the possibility of trapping in a local area when searching for the global minimum. It is significant to choose a start graph before rewiring. LTH starts from a graph with the best dK-1, the most

Table 4.1.: Network dataset statistics

Dataset	# of nodes	# of edges
ca-HepTh	9877	25998
Facebook	4039	88234
Enron	2977	7198

basic information. CAT starts from a graph with some dK-3 information, which restricts the level of err_3. Our two schemes use two routes to deal with the noise and the conflict problem. Each of them has its own advantages in reducing the error.

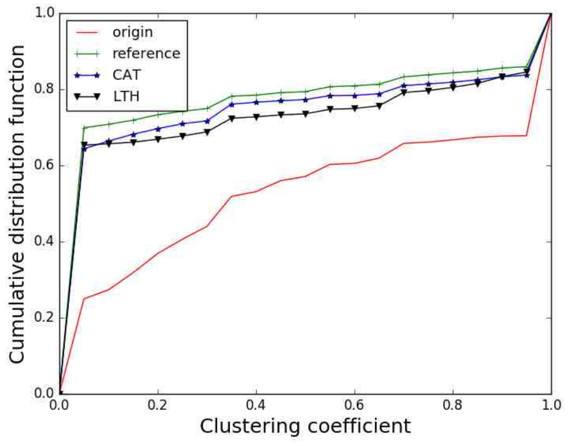
4.4 Evaluation

In this section, we evaluate our anonymization scheme over three real-world datasets, namely ca-HepTh, Facebook, and Enron [88].

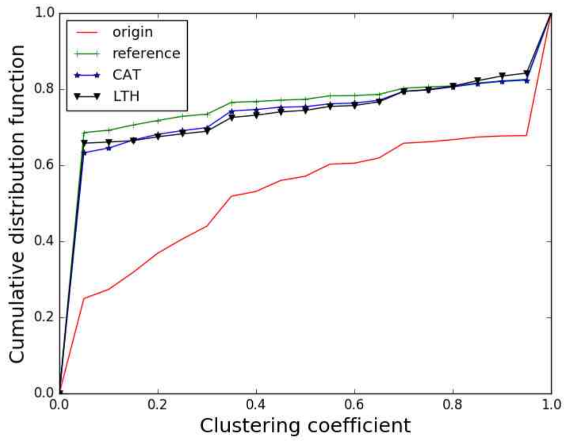
ϵ is a privacy parameter to measure the ability of hiding existing edges. Smaller ϵ means a more strict privacy guarantee, as well as more noise injected into the model. We generate ϵ-private graphs with $\epsilon \in [5, 100]$ to evaluate the performance under different different-privacy levels. For comparison purposes, we implement one state-of-the-art technique as the reference method, which is the differential privacy algorithm, with only the dK-2 model [129, 148]. In the following figures, results of this scheme are marked as 'reference', two of our sub-schemes are marked as 'CAT' and 'LTH', respectively.

To show the utility of the released networks, we apply experiments on the robust measures of network topology: the average shortest path length, the clustering coefficient, and the average degree. We also evaluate the three levels of errors.

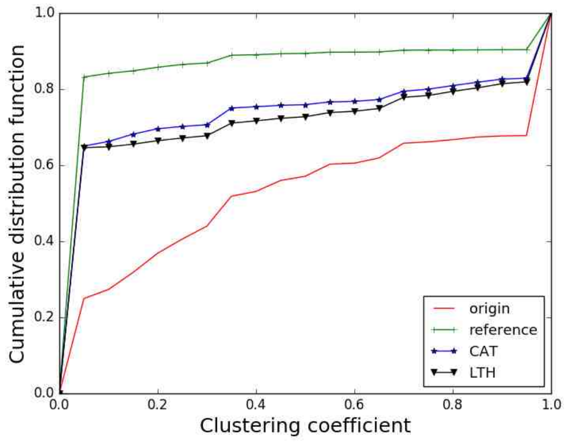
Clustering coefficient. Clustering coefficient is a measure of how nodes in a graph tend to cluster together. While the dK-2 model may break the features of a cluster, a scheme with the dK-3 series is believed to preserve partial clustering



(a) ca-HepTh, ϵ=5

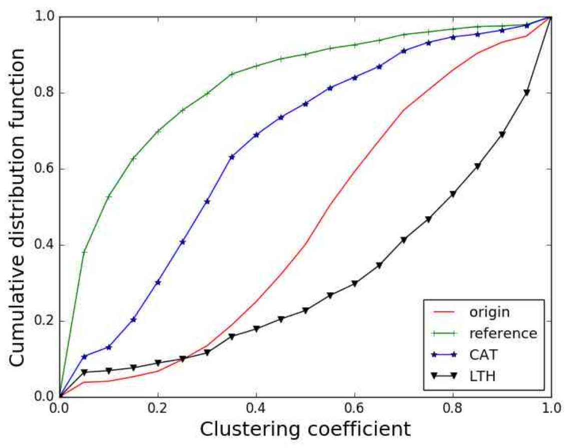


(b) ca-HepTh, ϵ=20

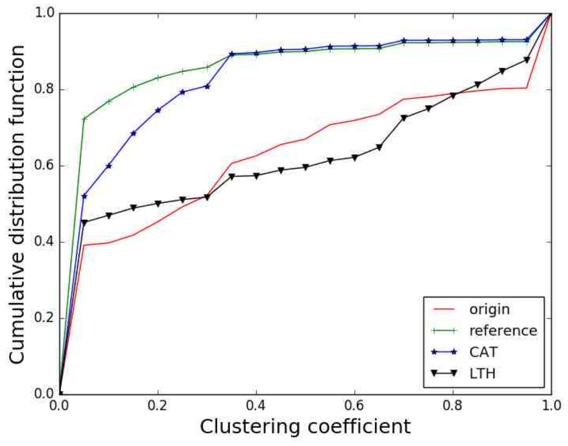


(c) ca-HepTh, ϵ=100

Fig. 4.10.: Clustering coefficient distribution under different ϵ



(a) Facebook, ϵ=20



(b) Enron, ϵ=20

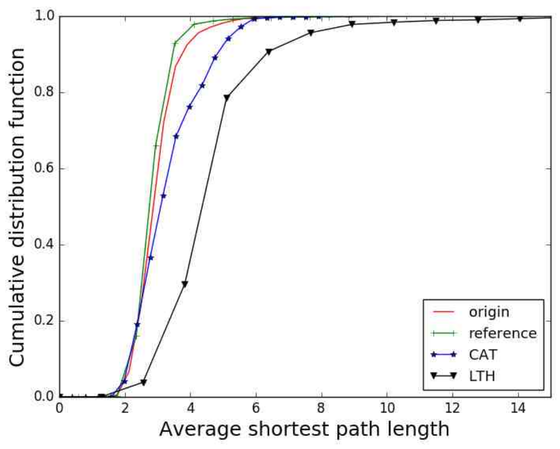
Fig. 4.11.: Clustering coefficient distribution in different datasets

information because structural information like the triangles and wedges are included. Figure 4.10 shows the clustering coefficient distribution under different ϵ. In the original ca-HepTh graph, 28% of nodes have the median clustering coefficient (0.2 to 0.8). However, when ϵ=20, this kind of node only occupies the 9% of total nodes in the reference result, 12% in the CAT graph, and 13% in the LTH graph. The three dK anonymization methods all lose some clustering information.

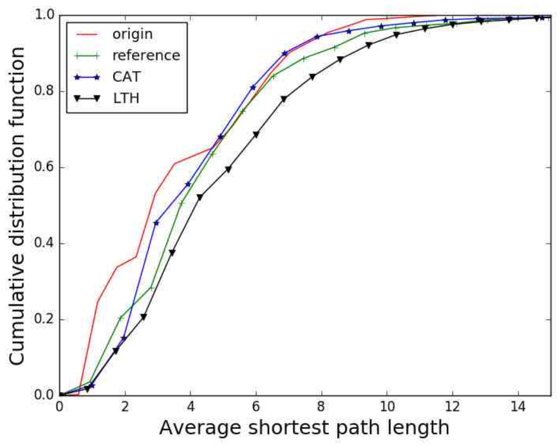
The original ca-HepTh dataset has an average clustering coefficient of 0.47. When $\epsilon = 5$, the average clustering coefficient of the reference result is 0.21, and it's 0.24 for CAT and 0.26 for LTH. When $\epsilon = 100$, the average clustering coefficient is 0.12, 0.25, and 0.27 for the reference, CAT, and LTH result, respectively. The figures show that the clustering coefficient distribution of our two schemes are always closer to the original distribution than the reference result. The dK-3 series in our scheme preserves the structure information of triangles and wedges, which determines the clustering coefficient. Hence, the reference scheme shows more randomness, while our schemes can preserve more clustering information.

Average shortest path length. The average shortest path length measures the average length of the shortest path from one node to every other node. Figure 4.12 shows the average shortest length distribution in three datasets when $\epsilon = 20$. Take the result of the Enron dataset as an example: the overall average shortest path length of the original data is 3.61, and the reference, CAT, and LTH schemes have the result 4.20, 3.54, and 11.35, respectively. The figure shows that the reference scheme and the CAT scheme can preserve the shortest path length information well in all three datasets. However, the nodes in the LTH anonymized graph have a longer distance between each other than the nodes in the original graph.

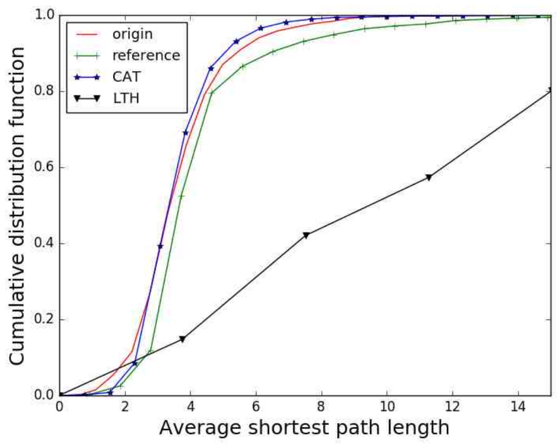
As shown in the analysis in Section 4.3.2, the dK-2 rewiring algorithm cannot help LTH to minimize err_2. The LTH scheme just uses the dK-1 information in graph regeneration, and the probability of high degree nodes linking with each other is much higher than the original. We can make an observation that the shortest path length is closely related with the dK-2 series.



(a) Facebook, ϵ=20



(b) ca-HepTh, ϵ=20



(c) Enron, ϵ=20

Fig. 4.12.: Shortest path length distribution in different datasets

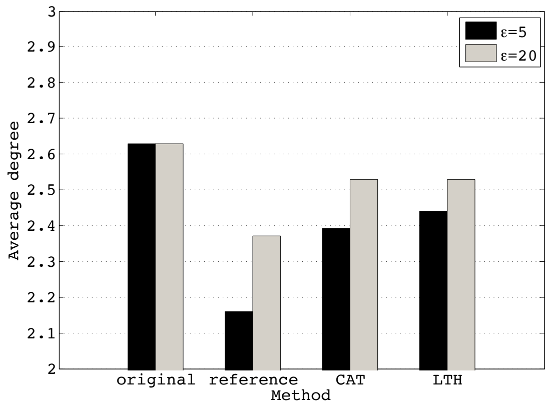


Fig. 4.13.: Average degree, ca-HepTh

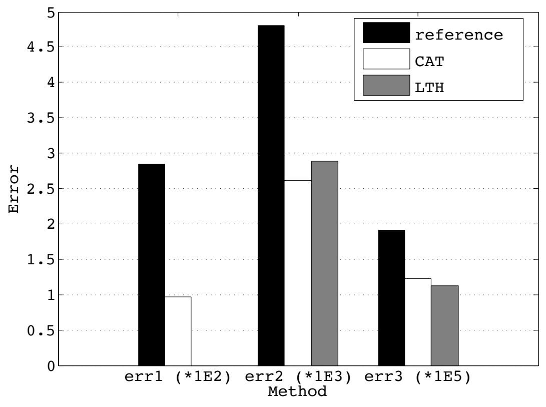


Fig. 4.14.: Error, Facebook

Degree. The degree of a node in the network is the number of edges incident to the node. Figure 4.13 shows the average degree in the ca-HepPh dataset. The published graph is more similar to the original when noise level is lower in all three schemes. When $\epsilon = 5$, the average degree difference of the reference result is 0.37, while CAT has a difference of 0.24, and LTH has a difference of 0.19. Compared with the reference method, the CAT scheme has an additional dK-1 rewiring algorithm, which effectively reduces err_1, so the CAT result is better than the reference result. The LTH scheme achieves the best result in reducing err_1. In the regeneration part, it just uses the dK-1 information while other schemes begin with the dK-2 series. Here we can make the observation that LTH has better performance than CAT even though CAT has the dK-1 rewiring algorithm, which is consistent with Property 1.

Error. We also compare the dK series in the original graph with different regeneration results. Figure 4.14 gives the three levels of error in the Facebook dataset. The reference result has 284 units of err_1, while the CAT result has 96, and the LTH result has no err_1. It shows that the dK-1 rewiring algorithm can reduce a large amount of err_1, but using dK-1 in graph regeneration is better. Reducing err_1 also helps the CAT result, which has smaller err_2 (2.6K) than the reference result (4.8K). Because of the cumulative error in the dK series and no dK-3 rewiring algorithm, the reference result has 0.19M err_3, while CAT has 0.12M and LTH has 0.11M. The closer dK-3 distance between our results and the original graph is a reason that our schemes can preserve more structural information in the published graph.

Conclusion. From the above experiments, we can conclude that the CAT and LTH schemes perform better in most measurements than the reference scheme. The LTH scheme can better preserve degree information but it lacks the ability to preserve average shortest path information. The CAT scheme generally produces better results in all other graph metrics we evaluated.

4.5 Conclusion

In this chapter, we propose a uniform scheme that combines three levels of dK graph models to publish a perturbed social network. We design two different sub-schemes, CAT and LTH, and three levels of rewiring algorithms to regenerate the graph and reduce the error under the differential privacy noise impact. The empirical study indicates that our two schemes have different merits in preserving graph utility. The design, analysis, and comparison also reveal more insights and challenges in using multiple levels of graph abstraction models together in differential private graph releasing for OSNs.

This chapter and Chapter 3 achieve differential privacy based on the two graph abstraction models, dK and HRG. The two chapters aim to reduce the unnecessary and unbearable utility loss in graph anonymization. The proposed schemes achieve a good utility and privacy balance under existing measurement. However, existing measurement of utility, i.e., the graph utility metrics, have some limitations. Next chapter may focus on these limitations and introduce a new measurement of utility.

5. ANONYMIZATION WITH UTILITY METRIC - PERSISTENT HOMOLOGY

In the previous two chapters, the proposed anonymization schemes are evaluated with some traditional graph utility metrics, i.e., degree distribution, clustering coefficient, and shortest path length. However, the true utility of the published graphs is questionable for two reasons: First, the chosen metrics are limited by the graph abstraction models. Previous studies have shown that none of the mechanisms have good performance under all the metrics [48]. Second, existing metrics only describe the graph in a certain angle. For example, while the degree distribution and the clustering coefficient disjointedly reveal the graph utility in two specific aspects, each aspect does not cover the other. Thus, lots of useful graph information gets lost or distorted during the graph anonymization process, especially when the anonymization mechanisms are based on these types of graph metrics.

This chapter introduces a comprehensive utility metric called persistent homology. We propose a novel anonymization scheme, called PHDP, which preserves persistent homology and satisfies differential privacy. To strengthen privacy protection, we add exponential noise to the adjacency matrix and find the number of adding/deleting edges. To maintain persistent homology, we collect edges along persistent structures and avoid perturbation on these edges. Our regeneration algorithms balance persistent homology with differential privacy, publishing an anonymized graph with a guarantee of both. Evaluation result show that the PHDP-anonymized graph achieves high graph utility, both in graph metrics and application metrics.

The major technical contributions of this chapter are the following:

1. We introduce a novel utility metric, persistent homology, in the analysis of OSNs.

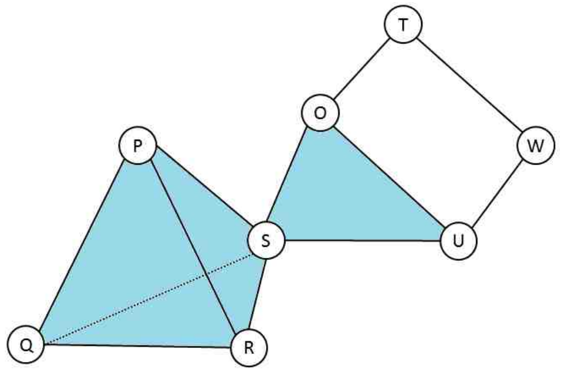


Fig. 5.1.: An example of the simplicial complex

2. We propose the PHDP scheme to balance differential privacy and persistent homology in graph anonymization.
3. We evaluate the PHDP scheme with two real-world datasets and comparing it with other anonymization schemes.

This chapter is previously published as a conference paper in IEEE Conference on Computer Communications (INFOCOM), 2019 [50].

5.1 Preliminaries

Persistent homology is a utility metric that summarizes the graph in multi-scales. Persistent homology is presented in the form of barcodes, which have two parts. The Vietoris-Rips (VR) simplicial complex describes the structural change at different spatial resolutions in one dimension, while the Betti number describes the dimensions [20].

VR simplicial complex. Persistent homology is based on the simplicial complex. A simplicial complex set K contains points, line segments, triangles and high-dimension components. K satisfies the following conditions:

1. Any face of a simplex from K is in K, where the face of K_n is the convex hull of the non-empty subset of the $n+1$ points, which define K_n.

2. The intersection of any two simplices, $\sigma_1, \sigma_2 \in K$, is either $\emptyset$ or a face of both σ_1 and σ_2.

In a simplicial k-complex, the highest dimension of simplices is k . For instance, the 1-complex is the line segment, the 2-complex is the convex hull of the triangle and the 3-complex is the convex hull of the tetrahedron.

The VR complex is one of the abstract models of the simplicial complex. It introduces the distance parameter δ, and then forms the simplicial complex set K, such that for all node pairs $(v_i, v_j) \in K$, the distance between v_i and v_j is less than or equal to δ.

Figure 5.1 shows an example with one 3-complex, one 2-complex and some 1-complexes. The node set {O,U,T,W} has no 2-complex because the pairwise distance within O-W and T-U both are above δ.

Barcode. While the VR complex is defined on the specific δ, persistent homology chooses various δ and gives an increasing sequence of VR complexes.

$$K_0 \subseteq K_1 \subseteq ... \subseteq K_n = K \tag{5.1}$$

Persistent homology collects the features in a wide range of distance and gives a comprehensive description of the structure.

Through applying Betti numbers, the persistent homology overcomes the restriction of dimension. The Betti number $Betti_n$ gives the number of (n+1)-dimensional holes. Particularly, $Betti_0$ is the number of connected components, $Betti_1$ is the number of holes and $Betti_2$ is the number of voids.

In n dimension, the vector space of n-holes is represented by H_n, which can be calculated by the n-cycles and n-boundaries [165]. This calculation gives the Betti intervals to describe the homology of H_n. These intervals are called barcode, where each interval means a component or a hole in the corresponding dimension [58]. For example, the H_1 interval $[1, 2)$ in Figure 5.3 is related with the H_1 hole {P, Q, S, R}. The intervals begin with the δ the holes born. And they end with the δ the holes die.

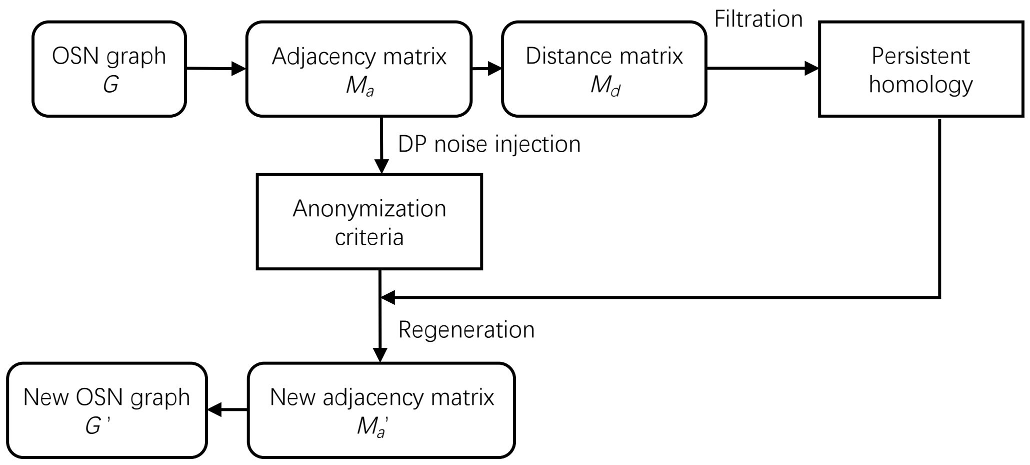


Fig. 5.2.: Scheme overview

The intervals show the birth time and death time of the components. In conclusion, the barcode collects the information of the existing periods of all components and holes when changing the distance δ.

In OSNs, a 2-D hole is a polygon with at least 4 sides. Because there is no ideal circle in the OSN, polygons are appropriate for the common definition of holes, which has a circular boundary and the inside is empty. The high-dimensional holes are the voids, which have triangles as surface and an empty interior. Further discussion of high-dimensional holes are given in Section 5.2.3. A polygon with at least 4 sides implies that all nodes on the polygon have at least one node which is not directly connected, while the triangles have all nodes pair-wisely connected. When we increase the δ, there are more triangles in the network. However, increasing the δ can both build and destroy holes. Hence, we can analyze the persistence of the hole structures and open a novel angle in graph anonymization.

5.2 Scheme

Given an OSN G, our goal is to publish an anonymized network G' that maximally preserves persistent homology while satisfying ϵ-differential privacy. The general idea of the PHDP scheme is to preserve the persistent structures in the anonymized graph. Figure 5.2 shows the structure of the scheme. Section 7.2.1 describes how the OSN graph is modeled as an adjacency matrix and the corresponding distance matrix. Section 5.2.2 describes the division of the adjacency matrix into four types depending on if the corresponding subgraph has holes. Afterwards, the PHDP scheme applies an MCMC procedure to output the number of flips required to achieve differential privacy. Section 5.2.3 describes the application of different regeneration algorithms to the varying types of sub-matrices in order to preserve existing holes and prevent the creation of new ones.

Anonymizing user identity without perturbing the graph structural information leaves the OSN vulnerable to potential de-anonymization attacks [76]. Hence, the PHDP scheme includes both the naive ID removal and the differential-privacy topological anonymization. Since the published graph G' contains no identity information, the original vertex label is trivial. Only the graph topology information is applied with anonymization and utility preservation.

5.2.1 System model

The PHDP scheme employs the adjacency matrix model. Compared to the other graph abstraction models, e.g., dK-2 and HRG, the adjacency matrix model has the least sensitivity (Δf=1). Because the differential privacy noise is proportional to sensitivity, the resulting adjacency model has the least distortion. Another reason for choosing the adjacency matrix model is that it can be easily transformed to a distance matrix—the input for barcode extraction.

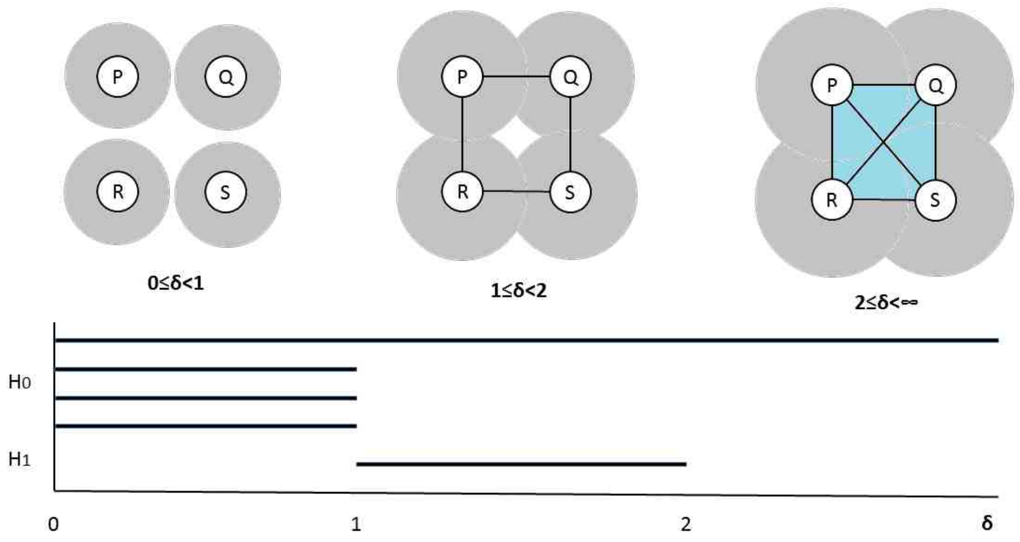


Fig. 5.3.: An example of the barcode

To link the adjacency matrix with the distance matrix, we must first establish the definition of distance. Given an unweighted OSN graph G, the most direct definition of distance δ is the length of the shortest path between a pair of users. Following that definition, persistent homology is captured based on the distance matrix. Figure 5.3 gives an example of the barcode. In the original network, four nodes P, Q, S, R form a square. When $\delta < 1$, there are no edges in the graph. Each node is a component in H_0, so there are four bars in $[0, 1)$. When $\delta \geqslant 1$, the nodes are connected together to form a component and this component exists until the end. So there is one bar of H_0 in $[1, \infty)$. When $\delta < 2$, the node pairs P-S and R-Q are not connected. Then the four nodes form a hole in 2-dimension. So there is one bar of H_1 in $[1, 2)$.

Obviously, under this definition of distance, there is an equivalent relationship among the graph topology, the adjacency matrix M_a and the distance matrix M_d. For example, we can use M_a to build an isomorphic graph for graph G. We can also traversing the edges to generate the distance matrix. Because the user identities are removed in the final graph G', the PHDP scheme chooses the adjacency matrix model.

The process of capturing persistent homology is a filtration in M_d. Taking Figure 5.3 as the example, we have,

$$M_a = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \qquad M_d = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 2 & 1 \\ \hline 1 & 0 & 1 & 2 \\ \hline 2 & 1 & 0 & 1 \\ \hline 1 & 2 & 1 & 0 \\ \hline \end{array}$$

Equation 5.1 suggests that filtration is also employed among three adjacency matrices with $\delta = 0, 1, 2$. And $\delta = 0$ is omitted because M_a is a zero matrix and $K_0 = \emptyset$.

$$\left\{ K_1 \in \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array} \right\} \subseteq \left\{ K_2 \in \begin{array}{|c|c|c|c|}\hline 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline \end{array} \right\} = K$$

The example shows that given δ, the new graph connects all the node pairs with distance less than or equal to δ in the original graph. While the general barcodes show the filtration of persistent structures, the barcodes in OSNs show the filtration of relationships among distances. Various δ can be viewed as different definitions of relationships. For example, the relationship definition expands from direct friendship to both direct friendship and having mutual friends when δ changes from 1 to 2. Under various relationship scales, barcodes reveal the persistent structures in the network.

5.2.2 Anonymization

With the adjacency matrix M_a, Algorithm 8 has two phases for anonymization. The first phase is dividing: M_a splits into sub-matrices according to the barcode. The second phase is noise injection: the number of flips of 0s and 1s in the sub-matrix are calculated based on the differential-privacy criteria. Then in the regeneration sub-scheme, the position of the 1s preserves the persistent homology.

Dividing. In the dividing phase, the nodes are divided into different groups according to the barcodes they involved. The input of the dividing algorithm is the whole graph and the corresponding adjacency matrix, while the output is a node sequence, placing nodes in the same group adjacent to each other.

In order to preserve the H_0 bars, we need to locate the connected components and extract the corresponding nodes. When $\delta = 0$, each node in the graph is a component. The number of H_0 bars equals the number of nodes, which is trivial because it equals the size of the adjacency matrix. When $\delta \geqslant 1$, the number of H_0 bars equals to the number of disconnected subgraphs. For preserving these H_0 bars, the adjacency matrix should have a node label sequence that groups the nodes according to the subgraphs they belong to.

In order to protect the H_1 and H_2 bars, we need to locate the holes in the network. The nodes involved in each bar are extracted based on the Morse Theory [102]. In particular, for each 2-D and 3-D hole, the 2-D boundaries and 3-D boundaries are extracted. The nodes belonging to the same boundary, i.e., the same hole, are grouped together. If the original graph has disconnected subgraphs, each hole is contained in a single subgraph. Hence, grouping them together does not violate the previous grouping result. Figure 5.1, for example, has one subgraph (itself) and one hole. The new node sequence can be {{O, T, U, W}, {P, S, Q, R}}.

The two steps of grouping give a new vertex label sequence and the corresponding adjacency matrix M_a. Then M_a is divided into four kinds of sub-matrices according to the node groups of barcodes.

- $M0$, which only contains 0, shows there are 0 edges between two disconnected subgraphs.
- $M1$, whose nodes are extracted from H_1 and H_2 bars, shows the edges within a hole component (in Figure 5.1, the nodes {O, T, U, W}).
- $M2$, whose nodes are not extracted from barcode, shows the edges in a subgraph without holes (in Figure 5.1, the nodes {P, S, Q, R}).

Algorithm 8 Anonymization algorithm

```
Input: Adjacency matrix $M_a$, privacy budget $\epsilon$.
Output: Number of flips $f0$ and $f1$ in each sub-matrix
1: Get the barcodes, group nodes correspond with each bar.
2: Get a sequence of node labels, rebuild $M_a$ with the sequence.
3: Divide $M_a$ into sub-matrices according to groups.
4: for each sub-matrix in $M_a$ do
5:     Apply the MCMC procedure, get the distribution of $(f0, f1)$.
6:     Apply the exponential mechanism, sample $(f0, f1) = (a, b)$ with probability
       $\exp\left(\frac{\epsilon \cdot p(a,b)}{2}\right)$.
7: end for
8: return $f0$, $f1$ for each sub-matrix.
```

- $M3$ shows the edges between the nodes from $M1$ and $M2$ matrices (in Figure 5.1, the nodes {O, U, S}).

From the description, $M1$ and $M2$ are matrices on the diagonal; $M0$ and $M3$ are not on the diagonal. Figure 5.4 shows the M_a corresponding to a simple graph with two disconnected subgraphs. One subgraph has one H_1 bar (in $M1_1$) while the other has two H_1 bars (in $M1_2$ and $M1_3$).

Noise injection. In this phase, each sub-matrix is perturbed to satisfy the differential-privacy criteria. The perturbed matrix M_a' is graphic, meaning it can regenerate a graph, if and only if M_a' has the following three properties: First, M_a' only contains 0s and 1s. Second, M_a' is symmetric. Third, the values on the diagonal of M_a' are all 0, because self-loop is not allowed in OSNs.

Hence, only one of two symmetric matrices needs the anonymization. For instance, in Figure 5.4, the algorithm perturbs $M3_1$ then copies it to $M3_3$. When the sub-matrix is on the diagonal, only the upper triangle of the matrix is perturbed. The row number and column number of a matrix is represented by h and w. Then to a

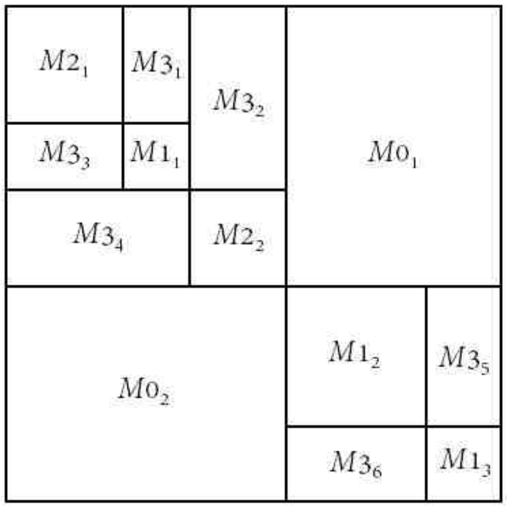


Fig. 5.4.: An example of dividing the M_a

sub-matrix not on the diagonal, the effective size $S = h \cdot w$. To a sub-matrix on the diagonal, the effective size S is the size of the upper triangle (except the diagonal), which is $\frac{h^2-h}{2}$.

Because the basic step of graph anonymization is edge addition or deletion when the differential privacy is defined on edges, we use two numbers $f0$ and $f1$ to model the anonymization process. $f0$ shows the number of 0s flipping to 1s, and $f1$ shows the number of 1s flipping to 0s. It requires the data structure, i.e., the adjacency matrix, to store the 0s and 1s.

In order to achieve ϵ-differential privacy, we apply the exponential mechanism to the adjacency matrix of the graph. However, the exponential mechanism requires the natural distribution of $(f0, f1)$, then it calculates the two numbers f_0 and f_1. Thus, we employ a Markov Chain Monte Carlo (MCMC) procedure to obtain the approximate natural distribution. MCMC is a class of algorithms for sampling from a probability distribution [59]. After the Markov chain reaches its stationary distribution, the subsequently visited states of the chain can be used to simulate the natural distribution.

In our work, the states of the Markov chain are adjacency matrices and the neighbor states are two adjacency matrices with one number difference, i.e., one edge adding or deleting. Particularly, beginning from the original sub-matrix, each step of the Markov chain has the following sub-steps: (1) Uniformly and randomly choose one out of S numbers in the adjacency matrix. (2) Flip that number, i.e., 0 changes to 1 or 1 changes to 0, and build the new adjacency matrix. (3) Compare the new matrix with the very beginning matrix to get the numbers of flips $f0$ and $f1$; record the two numbers. (4) Go to the next MCMC step.

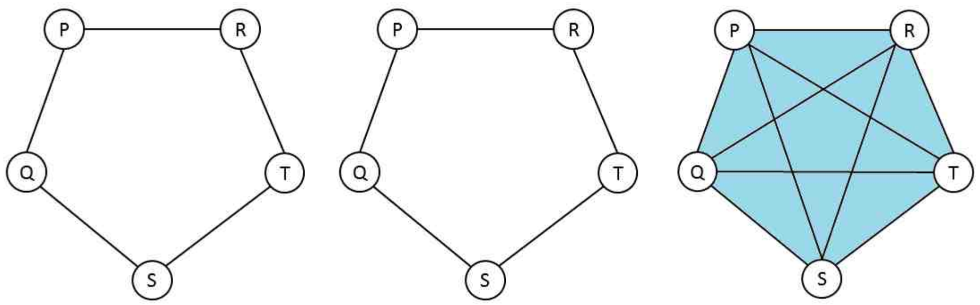
Applying the MCMC procedure described above from zero to a large step size, we can find the approximate distribution of $(f0, f1)$. Particularly, the probability of a target pair $(f0, f1) = (a, b)$ is denoted as $p(a, b)$, which is the number of times $f0 = a$ and $f1 = b$ divided by the total number of steps of the Markov chain.

Finally, the exponential mechanism is embedded with the MCMC procedure to satisfy differential privacy. Specifically, the Markov chain is the same. Instead of the unperturbed probability $p(a, b)$, the algorithm samples $(f0, f1) = (a, b)$ with perturbed probability $\exp\left(\frac{\epsilon \cdot p(a,b)}{2\Delta f}\right)$, where $\Delta f = 1$.

5.2.3 Regeneration

The regeneration sub-scheme designs algorithms to choose the 1s and 0s to flip, which preserves the persistent homology as well as satisfies the requests of $f0$ and $f1$. Although all sub-matrices have the corresponding flipping numbers, the four kinds of sub-matrix $M0, M1, M2, M3$ have different regeneration algorithms. For the matrices representing the barcodes, i.e., $M0$ and $M1$, we need to preserve the structures in it. For the matrices not directly representing the barcodes, i.e., $M2$ and $M3$, we have more freedom to change edges.

M0. In order to preserve H_0, the $M0$ matrix has a strict restriction that all values in it are 0. Although the $M0$ matrix does not produce any $f1$, the regenerated matrix cannot consume any $f0$ or $f1$ either. Similarly, the $f0$ and $f1$ consumption in $M1$

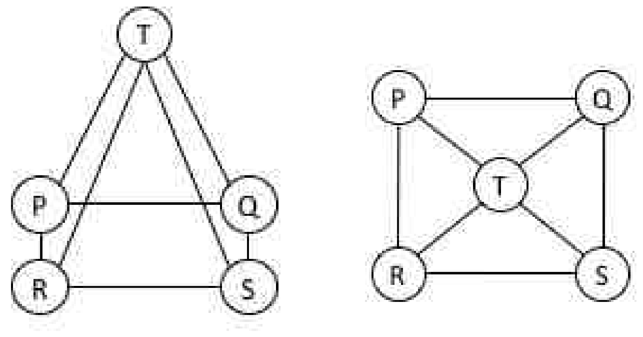


(a) Original graph (b) δ=1 (c) δ=2

Fig. 5.5.: Example of 2-D hole in H_1

matrices is also restricted because the holes need to be preserved. However, directly transferring $f0$ and $f1$ violates the differential privacy requirement. For instance, two matrices form a super-matrix. Because $p(a_1, b_1) + p(a_2, b_2) \neq p(a_1 + a_2, b_1 + b_2)$ (i.e., the sum of the two distributions of $(f0, f1)$ in the two matrices is not equal to the distribution in the super-matrix) the privacy is broken.

The PHDP scheme employs a procedure called conquer-and-divide to 'transfer' $f0$ and $f1$. Unlike other divide-and-conquer procedures, our procedure combines the matrices together, calculates the $(f0, f1)$ in the super-matrix, then divides the super-matrix, and distributes the $(f0, f1)$. Particularly, the procedure has three steps: (1) An $M0$ or $M1$ matrix combines with an $M2$ or $M3$ matrix to form a super-matrix. The two matrices individually calculate their own effective size, S, and the number of 1s, x. The S of the super-matrix is the sum of the two S of the two matrices as well as the x. (2) The super-matrix deploys the MCMC procedure in Section 5.2.2 to calculate $(f0, f1)$. (3) $f0$ and $f1$ are distributed to the two matrices. If all $f0$ and $f1$ can be consumed, the procedure ends. When the other matrix cannot consume any $f0$ or $f1$, the super-matrix chooses another super-matrix, $M2$ matrix, or $M3$ matrix, and returns to the first step. In the worst case, all sub-matrices are combined together. When the adjacent matrix with noise is graphic, the conquer-and-divide procedure can find a solution to assign the $f0$ and $f1$.



(a) Folding result 1 (b) Folding result 2

Fig. 5.6.: Example of none 2-D hole in H_1

M1. The $M1$ matrices are related with $H1$ and $H2$ bars. According to the definition of persistent homology, the barcode in H_n shows the (n+1)-dimensional holes. Figure 5.3 and Figure 5.5 give examples of 2-dimensional holes, which both have a $[1, 2)$ bar in H_1. Their distance matrices are,

$$M_d(4) = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 2 & 1 \\ \hline 1 & 0 & 1 & 2 \\ \hline 2 & 1 & 0 & 1 \\ \hline 1 & 2 & 1 & 0 \\ \hline \end{array} \qquad M_d(5) = \begin{array}{|c|c|c|c|c|}\hline 0 & 1 & 2 & 2 & 1 \\ \hline 1 & 0 & 1 & 2 & 2 \\ \hline 2 & 1 & 0 & 1 & 2 \\ \hline 2 & 2 & 1 & 0 & 1 \\ \hline 1 & 2 & 2 & 1 & 0 \\ \hline \end{array} \tag{5.2}$$

$M_d(4)$ is a square. $M_d(5)$ is a pentagon. The distance matrices suggest that the necessary condition of a 2-D hole (H_1) existing is that δ is less than the maximum value in M_d.

When $\delta = 0$, no edges are formed. Therefore, the birth time of H_1 bars is no less than 1. The 2-D hole is defined as a polygon with at least 4 sides. In OSNs, since the holes are in the form of polygons, we use polygons as the basic hole structure to analysis barcodes.

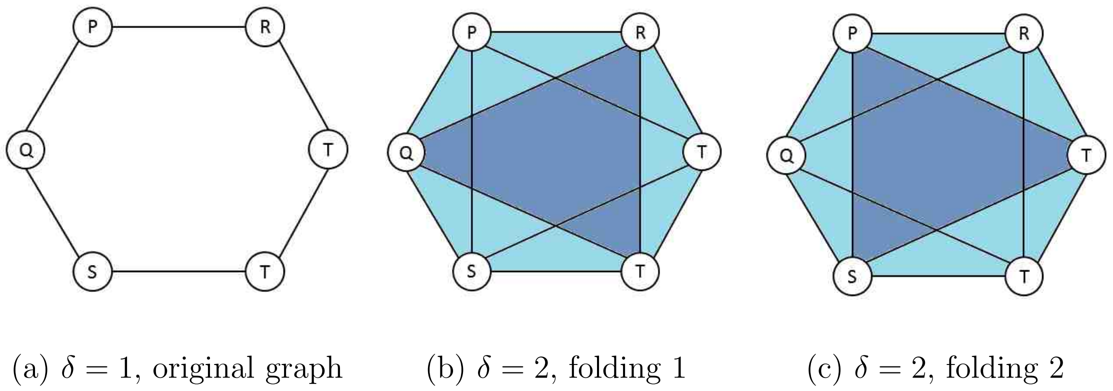
Table 5.1.: Barcodes of polygons

n-sided	barcode (higher than H_0)	$\lceil \frac{n}{3} \rceil$	$\lfloor \frac{n}{2} \rfloor$
n=4	$[1, 2)$ in H_1	2	2
n=5	$[1, 2)$ in H_1	2	2
n=6	$[1, 2)$ in H_1, $[2, 3)$ in H_2	2	3
n=7	$[1, 3)$ in H_1	3	3
n=8	$[1, 3)$ in H_1, $[3, 4)$ in H_3	3	4
n=9	$[1, 3)$ in H_1, $[3, 4)$ in H_2	3	4

Unlike other data structures with fixed positions for each node, OSNs only define the relationships between nodes. Consequently, OSNs have the possibility called folding. In a 2-D view of the OSN, a node can be put inside or outside a hole with different folding results, which influences the existence of the hole. Taking Figure 5.6 as the example, in the first result, the square $\{P, Q, S, R\}$ is a 2-D hole. However, in the second result, there are no holes because all the components are triangles. Considering all the folding results, there are no H_1 bars in the barcode of Figure 5.6. The necessary and sufficient condition of H_1 bar existing is that under a specific δ, in all folding results, there is at least one area which is not filled with a triangle.

The persistent homology also has the ability to capture high-dimensional holes. For example, a sphere with the surface and a void is a simple 3-D hole. When it folds to 2-D, there should be two layers of surface overlapping with each other. A 3-D hole is inferred from the two layers of surface and there is an H_2 bar.

The hexagon in Figure 5.7 is another example. The maximum distance is 3 in M_d. When $\delta = 2$, there are two folding results with the surface filled by triangles. Thus the H_1 bar dies and the H_2 bar exists when $\delta = 2$. And its barcode is in Table 5.1.



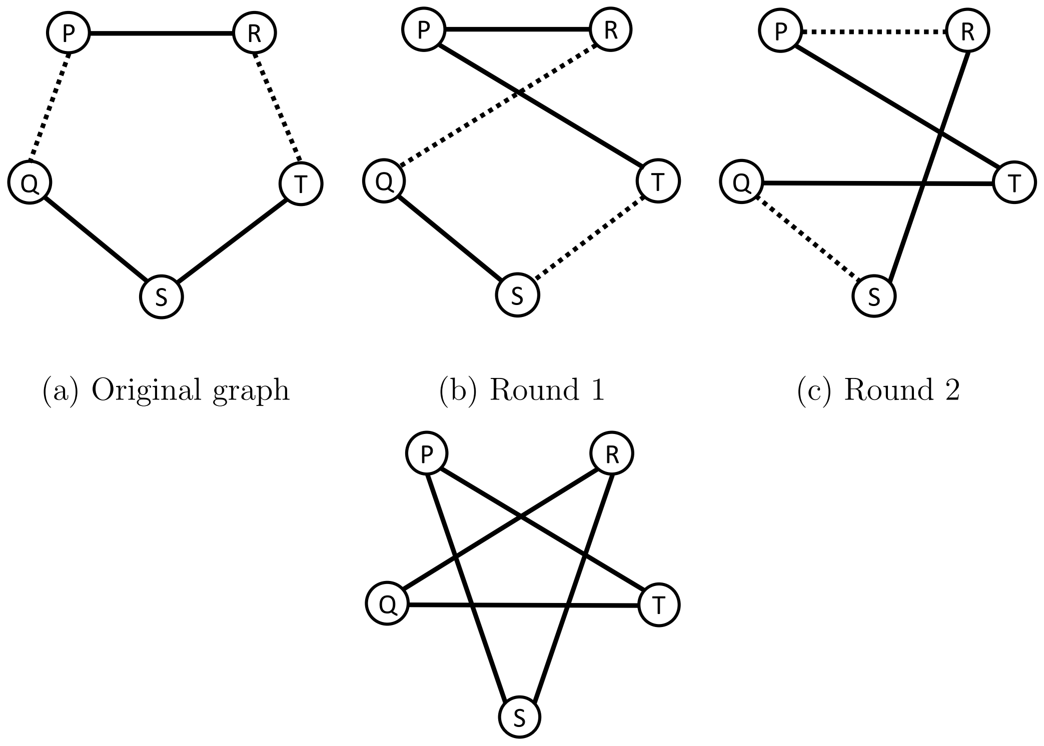
(a) $\delta = 1$, original graph (b) $\delta = 2$, folding 1 (c) $\delta = 2$, folding 2

Fig. 5.7.: Example of 3-D hole in H_2

We do experiment on the barcodes of polygons, as shown in Table 5.1. Because polygons having more than 7 sides rarely exist in OSNs, the H_0, H_1 and H_2 bars are suitable to represent the persistent structures. Having the mapping between persistent structures and the barcodes, we can generate $M1$ matrices without changing the barcodes.

Similar to $M0$, the regeneration of the $M1$ matrices also has the restriction that no edge is added or deleted. A conquer-and-divide procedure is also taken by the $M1$ matrix if the $f0$ or $f1$ is not exhausted. However, edge exchanging makes it possible to reduce $f0$ and $f1$ at the same time inside the $M1$ matrices. Figure 5.8 shows an example of edge exchanging.

The exchanging procedure has the following steps: (1) Choose two edges without mutual nodes, like the dotted lines shown in Figure 5.8. (2) Switch one node from the first edge with one node from the second edge, e.g., replacing the edge P-Q, R-T with the edge P-T, Q-R. (3) Calculate the effective exchanging number, which is the number of deleted edges existing in the original graph reduced by the number of added edges existing in the original graph. In Figure 5.8, it is two in Round 1 and Round 3 but one in Round 2. If the effective exchanging number is negative, the algorithm will forbid that exchange and return to the first step. (4) Decrease both $f0$ and $f1$ by the effective exchanging number. (5) Continue to do the first four steps



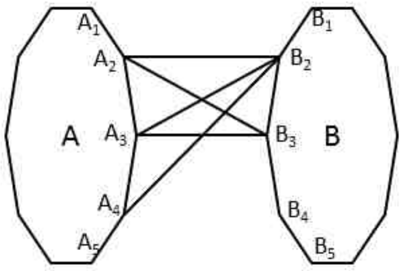
(a) Original graph (b) Round 1 (c) Round 2

(d) Round 3

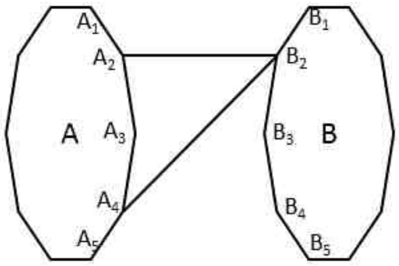
Fig. 5.8.: Example of edge exchanging steps

until $\min(f0, f1) = 0$ or all possible exchanges have been tried. Finally, generate a new adjacency $M1$ matrix and use the other parts in the super-matrix to consume the remaining $f0$ and $f1$.

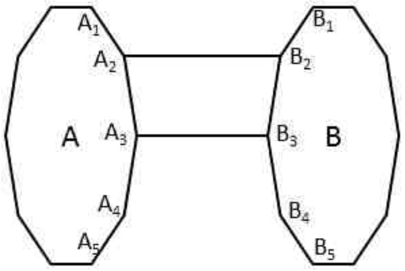
Observing Table 5.1, we also make a hypothesis about the high dimensional holes. For a polygon with n sides ($n > 3$), it has an H_1 bar $[1, \lceil \frac{n}{3} \rceil)$. When $\lceil \frac{n}{3} \rceil < \lfloor \frac{n}{2} \rfloor$, it has at least one bar $[\lceil \frac{n}{3} \rceil, \lfloor \frac{n}{2} \rfloor)$ in high dimension (higher than H_1). Several properties related to the hypothesis are analyzed in Section 5.3.



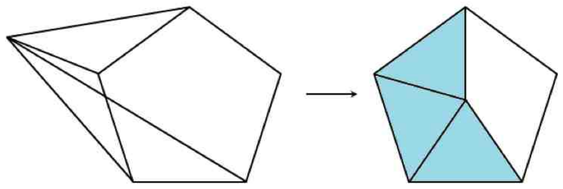
(a) Basic example



(b) Counter example 1



(c) Counter example 2



(d) Counter example 3

Fig. 5.9.: Examples of edges in $M3$ matrices

M3. The purpose of regenerating the $M3$ matrices is to avoid creating new holes while adding or deleting edges. As shown in Figure 5.9(a), the $M3$ matrices capture the edges between two structures, denoted by A and B. The nodes in A that connect to B form the set A^*. The nodes in B that connect to the node A_i forms the set A_i^*. In the basic example, $A^* = \{A_2, A_3, A_4\}$, $A_2^* = \{B_2, B_3\}$.

When the regeneration step successfully preserves the barcodes, the M_3 matrices and corresponding edges should obey the following rules:

1. The nodes in A^* should be adjacent to each other, i.e., $\forall A_i \in A^*, \exists A_j \in A^*, (A_i, A_j) \in E$.
2. Sorting the sequence of A_i^* according to size in non-decreasing order $A_i^*, A_j^*, ..., A_k^*$, the sequence should have $A_i^* \subseteq A_j^* \subseteq ... \subseteq A_k^*$.

3. The structure belonging to M_1 should have at most three nodes to connect to the other structure. For instance, if A is a hole, then $|A^*| \leqslant 3$, where $|A^*|$ shows the cardinality of the set A^*.

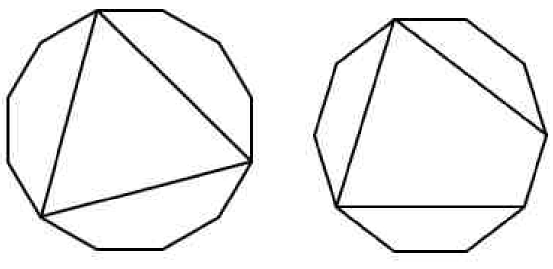
The examples of violating these three rules are shown in Figure 5.9(b), 5.9(c), and 5.9(d), respectively. When the nodes in B^* are not adjacent, they simply create polygons with more than three sides. When A_i^* and A_j^* both have exclusive nodes, they also create polygons. When $|A^*| = 4$, according to the second rule, there is at least one node in B connecting the four nodes in A. Then a polygon with n sides becomes a polygon with $n - 1$ sides, and the barcode has been changed.

The edges are added or deleted based on the three rules. In particular, when deleting edges, our scheme chooses the smallest set A_i^* and deletes the nodes in the set, ensuring $A_i^* \subseteq A_j^*$. After this step, our scheme chooses A_j^* and resumes the same deleting process. When adding edges, our scheme begins from adding nodes to the largest set A_k^*. Furthermore, A^* is restricted to three nodes if A belongs to $M1$.

When structure A contains both hole components and non-hole components, our scheme should not choose more than three nodes that originally belonged to the $M1$ matrices. Taking the $M3_2$ in Figure 5.4 as the example, we can first add three nodes from $M1_1$ to A^*. And if we want more edges, we can also add nodes from $M2_1$ to A^*, but nodes in A^* should be connected.

M2. Given the rules of the $M3$ matrix, regenerating $M2$ matrices becomes simple. Intuitively, an $M2$ matrix can be divided into two $M2$ matrices and two $M3$ matrices. And the two $M2$ matrices can be further divided until the size of each matrix is only one. Because the diagonal value of $M2$ matrices should all be 0, regenerating the $M2$ matrix can be viewed as regenerating a group of $M3$ matrices. In these $M3$ matrices, both A and B contain no holes. So only the first and second properties need to be considered in the regeneration.

After regenerating all the sub-matrices, we combine them together to form M_a', and use M_a' to build the new graph G'.



(a) $n = 12, \delta = 4$ (b) $n = 10, \delta = 3$

Fig. 5.10.: Comparing δ with $\frac{n}{3}$

5.3 Analysis

5.3.1 Privacy

Property 1. *The anonymization algorithm achieves ϵ edge differential privacy.*

Proof: In the MCMC procedure, the true distribution of $(f0, f1)$ is well approximated when the total step number is large enough. Then applying the exponential mechanism in sampling gives the $(f0, f1)$ under differential privacy.

The conquer-and-divide procedure achieves differential privacy when $f0$ and $f1$ of the super-matrix is calculated and satisfied. Because differential privacy is only concerned about the value $f0$ and $f1$, dividing the $(f0, f1)$ into different matrices does not violate the differential privacy criteria for the super-matrix. □

Furthermore, the persistent structure has the characteristic of preventing identity leakage. The polygons are the basic structure of persistent homology bars. The nodes on a polygon are isomorphic to each other. Also the barcode often has a group of the same bars. Hence, preserving the barcodes does not mean revealing the identity.

5.3.2 High-dimensional holes

Two properties related to the hypothesis of high-dimensional holes follow:

Property 2. *For a polygon with n nodes, holes do not exist when $\delta \geqslant \lfloor \frac{n}{2} \rfloor$.*

Table 5.2.: Network dataset statistics

Dataset	# of nodes	# of edges
ca-HepTh	574	2802
Facebook	2216	16308

Proof: In a polygon with n nodes, the maximum pairwise distance is $\lfloor \frac{n}{2} \rfloor$. When $\delta \geqslant \lfloor \frac{n}{2} \rfloor$, all pair of nodes are connected, and all holes die. $\square$

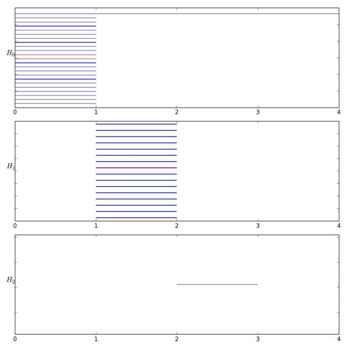
Property 3. *For a polygon with n nodes, there are no H_1 holes existing when $\delta \geqslant \lceil \frac{n}{3} \rceil$.*

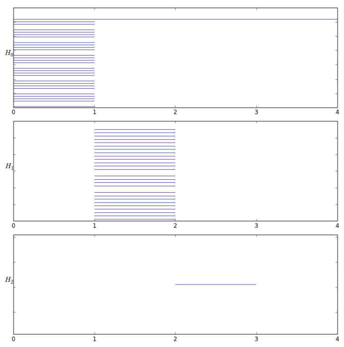
Proof: According to the definition of H_1 holes, it exists if and only if there is at least one area not filled with triangles in 2-D, which means there is at least one area that has more than three sides. As shown in Figure 5.10(a), when $\delta = \frac{n}{3}$, there are three areas in which all the distances are no greater than δ, and a triangle which is not a hole. When $\delta < \frac{n}{3}$, if the three areas still keep distances no greater than δ, then the remaining component is a hole like Figure 5.10(b). If δ is only defined to be integers, the upper limit of H_1 hole is $\lceil \frac{n}{3} \rceil$. $\square$

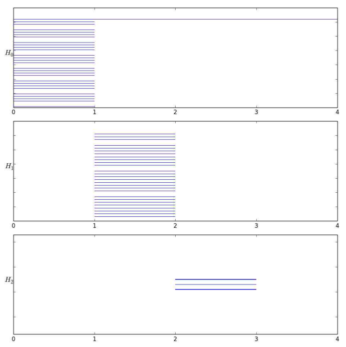
Intuitively, a high-dimensional hole exists only if the low-dimensional surface is complete. Considering a ball, when there is a hole on its surface, the void inside is broken. Then we have the hypothesis that high dimensional-holes only exist when $\delta \in [\lceil \frac{n}{3} \rceil, \lfloor \frac{n}{2} \rfloor)$.

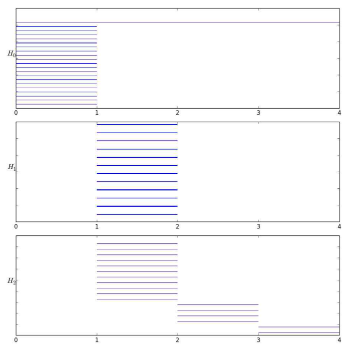
5.4 Evaluation

The proposed scheme aims to preserve the persistent structures in the OSN. However, the ultimate impact of the persistent homology on the utility of the graph needs further validation through evaluation. The evaluation is based on the Facebook dataset, and the ca-HepPh dataset [89]. The detailed information is shown in Table 5.2. The barcode extraction program is based on Perseus [102]. The dK-2 graph


(a) Original


(b) PHDP, $\epsilon = 10$


(c) PHDP, $\epsilon = 1$


(d) dK-2 model, $\epsilon = 10$

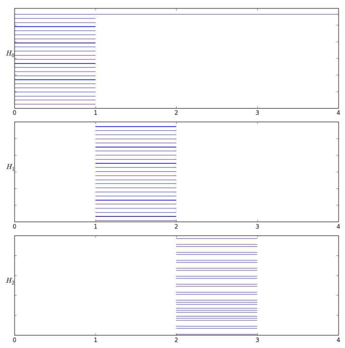
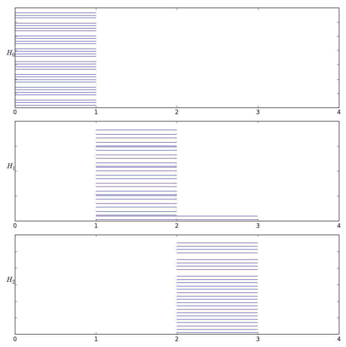

(e) E-R random graph

Fig. 5.11.: Barcodes of the ca-HepPh graph

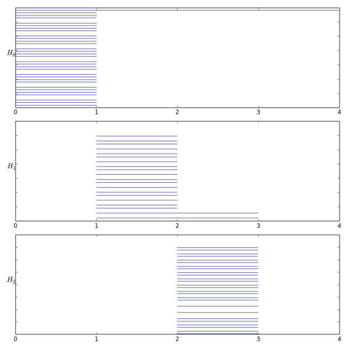
model [129], a differential privacy mechanism, as well as the Erdős-Rényi (E-R) model [45], are employed to compare with PHDP. In the following evaluation, the proposed scheme is marked as ‘PHDP’. The two differential privacy schemes, dK-2 and PHDP, are compared under the same differential privacy level $\epsilon = 10$. Furthermore, PHDP is also evaluated under a strict privacy level of $\epsilon = 1$.

5.4.1 Barcodes

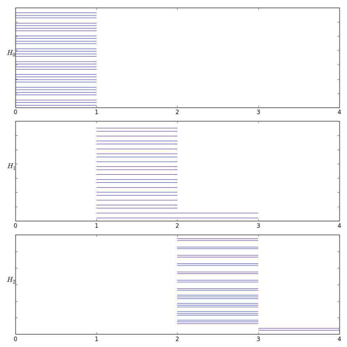
The first part of the evaluation is to validate the ability to preserve persistent homology of the schemes. Figure 5.11 and 5.12 report the persistent barcodes in the two datasets. Although all four anonymized graphs have more H_1 or H_2 bars, PHDP



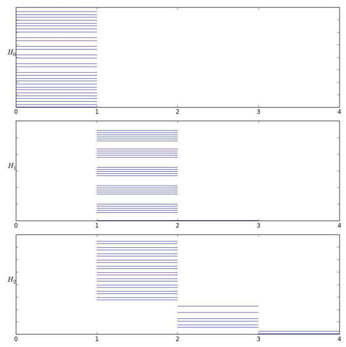
(a) Original



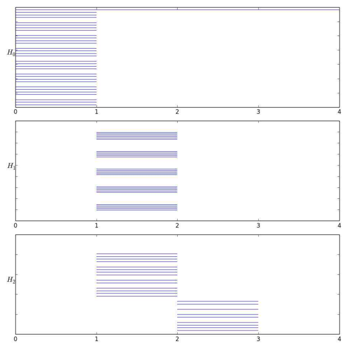
(b) PHDP, $\epsilon = 10$



(c) PHDP, $\epsilon = 1$



(d) dK-2 model, $\epsilon = 10$

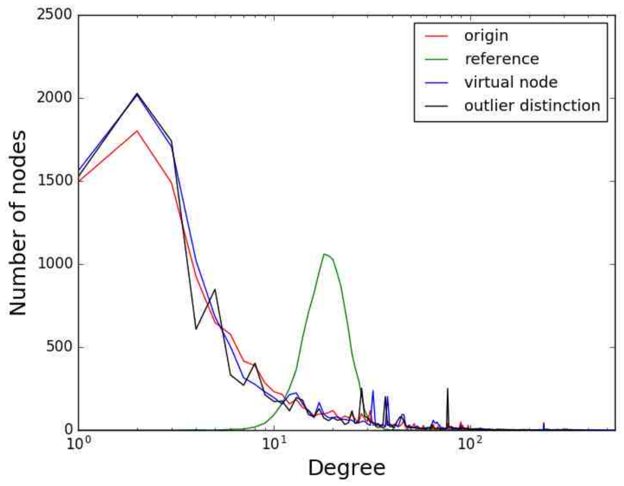


(e) E-R random graph

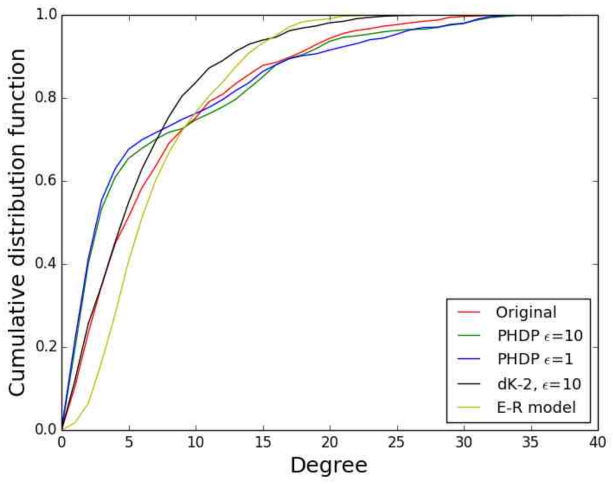
Fig. 5.12.: Barcodes of the Facebook network

has much less distortion in barcodes. In the original ca-HepPh graph, there are 16 H_1 bars and 1 H_2 bar. PHDP ($\epsilon = 10$) performs the best in preserving the bars: there are 22 H_1 bars and 1 H_2 bar. The PHDP ($\epsilon = 1$) result has 28 H_1 bars and 3 H_2 bars. The dK-2 result has 300 H_1 bars and 17 H_2 bars. The H_2 bars are $[3, 4)$ which implies that the anonymized graph has a 9-sided polygon. The E-R result has 591 H_1 bars and 49 H_2 bars.

The Facebook barcodes show a similar distribution. In the original graph, there are 185 H_1 bars and 28 H_2 bars. And the two numbers are 314 and 71 in the PHDP ($\epsilon = 10$) result, 327 and 58 in the PHSP ($\epsilon = 1$) result, 1142 and 76 in the dK-2 result, and 1688 and 21 in the E-R result.



(a) ca-HepPh



(b) Facebook

Fig. 5.13.: Degree distribution

The increase of the H_1 and H_2 bars suggests that there are more holes in the anonymized graph. The users 'on hole' are farther apart than the users 'on non-hole'. While PHDP is confirmed to preserve the persistent homology information under differential privacy criteria, the utility of dK-2 and E-R anonymized graphs is questionable because of the injected holes. The evaluation results demonstrate that an OSN with hundreds of nodes or thousands of nodes has very limited number of holes. Although the building of an OSN seems uncontrolled, the true OSNs are more strongly connected than artificial graphs in view of persistent homology.

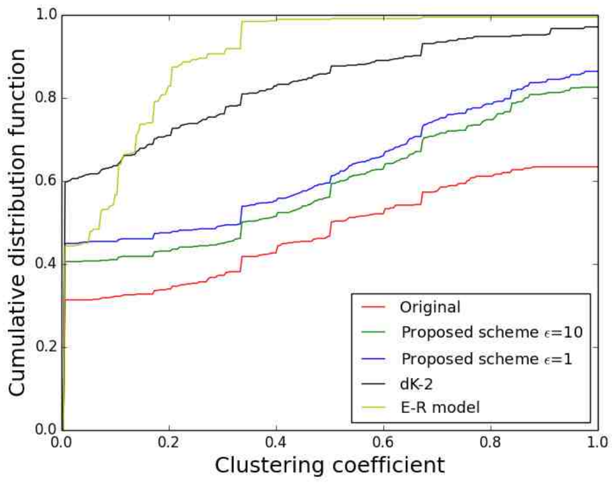
5.4.2 Utility metrics

To demonstrate the links between preserving persistent structures and graph utility, the performance of published graph under utility metrics are compared. The evaluation includes two graph utility metrics, the degree distribution and the clustering coefficient, and one application utility metric, the influence maximization.

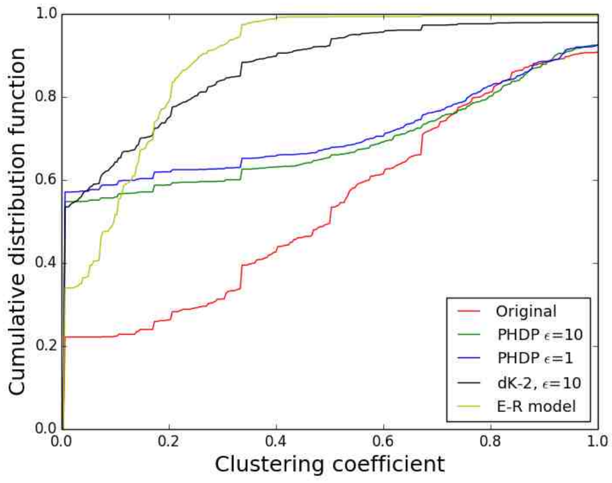
Degree distribution. Degree distribution is the number of connections of nodes among the graph. Figure 6.10 shows the degree distribution of the two datasets. The PHDP and dK-2 anonymized graphs match the degree distribution of the original graph. Compared to the original ca-HepPh graph, the degree distribution of the PHDP result ($\epsilon = 10$) has a root-mean-square error (RMSE) of 0.018, the PHDP result ($\epsilon = 1$) has a RMSE of 0.022, the dK-2 result ($\epsilon = 10$) has a RMSE of 0.018, but the E-R result has a RMSE of 0.110. Compared to the original Facebook graph, the PHDP ($\epsilon = 10$), PHDP ($\epsilon = 1$), dK-2 and E-R results have RMSE of 0.053, 0.062, 0.036 and 0.072.

The dK-2 anonymized graph maintains a similar degree distribution because it stores the paired degree information. The PHDP anonymized results suggest that the persistent homology information may have a soft impact on the degree. Intuitively, the holes restrict the edges. The E-R model only has the information of the average degree.

Clustering coefficient. The clustering coefficient shows the level of nodes clustering together. Figure 6.13 is the clustering coefficient of the two datasets. Only the PHDP anonymized graphs preserve some clustering information. The original ca-HepPh graph has an average clustering coefficient of 0.52, and it is 0.40 to PHDP ($\epsilon = 10$), 0.37 to PHDP ($\epsilon = 1$), 0.16 to dK-2 ($\epsilon = 10$) and 0.09 to E-R. The average



(a) ca-HepPh



(b) Facebook

Fig. 5.14.: Clustering coefficient distribution

clustering coefficients of the original Facebook graph, the PHDP ($\epsilon = 10$) result, the PHDP ($\epsilon = 1$), the dK-2 result and the E-R result are 0.45, 0.33, 0.30, 0.13 and 0.10, respectively.

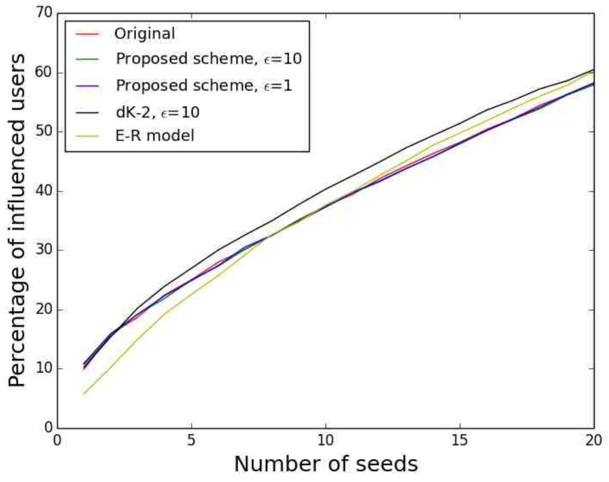
As shown in the evaluation of the barcodes, in a increasing order of number of holes the graphs are the original graph, the PHDP (ϵ=10), the PHDP (ϵ=10), the dK-2 and the E-R results. This order is also the decreasing order of clustering coefficient. It implies that holes occupy the position of clusters and then decrease the clustering coefficient. While holes are the opposite of clusters, the PHDP anonymized graphs preserve the clustering information by protecting the holes. When all holes in the graph are established, then remaining parts can be filled with clusters. In a real OSN, this shows that the number of holes are less than the number of clusters. Hence, storing holes opens a novel angle to maintain the graph structure.

Influence maximization. Influence maximization [73] is an application that first chooses the important users as seeds, then uses the seeds to influence other people. In the evaluation, a greedy algorithm based on the independent cascade model [25] is employed to choose the seeds who have the most ability to broadcast information. Then the percentage of influenced users are compared among different anonymized graphs, with the same propagation probability, 0.2.

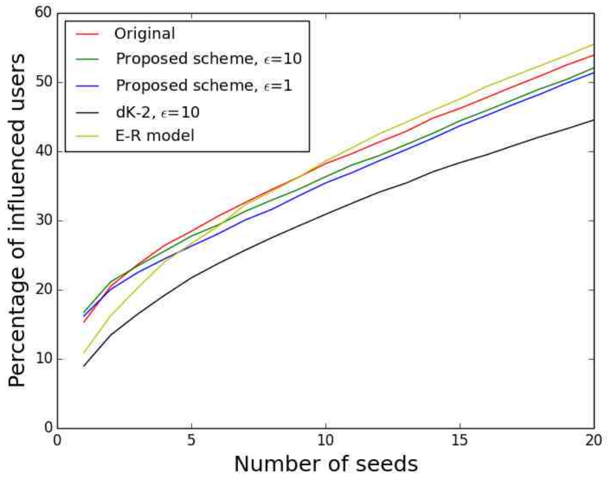
Figure 5.15 shows the percentage of influenced users. Although all four anonymized graphs achieve a similar data with the original graph, the PHDP results outperform the other. Compared to the original ca-HepPh data, the PHDP (ϵ=10) result has a RMSE of 0.40, the PHDP (ϵ=1) has a RMSE of 0.37, the dK-2 result has a RMSE of 2.50 and the E-R result has a RMSE of 2.30. Compared to the original Facebook data, the RMSEs are 1.58 of PHDP (ϵ=10), 2.43 of PHDP (ϵ=1), 7.57 of dK-2 (ϵ=10) and 2.13 of E-R.

This experiment suggests that the PHDP anonymized graphs are good at simulating the information broadcasting ability of OSNs. Since the influence maximization problem is closely related to the recommendation and advertisement application, the PHDP anonymized graph achieves high utility.

Conclusion. This evaluation proves that the PHDP scheme is capable of preserving both persistent homology and differential privacy. The PHDP anonymized result achieves high utility in both graph metrics and application metrics. Although



(a) ca-HepPh



(b) Facebook

Fig. 5.15.: Percentage of influenced users

the PHDP scheme is weaker than the dK-2 scheme in preserving the degree distribution, it achieves good results under all the utility metrics. It implies that the PHDP scheme does not target at a specific utility metric but comprehensively preserves the graph utility.

5.5 Conclusion

In this chapter, we address the utility concerns of the published graph by designing a novel anonymization scheme called PHDP under differential privacy. Unlike the existing anonymization schemes based on traditional components, e.g., node degree or clusters, PHDP employs a novel metric called persistent homology. When the persistent structures are in the form of holes, PHDP preserves the holes as well as satisfies the differential-privacy criteria. Evaluations on real OSNs confirm that protecting the holes help PHDP outperform the other schemes in both the graph utility metric and application metric. In the future, in addition to MCMC as the approximation method, we will try other methods to optimize the noise injection phase.

Although persistent homology gives a comprehensive measurement of graph utility, the proposed anonymization schemes, including PHDP, all passively affect the graph utility. While we actively set the differential privacy level ϵ, we cannot direct set the utility level. In the following chapter, we may discuss how to design an anonymization scheme which can actively adjust the privacy and utility balance.

6. ANONYMIZATION WITH NOVEL METHOD - SKETCHING

In Chapter 3, 4, and 5, the proposed anonymization schemes based on differential privacy can only directly adjust the privacy level but not the utility level. In this chapter, we aim to design a scheme which can actively adjust both utility level and privacy level, i.e., trim the privacy and utility balance.

In this chapter, we introduce the anonymization scheme based on All-Distance Sketch (ADS). Sketching can significantly limit attackers' confidence, as well as provide accurate estimation about shortest path length and other utility metrics. Because sketching removes large amounts of edges, it is invulnerable to seed-based and subgraph-based de-anonymization attacks. However, existing sketching algorithms do not add dummy edges and paths. Adversaries have low false positive in extracting linking information, which challenges the privacy performance. We propose the novel bottom-(l, k) sketch to defend against these advanced attacks. We develop a scheme to add and delete enough edges to satisfy our privacy demand.

The major technical contributions of this chapter are the following:

1. We embed the ADS algorithm, which preserves the distance information with bounded error, in OSN anonymization.
2. We propose the bottom-(l, k) sketch algorithm to improve the privacy of our anonymization scheme, which is suitable to defend against the advanced attacks.
3. We design the edge addition/deletion algorithm based on the ADS graph. Our algorithm intelligently add/delete edges without interrupting the properties of bottom-(l, k) sketch.

This chapter is previously published as a conference paper in IEEE International Conference on Sensing, Communication, and Networking (SECON), 2019 [53].

6.1 Preliminary

In this chapter, an OSN is modeled as an unweighted graph $G = (V, E)$, where V is the set of all vertices and E is the set of all edges. $|V|$ shows the cardinality of set V, i.e., the number of vertices. Moreover, we denote the ADS graph as $G_s = (V, E_s)$ and the final published graph as $G_p = (V, E_p)$.

MinHash sketch. MinHash sketch randomly summarizes a subset of k items from the original set [16]. Researchers designed three variations of MinHash sketch, named bottom-k, k-mins, and k-partition [30, 32, 34]. Specifically, bottom-k sketch samples k items with the lowest hash values; k-mins sketch samples one item each iteration with the lowest hash value and repeats the iteration k times (in each iteration, the hash values are different); k-partition sketch divides the original set into k subsets and samples one item from each subset. When the hash value of each item is randomly assigned, the sampled subset is a sketch of the original set. Among these sketching methods, bottom-k was proved to have the following benefits: (1) uniformly distributed ranking is an unbiased estimator; (2) with the same value of k, it gives higher performance in distance estimation than k-mins. Hence, our chapter concentrates on bottom-k sketch.

All-distance sketch. Based on MinHash sketch, researchers define the all-distance sketch to sample the data with graph structures [31]. The main idea of ADS is to keep nodes with the lowest hash values within a specific distance to the central node. In particular, we uniformly assign a rank $r(u)$ to each node u, where $r(u) \in [0, 1]$. ADS(v) contains a set of nodes and their distances to the central node v. If $u \in$ ADS(v), comparing all nodes w with distances $d_{vw} \leqslant d_{vu}$, node u has the bottom-k ranks.

Figure 6.1 gives an example of ADS when $k = 1$. In this graph, each node has a number to show the rank, e.g., $r(\text{A}) = 0.74$. Each edge has a number to show the distance, e.g., $d_{\text{AB}} = 1$. When sketching ADS(A), we first add node A itself in its ADS. Then, we examine nodes within distance one, i.e., node B. Node B has the

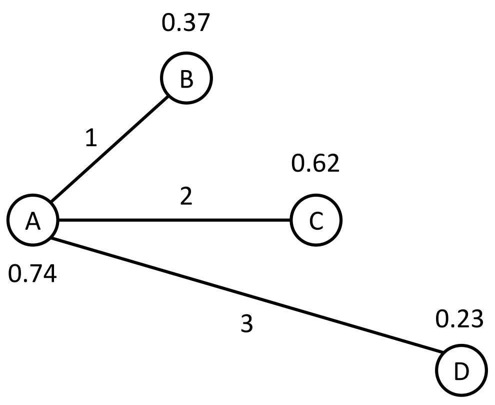


Fig. 6.1.: Example of bottom-1 ADS. $\mathrm{ADS(A)} = \{(\mathrm{A}, 0), (\mathrm{B}, 1), (\mathrm{D}, 3)\}$.

bottom-1 rank so add it to ADS(A). We examine nodes within distance two, i.e., nodes {B,C}. Because node C does not have the bottom-1 rank in the set, we do not add it to ADS(A). Finally, in the set of nodes within distance three, node D has the bottom-1 rank. Hence, $\mathrm{ADS(A)} = \{(\mathrm{A}, 0), (\mathrm{B}, 1), (\mathrm{D}, 3)\}$.

Hitherto two main approaches are introduced to compute ADS [31]. One combines pruning algorithms with Dijkstra's algorithm. The other is based on the dynamic programming process of the Bellman-Ford algorithm. Because the ADS computation algorithms can ignore nodes with high ranks, calculating ADS for one node is more efficient than other traversing algorithms, e.g., breadth-first search. Cohen and Kaplan prove that the bottom-k ADS is $O(k|E| \log |V|)$ [32].

6.2 Threat modeling

In this section, we model the ability of adversaries to capture true relationships, i.e., to ascertain the existence of edges in the original graph G, from examining the published graph G_p. Modeling helps us derive privacy and utility of our ADS anonymization scheme. Specifically, we model two kinds of adversaries who both have some information about the total number of added/deleted edges and the ADS anonymization scheme. The first kind of adversary focuses on target relationships.

The other kind of adversary wants to collect as much information as they can. These two kinds of adversaries are called 'spear attacker' and 'general attacker', respectively, in the following sections.

6.2.1 Spear attacker

In a spear attack, an attacker has some target users and he or she wants to know the properties of these users, e.g., the distances between two target users. The OSN service provider cannot predict the target users; however, the service provider can calculate the average of privacy and utility among all users to simulate an attack.

We use a parameter C to show the average confidence of the adversary to believe an edge exists in the original graph G when the adversary has the ADS graph G_s. In our assumption, this adversary knows the original graph size and the parameter of k in the ADS graph. In a graph with $|V|$ nodes, there are at most $|E_f|$ edges, where $|E_f| = \frac{|V|^2-|V|}{2}$. Then the average confidence C is given by,

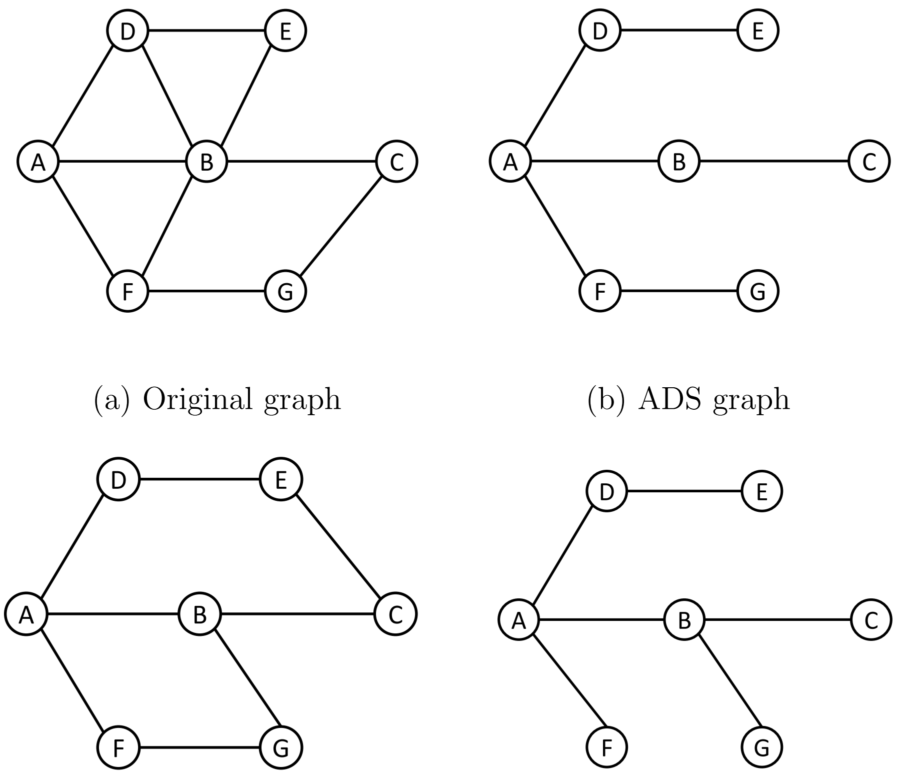
$$C = \frac{|E_s|}{|E_f|} + \frac{|E_f| - |E_s|}{|E_f|} \cdot \frac{|E_f| - |E|}{|E_f| - |E_s|} = \frac{|E_f| + |E_s| - |E|}{|E_f|}. \tag{6.1}$$

Here $\frac{|E_s|}{|E_f|}$ is the probability that the adversary chooses a pair of nodes that has an edge in G_s. Because all edges in G_s are inherited from edges in G, they are all true relationships. $\frac{|E_f|-|E_s|}{|E_f|}$ is the probability that the adversary chooses a pair of nodes that does not have an edge in G_s. For these nonexistent edges, the probability that the edge also does not exist in G is $\frac{|E_f|-|E|}{|E_f|-|E_s|}$. Hence, the adversary has the confidence $C = \frac{|E_f|+|E_s|-|E|}{|E_f|}$.

Utility of the ADS graph can be evaluated in several ways, according to specific utility metrics. For example, the theoretical bounds of distance is given by,

$$\overline{d_{uv}} \in \left[d_{uv}, \left(2\left\lceil\frac{log|V|}{logk}\right\rceil - 1\right) d_{uv}\right], \tag{6.2}$$

where d_{uv} is the true distance in G and $\overline{d_{uv}}$ is the distance in the sketched graph G_s [33].



(a) Original graph (b) ADS graph

(c) ADS graph with edge perturbation (d) Second-round ADS graph

Fig. 6.2.: Example of second-round ADS attack

6.2.2 General attacker

A general attacker does not have target users. Instead, he or she would like to infer as much information as possible from the published graph. For instance, the attacker wants to know if two users are connected in the original graph when these two users are connected in the published graph. Because important edges have a higher probability of being preserved in the ADS graph, the adversary can apply a second-round ADS attack to eliminate some unimportant edges.

Figure 6.2 shows an example of that attack. In Figure 6.2(c), we use an anonymization algorithm to add perturbed edges. However, having this perturbed graph, the adversary can make the ADS graph from Figure 6.2(c), which is shown in Figure 6.2(d).

Edges in the second-round ADS graph have a high probability of being the edges in the original graph. For example, path A − B − C is the only path that maintains the distance between A and C. The anonymization algorithm cannot use other paths to replace this path, because if it did, the distance measure between A and C would not be true.

By contrast, the misleading edges, e.g., edge E − C, are eliminated in the second-round ADS graph. If the anonymization algorithm cannot add dummy edges that have the same importance as the edges in the ADS, other dummy edges will not obfuscate adversaries as intended. After applying a second-round ADS to eliminate misleading edges, true edges in the original graph become distinct.

6.3 Scheme

When analyzing two kinds of adversaries, we find two main problems with directly applying ADS to anonymize OSNs. First, ADS only has edge removal, so spear attackers have no false positives when extracting linking information. Second, a second-round ADS attack enables the general attackers to extract linking information even though we randomly add dummy edges to the graph. Hence, we need to design an anonymization algorithm to effectively sketch the graph and add dummy edges.

In order to defend against the second-round ADS attack, we propose the bottom-(l, k) ADS graph, which has the following definition:

Definition 5 (Bottom-(l, k) ADS graph). *If node $u \in ADS(v)$ with the bottom-k sketch, there should be at least l paths between u and v in G_s, except that $d_{uv} = 1$. The graph G_s is the bottom-(l, k) ADS graph of G if and only if for all nodes v in G, G_s contains all nodes in v's ADS and all these l paths.*

Note that it is meaningless to build l paths between two connected nodes. Because OSNs are simple graphs without multiple edges, the sketched graph should also be simple graphs, otherwise the adversary can easily remove redundant edges.

The purpose of our scheme is to output a published graph G_p when the input is the original graph G. G_p is based on the bottom-(l, k) ADS graph G_s and G_p should maintain a balance between utility and privacy. Our scheme has the following parts:

1. We design an algorithm to generate the bottom-(l, k) ADS graph G_s.
2. We analyze the privacy and utility impact of edge addition and deletion, which guides graph perturbation.
3. We change the edges in G_s and publish G_p to achieve the privacy and utility balance.

6.3.1 Bottom-(l, k) ADS

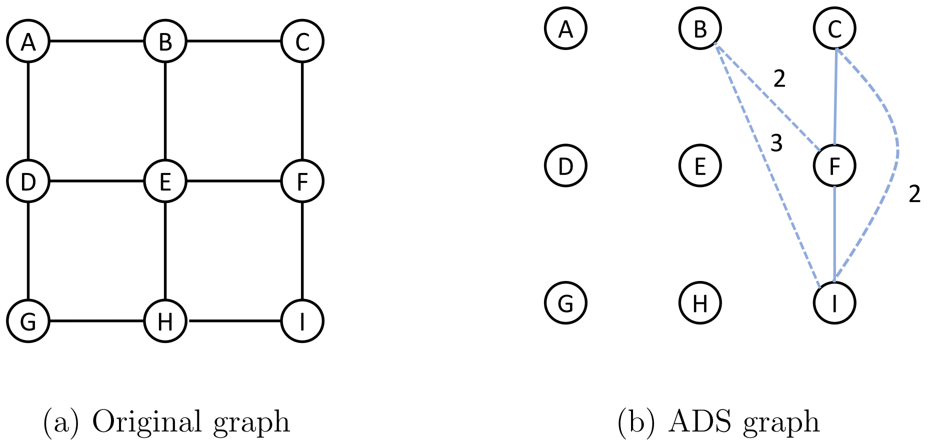
Existing bottom-k ADS algorithms have proven to be effective and efficient. However, these algorithms focus on generating the ADS sets and graphs, while our purpose is to preserve privacy in the sketched graph. For privacy purpose, reviewing the previous example in Figure 6.1, after knowing node D is in ADS(A), we should build at least l paths between nodes A and D.

The approach to draw the sketched graph has two parts: (A) extracting ADS for each node, and (B) building paths between nodes. Our new sketching algorithm, Algorithm 9, has the following steps:

1. We get the distance matrix M_d from the graph.
2. For each node in the matrix, we get its ADS set, and mark the nodes in this set.
3. We build a residual matrix M_r that initially sets the marked place to l.
4. For each number $\hat{l}$ in the residual matrix, we try to build $\hat{l}$ paths in three ways: directly linking, leaving to other paths, and adding new edges.

Algorithm 9 Bottom-(l, k) ADS

```
Input: Original graph $G$
Output: Sketched graph $G_s$, newly added edge $|E_a|$
1: Get distance matrix $M_d$ of the graph $G$
2: For each node $u$, if node $v \in ADS(u)$, mark position $(v, u)$
3: Create residual matrix $M_r$. If position $(v, u)$ is marked in $M_d$, set $M_r(v, u) = l$.
4: Create graph $G_s$ with no edge.
5: for each position $(v, u)$ in $M_r$ do
6:     If $M_d(v, u) = 1$, link $u$ and $v$, set $M_r(v, u) = 0$.
7:     Else if there is a node $w \in$ ADS$(v)$, and it has $M_d(v, u) = M_d(v, w) + M_d(w, u)$
   and $M_d(v, u) > 2$, set $M_r(v, u) = 0$.
8:     Else add $M_r(v, u)$ paths between $v$ and $u$
9: end for
```



(a) Original graph (b) ADS graph

Fig. 6.3.: Graph example of ADS

Here we use an example to show the sketching steps. Initially, we have the original graph G, which is shown in Figure 6.3(a). Our purpose is to generate the bottom-$(2, 1)$ sketched graph. This means that the ADS set is the bottom-1 sketch of the graph. And for each node $u \in$ ADS(v), there are at least 2 paths between u and v.

In the first step, we build the distance matrix M_d, $M_d(u, v) = d_{uv}$. In the second step, we extract the ADS set for each node. For example, if we want to get the ADS set of nodes I and F, we extract the two lines related to these two nodes (shown in Figure 6.4).

node		A	B	C	D	E	F	G	H	I
	rank	0.6	0.1	0.3	0.2	0.8	0.4	1	0.5	0.9
I	0.9	4	3	2	3	2	1	2	1	0
F	0.4	3	2	1	2	1	0	3	2	1

Fig. 6.4.: A part of M_d with nodes I and F

Having a line of M_d, we can get the ADS set for the corresponding nodes. For example, we have $\text{ADS(I)} = \{\text{B}, \text{C}, \text{F}, \text{I}\}$ and their corresponding distances. In Figure 6.4, we mark these nodes in blue. In Figure 6.3(b), we link the nodes in ADS(I) with node I. The dotted edge between C and I, for instance, means that $\text{C} \in \text{ADS(I)}$ and the distance between the two nodes is 2.

In the third step, we build the residual matrix M_r to show the number of residual paths between nodes. Initially, M_r is a zero matrix. If a position is marked in blue in M_d, we set this position to l. There is an exception for the same node cases, e.g., $M_r(\text{I}, \text{I}) = 0$.

node		A	B	C	D	E	F	G	H	I
	rank	0.6	0.1	0.3	0.2	0.8	0.4	1	0.5	0.9
I	0	0	2	2	0	0	2	0	0	0
F	0	0	2	2	0	0	0	0	0	0

Fig. 6.5.: Initial M_r with nodes I and F

In the forth step, we need to build paths according to M_r. Specifically, there are three cases: First, if we need to build l paths between u and v, and the two nodes are directly linked in G, we link the two nodes in G_s to complete this step. The edge F$-$I in Figure 6.3(b) is an example of this case.

Second, we try to find an intermediate node $w \in \mathrm{ADS}(v)$ between u and v. It is clear that if there are l paths between w and u (or between w and v), there should be at least l paths between u and v. However, we cannot leave all building tasks to w without checking the distances. Because between two directly connected nodes there is only the direct link, then if both $u - w$ and $v - w$ are directly connected, there is only one path between u and v instead of l different paths. Hence, we require that $M_d(u, v) > 2$, which means that the intermediate node is disconnected with at least one end node. Path B$-$I is an example of the second case. However, path C$-$I is a counter example since $M_d(\mathrm{C}, \mathrm{I}) = 2$.

Third, we need to build paths according to the updated M_r. In this example, we only need to build 2 paths between B$-$F and 2 paths between C$-$I. The process of building paths is the same as the process of choosing intermediate node sets. Consequently, building l paths is the same as choosing l node sets, and every two sets differ by at least one element. We have two cases in choosing intermediate nodes:

(1) The intermediate nodes can link the two end points with the desirable distance. For example, node E is the intermediate node between path B$-$F. The chosen intermediate nodes should satisfy the following criteria: First, the distances between

two end points should be larger than the distance between each intermediate nodes and each one of end points. Second, the distances between the intermediate nodes and one end point are in an incremental order, e.g., 1, 2, 3,... In this example, $M_d(\mathrm{B},\mathrm{E}) = 1$ and $M_d(\mathrm{B},\mathrm{F}) = 2$. Although $\mathrm{E} \notin \mathrm{ADS(B)}$ and $\mathrm{E} \notin \mathrm{ADS(F)}$, choosing E as the intermediate node between B and F still maintains the distance. Similarly, path C−F−I and path B−C−F are in this case.

(2) When we cannot find intermediate nodes in the first case, we choose a node set $\{w\}$ with the lowest distance increment. For example, we need to build another path between C and I in addition to C−F−I. When we choose node E (or B, H) as the intermediate node, we have the lowest distance increment 2. In general cases, when $\{w\} = \{w_1, w_2, ..., w_n\}$, and the two end nodes are marked as u and v, it requires that $M_d(u, w_1) + M_d(w_1, w_2) + ... + M_d(w_n, v) - M_d(u, v)$ is lowest compared to other sets of intermediate nodes.

The first case do not change the distances between nodes because they still sample edges in the original graph to build paths. However, the second case builds new edges and changes distances between some pairs of nodes. For example, after adding path C−E−I, d_{CE} changes. Fortunately, our algorithm has tolerance of edge addition, and its impact is analyzed in Section 6.3.2. Here we need to record the number of added edges, denoted by $|E_a|$. In the following sections, these edges occupies a part of our edge changing budget.

Finally, we get the sketched graph G_s. G_s of this example is shown in Figure 6.6.

6.3.2 Impact analysis of edge addition/deletion

As discussed in Section 6.2.1, a simple ADS graph is not suitable for preserving privacy in OSN data publication. In this section, we analyze the achieved privacy and preserved utility when we add/delete edges in the ADS graph.

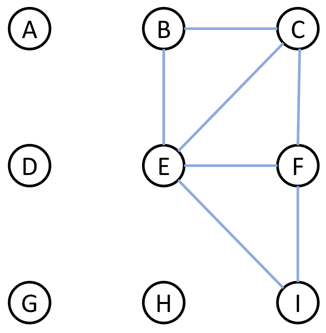


Fig. 6.6.: Part of the sketched graph G_s

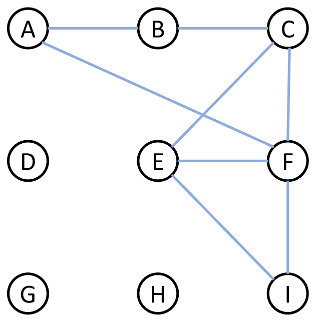


Fig. 6.7.: Part of the published graph G_p

After sketching the ADS graph G_s, we delete $p_d \cdot |E_s|$ edges and add $p_a \cdot |E_s|$ edges to generate the published graph G_p. The total number of changed edges is given by $p_c \cdot |E_s|$; p_c is the change rate and $p_c = p_a + p_d$.

The probability that the adversary chooses a pair of nodes that has an edge in G_p is given by $\frac{|E_p|}{|E_f|} = \frac{|E_s| \cdot (1-p_d) + (|E_f| - |E_s|) \cdot p_a}{|E_f|}$. For these existing edges, the probability that this chosen edge also exists in G is $\frac{|E_s| \cdot (1-p_d)}{|E_p|}$. The probability that the adversary chooses a pair of nodes that does not have an edge in G_p is given by $\frac{|E_f| - |E_p|}{|E_f|}$. For these nonexistent edges, the probability that this chosen pair of nodes also does not have an edge in G is $\frac{|E_f| - |E| - |E_s| \cdot p_a}{|E_f| - |E_p|}$. Hence, the new average confidence C of an adversary is given by,

$$\begin{aligned} C &= \frac{|E_s| \cdot (1 - p_d)}{|E_f|} + \frac{|E_f| - |E| - |E_s| \cdot p_a}{|E_f|} \\ &= \frac{|E_f| - |E| + |E_s| \cdot (1 - p_c)}{|E_f|}. \end{aligned} \tag{6.3}$$

Equation 6.3 gives an interesting property. One added edge has the same influence upon an adversaries' confidence as one deleted edge. In other words, when generating G_p from G_s, the confidence C is only related to the total number of changed edges, including both addition and deletion.

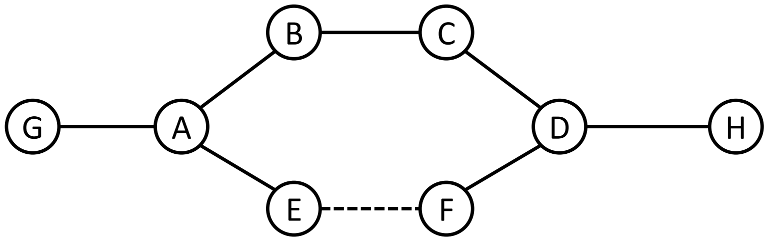


Fig. 6.8.: Example of edge addition and deletion; solid edges mean true relationships in G_s

How does the edge addition/deletion affect utility? We analyze the example in Figure 6.8. We assume node D is in ADS(A). In this example, we would like to switch the path from A − B − C − D to A − E − F − D. This means that we need to delete three edges and add the new edge E − F. In this example, $d_{\text{AD}} = 3$ is the total length of the path. In a general case, if we add d' edges ($d' \leqslant d$), the total number of changed edges is given by $d + d'$. Moreover, the total number of changed edges is also given by $p_c \cdot |E_s|$, we have $p_c \cdot |E_s| = d + d'$.

When calculating utility metrics, e.g., distances between nodes, we find that newly added edges cause some wrong results. These wrong results happen only if the following two conditions are satisfied: (1) We call the end nodes of newly added edges as 'trigger node'. Then the source node should be a 'trigger node', i.e., nodes E and F. When the source node is not a 'trigger node', e.g., nodes G and H, the utility metrics calculation is correct. (2) The distance between the source node and the target node should be farther than the distance between another 'trigger node' and the target node. For example, when calculating d_{DE}, the shortest path uses the newly added edge E − F and the distance is incorrect. In another example, when calculating d_{DF}, the newly added edge is not used. In a general case, each 'trigger node' chooses a target node and outputs a wrong distance result with probability about 50%. If 'trigger nodes' exists, each node averagely has 50% probability to give a wrong distance result. The number of 'trigger nodes' is $d' + 1$. Hence, the probability of getting a correct distance is given by,

$$\begin{aligned} P_{distance} &= 1 - \frac{d' + 1}{2|V|} \\ &= 1 - \frac{p_c \cdot |E_s| - d + 1}{2|V|} \\ &\leqslant 1 - \frac{p_c \cdot |E_s| + 2}{4|V|}. \end{aligned} \tag{6.4}$$

Reviewing the impact of edge addition/deletion upon privacy and utility, we made the following observation: When the release graph can give confidence $C = \frac{|E_f| - |E| + |E_s| \cdot (1 - p_c)}{|E_f|}$, the probability of getting the distance between nodes u and v as described in Equation 6.2 is given by $P_{distance} \leqslant 1 - \frac{p_c \cdot |E_s| + 2}{4|V|}$. Therefore, both privacy and utility is affected by p_c, i.e., the total number of added and deleted edges.

6.3.3 Edge changing

Based on the impact analysis, this subsection provides method to further perturb the sketched graph G_s. We control the privacy and utility tradeoff in the edge changing step by setting a desirable edge change rate p_c. Having a target p_c, we would like to change $p_c \cdot |E_s|$ edges in total, and publish the graph G_p. Since $|E_a|$ edges have already been added when sketching G_s, we need to change $|E_c|$ edges. $|E_c|$ is the edge changing budget and $|E_c| = p_c \cdot |E_s| - |E_a|$.

The main challenge of this part is that our perturbation should not violate the properties of bottom-(l, k) sketch. Specifically, if the perturbation deletes one path between u and v in this phase, we should add another path from u to v. Otherwise, our published graph cannot preserve as much utility as discussed in the impact analysis. Hence, our edge changing algorithm, Algorithm 10, has the following steps:

1. We build an edge-to-path matrix M_p to show the mapping relationships from edges to paths.

Algorithm 10 Edge changing

```
Input: Sketched graph $G_s$, change rate $p_c$, number of added edges $|E_a|$
Output: Final published graph $G_p$
1: Get number of edges need to change $|E_c| = p_c \cdot |E_s| - |E_a|$
2: Get path-to-edge matrix $M_p$
3: Initialize $G_p$ as $G_s$
4: while $|E_c| > 0$ do
5:     Randomly pick an edge in $G_p$, delete it, $|E_c| -= 1$
6:     Search this edge's mapping paths.
7:     For each path, find the alternative path and build it.
8:     For each adding edge, $|E_c| -= 1$.
9:     Update $M_p$
10: end while
```

2. We randomly choose an edge in M_p, delete that edge and add other edges to maintain the paths.
3. We continue doing the second step until our budget $|E_c|$ is exhausted.

Similarly, we explain the details of edge changing steps in this example. The input graph is G_s, shown in Figure 6.6. Our purpose is to generate the published graph G_p.

path	edge	B-C	B-E	B-F	C-E	C-F	E-F	E-I	F-I	E-H	F-H	H-I
F-I		0	0	0	0	0	0	0	1	0	0	0
C-F		0	0	0	0	1	0	0	0	0	0	0
C-I		0	0	0	1	1	0	1	1	0	0	0
B-F		1	1	0	0	1	1	0	0	0	0	0
B-I		1	1	0	0	1	1	0	1	0	0	0

Fig. 6.9.: Edge-to-path matrix M_p of the graph in Figure 6.6

In the first step, we build the edge-to-path matrix M_p, which is shown in Figure 6.9. The first line shows the edges in G_s. An edge is marked in grey while it exists in the original graph G. In the following steps, we choose only these marked edges to delete because deleting other edges does not influence the confidence of the adversary. The first column shows the paths (named in two end points). The remaining part contains 0s and 1s to show whether an edge contributes to paths between the two end nodes.

In the second step, we randomly choose an edge and delete it. After we delete an edge, we search M_p for the corresponding paths and build these paths. In the path-building process, we search for intermediate nodes, which have the same two cases as previously discussed in Section 6.3.1.

In this example, we plan to delete the edge B−E. From matrix M_p, we find that we need to build one new path for B−F and one for B−I, when deleting B−E. Because path B−F is part of path B−I, building a new path with B−F will also bring a new path with B−I. Hence, we only focus on building B−F here. $d_{\mathrm{BF}} = 2$ means we need to find one intermediate node. Because we cannot find the intermediate node in the first case, i.e., with the desirable distance, we need to search nodes in the second case, i.e., with the lowest distance increment. In this example, it is node A. Then, we link edges B−A and A−F to form the path B−F.

In the third step, we count the number of edges that have already been added or deleted and compare this number with our budget $|E_c|$. In this example, we delete one edge B−E and add two edges B−A and A−F, so the budget is consumed by three. If $|E_c| \leqslant 3$, the budget is exhausted and we meet our privacy requirement. Finally, we get the published graph G_p. G_p of our example is shown in Figure 6.7.

Table 6.1.: Number of edges and non-edges in G and G_p

	Edges in G	Non-edges in G	Sum
Edges in G_p	$\lvert E_s\rvert \cdot (1-p_d)$	$\lvert E_s\rvert \cdot p_a$	$\lvert E_s\rvert \cdot (1-p_d+p_a)$
Non-edges in G_p	$\lvert E\rvert - \lvert E_s\rvert + \lvert E_s\rvert \cdot p_d$	$\lvert E_f\rvert - \lvert E\rvert - \lvert E_s\rvert \cdot p_a$	$\lvert E_f\rvert - \lvert E_s\rvert \cdot (1-p_d+p_a)$
Sum	$\lvert E\rvert$	$\lvert E_f\rvert - \lvert E\rvert$	$\lvert E_f\rvert$

6.4 Analysis

6.4.1 Privacy analysis

Privacy is always measured from the third parties' perspective, i.e., how much information is released to the third parties. This chapter focuses on edge privacy. Hence, C denotes the confidence of the attacker about a target edge exists (or does not exist) in the original graph G, when he or she holds the published graph G_p.

$$\begin{aligned} C &= \frac{|E_f| - |E| + |E_s| \cdot (1-p_c)}{|E_f|} \\ &= \frac{|V|^2 - |V| - 2|E| + 2|E_s| \cdot (1-p_c)}{|V|^2 - |V|}. \end{aligned} \tag{6.5}$$

Differential privacy gives another measurement of privacy. It formalizes the relationship between G and G_p. Specifically, for an anonymization mechanism $\mathcal{A}$ to achieve ϵ-differential privacy, two neighboring graphs G_1 and G_2 should have the following relationship,

$$\Pr[\mathcal{A}(G_1) \in S] \le e^{\epsilon} \times \Pr[\mathcal{A}(G_2) \in S], \tag{6.6}$$

where S is the output space of $\mathcal{A}$. To satisfy edge differential privacy, the neighboring graphs G_1 and G_2 have one edge difference. Equation 6.6 suggests that the probability of G_p having one specific edge is not greater than e^{ϵ} times of the probability of this edge does not exist, and vice versa. Hence, information leakage in G_p is bounded with the parameter ϵ.

Here, we propose a similar privacy bound in an average manner.

Property 4. *To an attacker who holds the published graph G_p, the average probability that all edges exist (resp. do not exist) in the original graph is not higher than e^{ϵ_a} times the probability that these edges do not exist (resp. exist) in the original graph, where* $\epsilon_a = \ln\left(\frac{|E_s|\cdot(|E_f|-|E|)\cdot(1-p_d)\cdot(1-p_d+p_a)}{|E_f|\cdot|E|\cdot p_a} + \frac{|E_f|-|E_s|\cdot(1-p_d+p_a)}{|E_f|}\cdot\frac{(|E|-|E_s|+|E_s|\cdot p_d)(|E_f|-|E|)}{(|E_f|-|E|-|E_s|\cdot p_a)\cdot|E|}\right)$.

Proof: Without loss of generality, we assume the attacker randomly chooses a pair of nodes (u, v). In the two neighboring graphs G_1 and G_2, this node pair is an edge in G_1 and a non-edge in G_2. Hence, we have $(u, v) \in E_1$ and $(u, v) \notin E_2$. After anonymization, the two published graphs G_{p1} and G_{p2} should have the same output. There are two different cases, $(u, v) \in E_p$ and $(u, v) \notin E_p$.

In the first case, the two published graphs both contain edge (u, v), $S = \{(u, v) \in E_p\}$. We have,

$$\begin{aligned}\Pr[\mathcal{A}(G_1) \in S] &= \Pr[(u,v) \in E_{p1}|(u,v) \in E_1]\\ \Pr[\mathcal{A}(G_2) \in S] &= \Pr[(u,v) \in E_{p2}|(u,v) \notin E_2]\end{aligned} \tag{6.7}$$

Table 6.1 shows the number of edges in G and G_p. For example, the number of node pairs that are connected both in G and G_p is $|E_s| \cdot (1-p_d)$. Table 6.1 gives that,

$$\begin{aligned}\Pr[(u,v) \in E_{p1}|(u,v) \in E_1] &= \frac{\Pr[(u,v) \in E_{p1} \cap (u,v) \in E_1]}{\Pr[(u,v) \in E_1]}\\ &= \frac{|E_s| \cdot (1-p_d)}{|E|}\\ \Pr[(u,v) \in E_{p2}|(u,v) \notin E_2] &= \frac{\Pr[(u,v) \in E_{p2} \cap (u,v) \notin E_2]}{\Pr[(u,v) \notin E_2]}\\ &= \frac{|E_s| \cdot p_a}{|E_f| - |E|}\end{aligned} \tag{6.8}$$

Then,

$$e^{\epsilon_1} = \frac{\Pr[\mathcal{A}(G_1) \in S]}{\Pr[\mathcal{A}(G_2)]} = \frac{(|E_f| - |E|) \cdot (1-p_d)}{|E| \cdot p_a} \tag{6.9}$$

In the second case, the two published graphs both contain non-edge (u, v), $S = \{(u, v) \notin E_p\}$. Similarly, we get,

$$\begin{aligned}
\Pr[\mathcal{A}(G_1) \in S] &= \Pr[(u, v) \notin E_{p1} | (u, v) \in E_1] \\
&= \frac{|E| - |E_s| + |E_s| \cdot p_d}{|E|} \\
\Pr[\mathcal{A}(G_2) \in S] &= \Pr[(u, v) \notin E_{p2} | (u, v) \notin E_2] \qquad (6.10) \\
&= \frac{|E_f| - |E| - |E_s| \cdot p_a}{|E_f| - |E|} \\
e^{\epsilon_2} &= \frac{(|E| - |E_s| + |E_s| \cdot p_d)(|E_f| - |E|)}{(|E_f| - |E| - |E_s| \cdot p_a) \cdot |E|}
\end{aligned}$$

The probabilities of the first and second case are $\frac{|E_s| \cdot (1 - p_d + p_a)}{|E_f|}$ and $\frac{|E_f| - |E_s| \cdot (1 - p_d + p_a)}{|E_f|}$, respectively. Finally, we can get the average privacy parameter ϵ_a.

$$\begin{aligned}
e^{\epsilon_a} &= \frac{|E_s| \cdot (1 - p_d + p_a)}{|E_f|} e^{\epsilon_1} \\
&+ \frac{|E_f| - |E_s| \cdot (1 - p_d + p_a)}{|E_f|} e^{\epsilon_2} \\
\epsilon_a &= \ln \left(\frac{|E_s| \cdot (|E_f| - |E|) \cdot (1 - p_d) \cdot (1 - p_d + p_a)}{|E_f| \cdot |E| \cdot p_a} \right. \qquad (6.11) \\
&+ \frac{|E_f| - |E_s| \cdot (1 - p_d + p_a)}{|E_f|} \\
&\left. * \frac{(|E| - |E_s| + |E_s| \cdot p_d)(|E_f| - |E|)}{(|E_f| - |E| - |E_s| \cdot p_a) \cdot |E|} \right)
\end{aligned}$$

□

The main idea behind Property 4 is to apply the parameter e^{ϵ_a} to measure the information leakage. Our sketch scheme does not follows steps in the traditional differential privacy mechanisms, e.g., using sensitivity to calibrate the amount of noise and designing the perturbed distribution. However, we claim that sketching has a similar impact to data with traditional differential privacy perturbation with the proof.

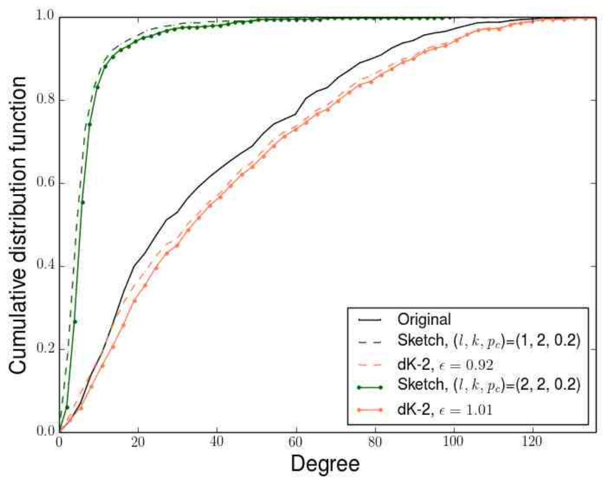


Fig. 6.10.: Degree distribution, Facebook.

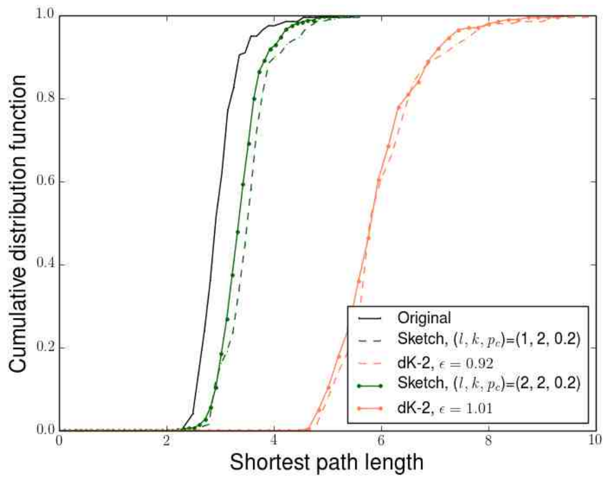


Fig. 6.11.: Shortest path length distribution, Facebook.

6.5 Evaluation

In this section, we evaluate our sketching scheme over three real-world datasets, namely ca-HepTh, Facebook, and Enron [88]. We demonstrate the utility of our published graphs with different (l, k) and desired change rate p_c. For comparison

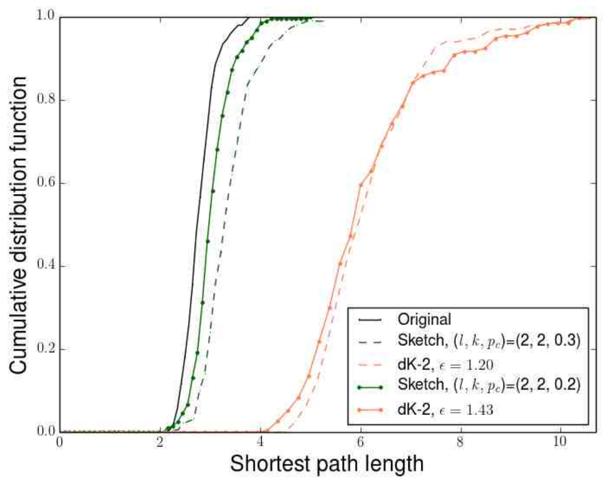


Fig. 6.12.: Shortest path length distribution, ca-HepPh.

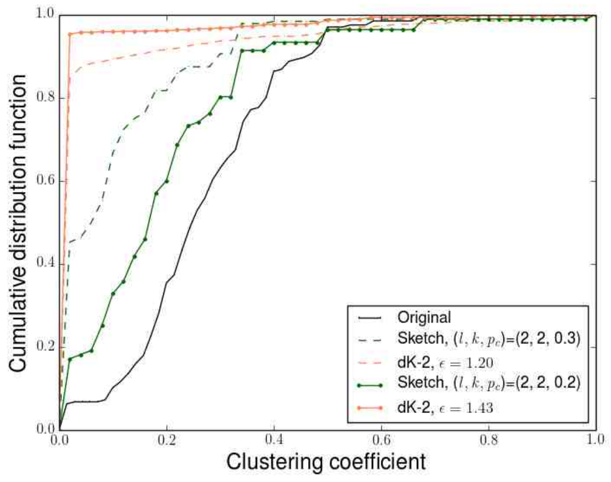


Fig. 6.13.: Clustering coefficient distribution, ca-HepPh.

purposes, we implement one anonymization mechanism as reference, which is the differential privacy algorithm with the dK-2 model [129, 148]. We use Equation 6.11 to calculate the corresponding ϵ_a of our scheme; then we set $\epsilon = \epsilon_a$ as the

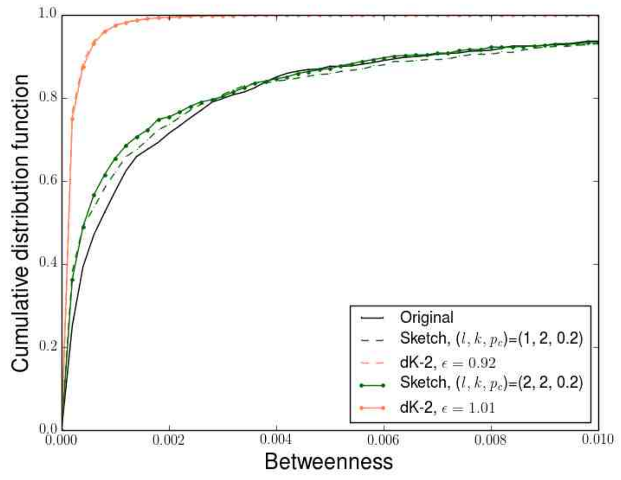


Fig. 6.14.: Betweenness distribution, Facebook

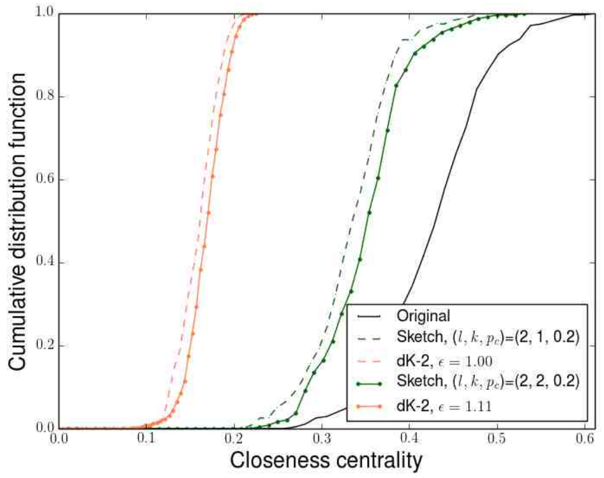


Fig. 6.15.: Closeness centrality distribution, Enron

privacy requirement in the reference dK-2 anonymization method. When the two anonymization schemes have similar privacy requirement, we evaluate and compare the utility preservation of the published graphs.

Degree. The degree of a node in the network is the number of edges incident to the node. While the dK-2 model preserves the degree of nodes, our sketching algorithm removes large amounts of unnecessary edges, which has a great impact on degree. Figure 6.10 shows the degree distribution of the Facebook dataset. When $(l, k, p_c) = (1, 2, 0.2)$, we have $\epsilon_a = 0.92$. When $(l, k, p_c) = (2, 2, 0.2)$, we have $\epsilon_a = 1.01$. Then the degree distributions of corresponding ϵs are also analyzed.

The average degrees of the original Facebook graph, our sketching result ($l = 1$), the dK-2 anonymized result ($\epsilon = 0.92$), our sketching result ($l = 2$), and the dK-2 anonymized result ($\epsilon = 1.01$) are 36.14, 6.51, 40.08, 7.36, and 41.74, respectively. In all of our experiments, there are about 20% edges remaining in our sketched results. The deletion of edges greatly impacts the degree distribution of our published graph. Moreover, when there are less paths remaining in the graph (lower l), more edges are deleted.

Average shortest path length. Average shortest path length implies the information transmission speed in the network. The dK-2 node can only indirectly preserve the shortest path length information when combining two edges with one same-degree node. ADS is appropriate to preserve distance information, as is our anonymized sketching algorithm. Figure 6.11 and 6.12 show the shortest path length distribution in the Facebook and the ca-HepPh datasets. In the ca-HepPh dataset, when $(l, k, p_c) = (2, 2, 0.3)$, we have $\epsilon_a = 1.20$. When $(l, k, p_c) = (2, 2, 0.2)$, we have $\epsilon_a = 1.43$. Then the shortest path length distributions of corresponding ϵs are also analyzed.

The average shortest path length of the original ca-HepPh graph, our sketching result ($p_c = 0.3$), the dK-2 anonymized result ($\epsilon = 1.20$), our sketching result ($p_c = 0.2$), and the dK-2 anonymized result ($\epsilon = 1.43$) are 2.78, 3.36, 6.14, 3.04, 6.11, respectively. While the dK-2 anonymized graphs double the shortest path lengths, our published graphs preserve these lengths with only about 20% edges. Moreover, when there are more paths (higher l) or lower privacy demand (lower p_c), our sketched graph can more precisely preserve the shortest path length information.

Clustering coefficient. Clustering coefficient is a measure of how nodes in a graph tend to cluster together. Figure 6.13 shows the clustering coefficient distribution of the ca-HepPh dataset. The average clustering coefficient of the original ca-HepPh graph, our sketching result ($p_c = 0.3$), the dK-2 anonymized result ($\epsilon = 1.20$), our sketching result ($p_c = 0.2$), and the dK-2 anonymized result ($\epsilon = 1.43$) are 0.27, 0.10, 0.05, 0.18, 0.02, respectively. Neither anonymization algorithm can preserve this information theoretically or experimentally.

Betweenness. Betweenness centrality shows the number of shortest paths that pass through a node. Figure 6.14 shows the betweenness distribution of the Facebook dataset. The average betweenness of the original Facebook graph, our sketching result ($l = 1$), the dK-2 anonymized result ($\epsilon = 0.92$), our sketching result ($l = 2$), and the dK-2 anonymized result ($\epsilon = 1.01$) are 0.0026, 0.0035, 0.0002, 0.0034, and 0.0002, respectively.

Because our anonymized sketching algorithm preserves a part of shortest paths, its distribution is very similar to the original distribution. Moreover, in our bottom-(l, k) sketch, when we slightly increase l, the published graph has better performance in preserving betweenness than the basic bottom-k sketch. Because there always have more than one path between two nodes, l is a compensation to edge removal in sketching.

Closeness centrality. Closeness centrality measures the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Figure 6.15 shows the closeness centrality distribution of Enron network. In the Enron dataset, when $(l, k, p_c) = (2, 1, 0.2)$, we have $\epsilon_a = 1.00$. When $(l, k, p_c) = (2, 2, 0.2)$, we have $\epsilon_a = 1.11$. Then the closeness centrality distributions of corresponding ϵs are also analyzed.

The average closeness centrality of the original Enron graph, our sketching result ($k = 1$), the dK-2 anonymized result ($\epsilon = 1.00$), our sketching result ($k = 2$), and the dK-2 anonymized result ($\epsilon = 1.11$) are 0.34, 0.30, 0.17, 0.30, and 0.18, respectively. Similarly, we find that our anonymized sketching algorithm preserves closeness centrality distribution better than the dK-2 anonymization algorithm. To conclude, the proposed scheme outperforms the existing dK-2 scheme in preserving utility under the similar privacy requirement, which shows a better balance between privacy and utility in the published graph.

6.6 Conclusion

In this chapter, we apply sketching to anonymize OSN data. We propose the bottom-(l, k) sketch algorithm to prevent second-round ADS attack. We introduce the edge changing algorithm to increase attackers' uncertainty and strike the balance between utility and privacy. The experiments show that our anonymized sketching algorithm can better preserve graph utility when the privacy preservation level is close to differential privacy.

In this chapter and previous chapters, we study the anonymization scheme, e.g., the privacy and utility preservation, from the defenders' point of view. However, the attacker also plays an important role in the privacy preservation problem. Analyzing the attackers' abilities and behaviors may give us hints about how to prevent them. In the following chapters, we may discuss how can the attackers de-anonymize the anonymized data.

7. DE-ANONYMIZATION WITH MAPPING SEEDS - PERSISTENT STRUCTURES

In previous four chapters, we study the OSN data anonymization problem. In this chapter, we focus on the OSN data de-anonymization problem. We start with a traditional de-anonymization techniques called seed-and-grow mapping. The OSN service providers may sequentially release data and the adversaries can get the dynamic OSN data. This chapter introduces a scheme which extracts dynamic information and improves existing seed finding algorithm.

In this chapter, we deploy persistent homology to capture the evolution of OSNs. Persistent homology barcodes show the birth time and death time of holes, i.e., polygons, in dynamic OSNs. By extracting the evolution of holes, persistent homology captures the addition/deletion of edges, which is the crucial feature of dynamic OSNs. After extracting the evolution of holes, we apply a two-phase de-anonymization attack. First, holes are mapped together according to the similarity of birth/death time. Second, already mapped holes are converted into super nodes, which are seed nodes; we then grow the mapping based on these seed nodes. Our de-anonymization scheme is extremely compatible to the adversaries who suffer latency in relationship collection, which is very similar to real-world cases.

The major technical contributions of this chapter are the following:

1. We probabilistically model the ability of adversaries to get the true relationships, which is realistic in real-world cases.
2. We apply persistent homology to capture the persistent structures in dynamic graphs.
3. We introduce a seed-and-grow algorithm to map nodes in dynamic graphs.

This chapter is previously published as a conference paper in IEEE International Conference on Communications (ICC), 2019 [51].

7.1 Threat model

In this chapter, the OSN service provider continues releasing the time-series graph G^A. $G^A = (V^A, E^A)$, where V^A is the set of vertices, if that vertex exists in at least one time period in G^A, and E^A is the set of edges. G^A is modeled as a set of static graphs. $G^A = \{G_1^A, G_2^A, ..., G_n^A\}$, where G_i^A is the graph in i-th epoch. Meanwhile, the attacker builds another time-series graph G^B with his/her background knowledge. The purpose of this chapter is to map nodes from G^B to nodes in G^A.

In this chapter, we assume all nodes in G^B form a node set V^B, which is the subset of V^A (all nodes in G^A). The edge set E_s^A is a subset of the edge set E^A, E_s^A contains all edges whose both nodes have corresponding nodes in V^B. When discussing the edges in G^A, we only focus on the edges in E_s^A.

However, the mapping task has two major challenges. First, the service provider does not directly release the graph. The service provider not only removes the identities in each static graph, but also perturb the graph before releasing. In the original graph, because new links may form and old links may drop, the self mapping of G^A is as hard as the de-anonymization work between two static graphs.

Second, the attacker cannot have perfect background knowledge graph, which is reflected in G^B. In this chapter, we model the shortfall of the attacker's background knowledge by two factors: latency and unknown. Latency means that for some relationships forming/dropping in epoch i in the original OSN, the attacker knows this relationship is forming/dropping in epoch j, where $j \geqslant i$. Unknown means the attacker always have wrong information about some specific relationships. Particularly, if an edge is added/deleted in G_i^A, G_j^B adds/deletes that corresponding edge with probability,

$$p = 1 - \beta \cdot \exp^{-\alpha \cdot (j-i)}, \quad j \geqslant i \tag{7.1}$$

Two scaling parameters, α and β, represent the similarity between G^A and G^B. $\alpha \in [0, \infty)$ shows the latency of the attacker to collect the information. If $\alpha = \infty$, the attacker instantly collects the edge forming/dropping information. β shows the error of the attacker's collected information. If $\beta = 0$, there are no unknown relationships to the attacker. Having an edge forming in epoch i_1 and dropping in i_2, the existing probability of this edge in epoch j is,

$$p = \begin{cases} \beta, & j \in [0, i_1) \\ 1 - \beta \cdot \exp^{-\alpha \cdot (j - i_1)}, & j \in [i_1, i_2) \\ \beta \cdot \left(\exp^{-\alpha \cdot (j-i_2)} - \exp^{-\alpha \cdot (j - i_1)}\right), & j \in [i_2, \infty) \end{cases} \tag{7.2}$$

7.2 Scheme

Given two dynamic graphs G^A and G^B, our goal is to map the nodes between G^A and G^B. The general idea of the scheme is to use the persistent structures as seeds in mapping. Our scheme has the following steps:

1. Self mapping: Having $G^A = \{G_1^A, G_2^A, ..., G_n^A\}$, if the node identifiers are not retained in G^A, we map the nodes between the series G^A.
2. Persistent structure extracting and mapping: We extract the barcode and the corresponding structures of G^A and G^B. Then for each persistent structure in G^A, we try to find a similar structure in G^B.
3. Match growing: After getting the pairwise persistent structures between G^A and G^B, we use the persistent structures as seeds to grow node mapping.

7.2.1 Self mapping

In some cases, the time-series graph is anonymized and then published by the OSN provider. The adversaries cannot have coherent identities of users in each epoch. Hence, we need to first mapping nodes of graphs in different epochs in G^A.

The process of self mapping a dynamic graph is similar to the process de-anonymization process with static graphs. Particularly, we sequentially de-anonymize the graphs G_1^A with G_2^A, G_2^A with G_3^A, and so on. For each mapping, we perform a two-stage de-anonymization attack in Algorithm 12 [113]. The general idea of Algorithm 12 is that

Algorithm 11 Self mapping

```
Input: Two neighboring slices of $G^A$, $G_i^A$ and $G_{i+1}^A$
Output: A node mapping result between users in $G_i^A$ and $G_{i+1}^A$
 1: Set $G^L = G_i^A$, $G^R = G_{i+1}^A$,
 2: In $G^L$ and $G^R$, both select $k$ users with highest degree values as seeds,
 3: for each pair of seeds $u$ and $v$ do
 4:     Compute similarity score $S^1 = \alpha S_p^1 + \beta S_a^1$
 5: end for
 6: Exhaustively search mapping results to get $\max \sum_{u,v} S^1$,
 7: for each pair of seeds $u$ and $v$ do
 8:     Compute similarity score $S^2 = \alpha S_p^2 + \beta S_a^1$
 9: end for
10: loop
11:     Pick a node $u$ with the BFS algorithm,
12:     Find the best match nodes $v$ with $\max S^2$
13: end loop
```

the pop stars are more difficult to hide during anonymization than common users. Moreover, there is low probability that these popular users are newly added/deleted in OSNs.

In the two graphs G^L and G^R, e.g., G_1^A and G_2^A in the dynamic graph, we first choose k nodes with the highest degree. To each pair of nodes, which chosen from G^L and one chosen from G^R, we calculate a similarity score S. The similarity score considers both the topology similarity S_p and the attribute similarity S_a.

S_p is based on the cost $Cost$ of optimally matching two neighbor degree lists (NDLs) [137]. Specifically, for the chosen node in G^L, we first get its neighbor list and the corresponding degrees. Then we construct the NDL of that node and compare it with the NDL of the node in G^R. Finally, we calculate S_p.

$$S_p(u, v) = -Cost(\text{NDL}(u), \text{NDL}(v)), \quad u \in G^L, \ v \in G^R. \tag{7.3}$$

Table 7.1.: Example of calculating the optimal matching cost $Cost$

NDL(u)	5	3	0
NDL(v)	4	2	1
Costs	-1	-1	+1

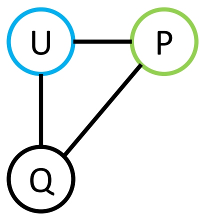
See Table 1 for an illustration of calculating costs to determine S_p. If, for example, the node u in G^L has two neighbors with degrees 3 and 5, and the node v in G^R has three neighbors with degrees 1, 4, and 2, then we can use the method shown in Table 7.1 to calculate the cost. First, we transform the two NDLs into decreasing order and add 0 to make them having the same length. Second, we calculate the differences between the two bits in the same position. Third, we get the total cost, which is the sum of the absolute values of all costs. In this example, $Cost$ equals 3 and $S_p = -3$.

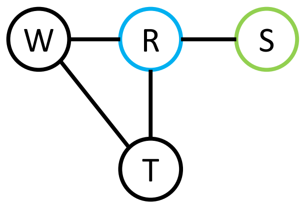
S_a is the similarity of two users' attributes. We use the Jaccard index to measure the two sets of attributes $A(u)$ and $A(v)$.

$$S_a^1(u, v) = \frac{A(u) \cap A(v)}{A(u) \cup A(v)} \tag{7.4}$$

Then we have $S^1 = S_p^1 + \theta S_a^1$, where θ is a scaling parameter to balance topology similarity and attribute similarity. After getting the similarity score for top-k users, we assign a bipartite matching process to obtain the maximum sum of scores. Because the k users only occupy a very small part of the graph, we can exhaustively search all matching pairs to get the optimal result. We set a threshold S_t in order to prevent mismatching, in the case that some seeds in G^L are not seeds in G^R. If the similarity score $S < S_t$, that pair is eliminated.

After matching the seeds of G^L and G^R, we further grow the user mapping based on these seeds. Similarly, each pair of nodes, except the seeds, have a similarity score $S^2 = \alpha S_p^2 + \beta S_a^2$. While the attribute similarity is the same as the one in Equation 7.4, the topology similarity score considers the current and potential matching pairs. Specifically, each node has two sets: $N1$ shows the mapped neighbors, and $N0$ shows





(a) Graph G^L (b) Graph G^R

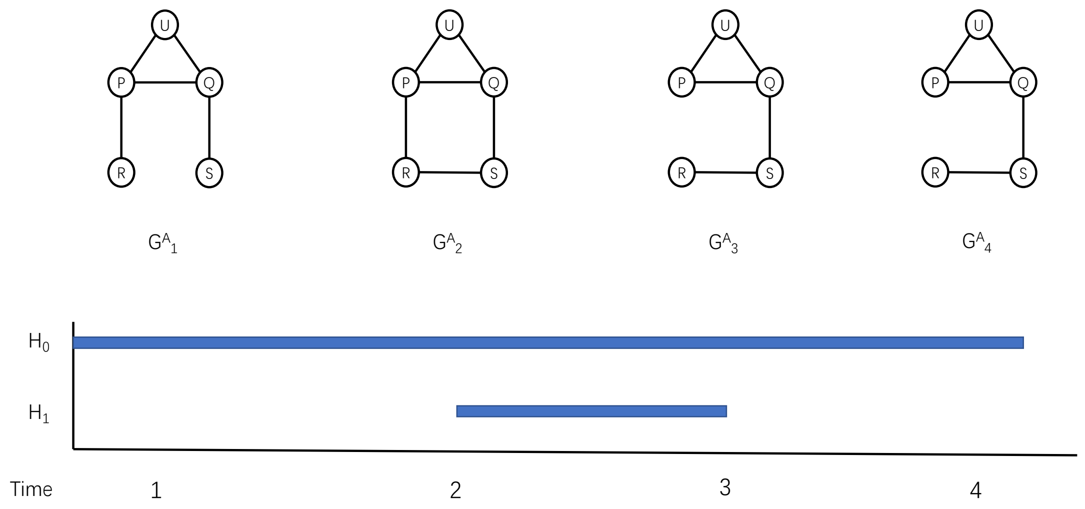
Fig. 7.1.: Example of growing mapping

the unmapped neighbors. The similarity score is given by both the mapped neighbors (with Jaccard index) and unmapped neighbors (with elements count).

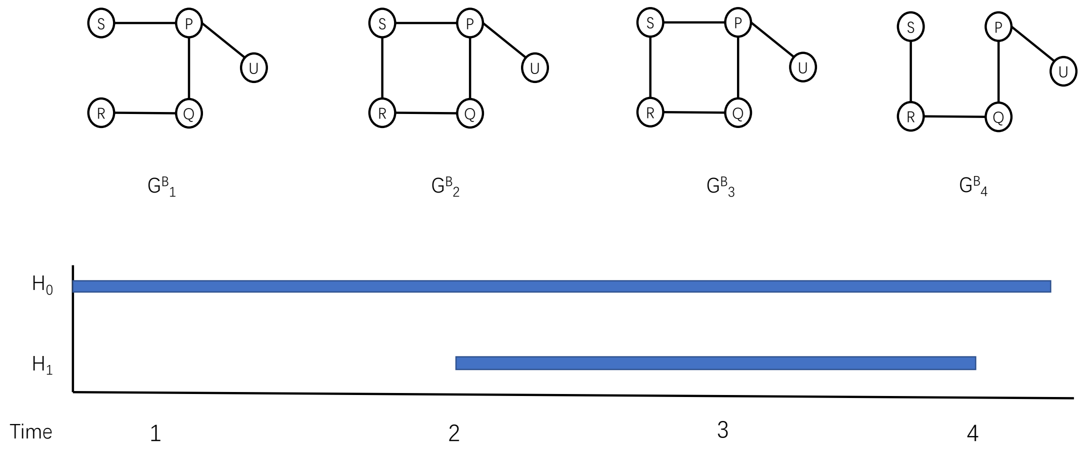
$$S_p^2(u,v) = \frac{N1(u) \cap N1(v)}{N1(u) \cup N1(v)} - \frac{||N0(u)| - |N0(v)||}{\max(|N0(u)|, |N0(v)|)} \tag{7.5}$$

In the example of Figure 7.1, we have two pairs of already mapped nodes, $U - R$ and $P - S$. Consider the topology similarity between Q and T: we have $N1(Q) = \{U, P\}$, $N0(Q) = \emptyset$, $N1(T) = \{R\}$, $N0(T) = \{W\}$. Then the similarity score is $S_p^2(Q,T) = \frac{1}{2} - \frac{1}{1} = -0.5$.

After getting the similarity scores for all pairs of users, we need to do another round of bipartite matching to get the optimal mapping result. However, the searching space is almost all the users (except the seeds), which is much larger than seed mapping. Hence, we implement a heuristic searching method based on the breadth-first-search (BFS) algorithm. We set the seeds as the first layer of the tree to do BFS. The intuition of our algorithm is that the users neighboring the seeds should map with each other first to grow the mapping result.



(a) Time-series graph G^A and its barcode



(b) Time-series graph G^B and its barcode

Fig. 7.2.: Example of hole mapping

7.2.2 Persistent structure extracting and mapping

Persistent homology is a utility metric that summarizes the graph in multi-scales. Persistent homology is presented in the form of barcodes [58]. In OSN graphs, a H_1 hole is a polygon with at least 4 sides. A polygon with at least 4 sides implies that all

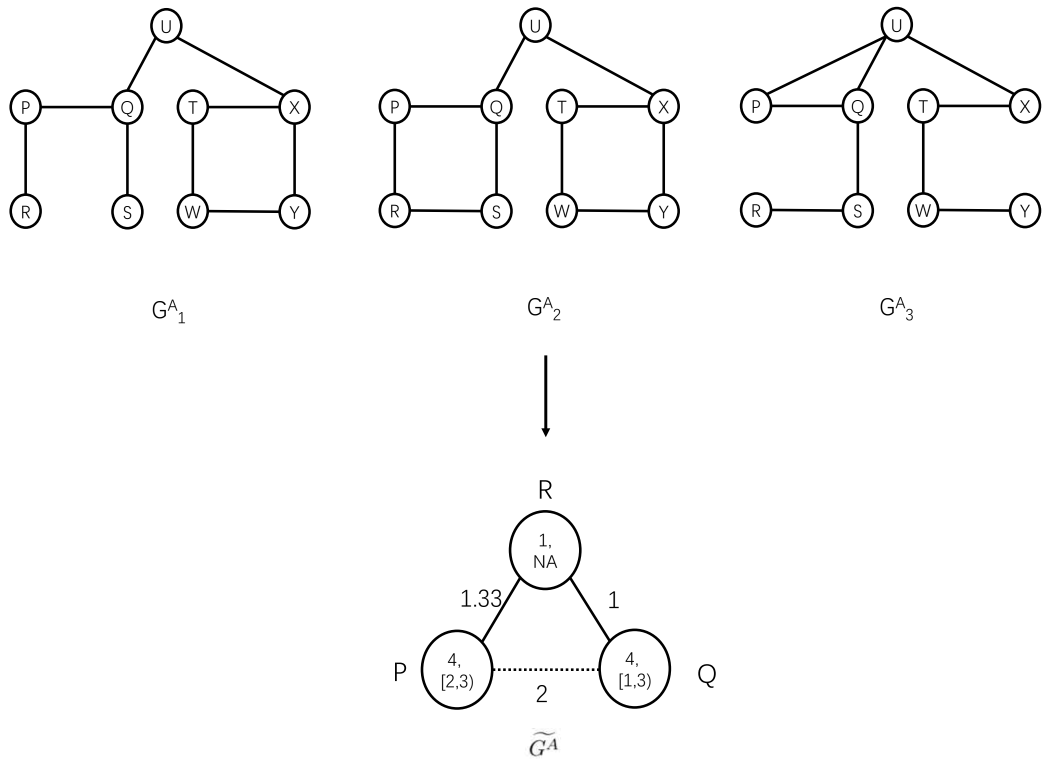


Fig. 7.3.: Example of converted graph $\widetilde{G^A}$

nodes on the polygon have at least one node which is not directly connected, while the triangles have all nodes pair-wisely connected. Persistent homology barcodes can capture the birth time and death time of these polygons.

Figure 7.2 gives a simple example of hole mapping. G^A has a hole from G_2^A to G_3^A. Then its barcode has an H_1 bar $[1, 2)$. G^B has a hole from G_2^B to G_4^B. Then its barcode has an H_1 bar $[1, 3)$. If the two holes match each other, the nodes along the hole, $\{P, Q, R, S\}$, are successfully mapped. Also, the node outside the hole, e.g., U in the example, can find its mapping with the help of the seed nodes on the hole. In the example, the barcode is not perfectly matched. However, this scenario also occurs in real-world cases because the adversaries' relationship building or breaking information may have errors. With the help of persistent homology, we can extract the similarity over several continuous time periods.

Each persistent structure has three important features: the number of nodes involved in the structure, birth time, and death time. When we map the persistent structures, each structure is combined into a super node. We assign a weight w on each edge. There are two kinds of edges in the converted graph: (1) edges between two super nodes, and (2) edges between a simple node and a super/simple node. Hence the weights also have two meanings. On the first kind of edge, the weight shows the distance between two super nodes. On the second kind of edge, the weight shows the connectivity of the two nodes. Finally, we have the converted graph $\widetilde{G^A}$ based on the original graph G^A.

For example, Figure 7.3 shows a converted graph $\widetilde{G^A}$ and its original time-series graph G^A. In the converted graph $\widetilde{G^A}$, the node P with label '$4, [2, 3)$' means that it is a super-node of the persistent structure with 4 nodes, and the persistent structure exists from time 2 to time 3. The node R with label '$1, \mathrm{NA}$' means that it is a simple node. The edge P-Q has a weight $w = 2$ because the two persistent structures have the minimum distance 2 in all static graphs of G^A. The edge P-R has a weight $w = 1.33$, which equals to the total number of edges, which is $(1 + 1 + 2)$, divided by the number of time slots, which is 3. This weight shows that on the average 1.33 edges exist between the two nodes in all time slots.

After getting the converted graph, we map the nodes according to this graph. Specifically, we first map super nodes in this graph as seeds, then we grow the map. When we map super nodes, we temporarily discard all simple nodes and related edges, e.g., solid edges in $\widetilde{G^A}$ in Figure 7.3.

The mapping of super nodes has the following steps:

1. We divide the super nodes into groups according to the first number on their label (which indicates the number of nodes involved).

Algorithm 12 Seed and grow mapping with weight

```
Input: Two weighted graph $\widetilde{G^A}$ and $\widetilde{G^B}$
Output: A node mapping result between users in $\widetilde{G^A}$ and $\widetilde{G^B}$
 1: ——————————Seed mapping——————————
 2: Set $G^L = \widetilde{G^A}$, $G^R = \widetilde{G^B}$,
 3: In $G^L$ and $G^R$, both select $k$ users with highest degree values as seeds,
 4: for each pair of seeds $u$ and $v$ do
 5:     Compute similarity score $S^3$
 6: end for
 7: for each pair of seeds $u$ and $v$ with the highest $S^3$ do
 8:     Exhaustively search mapping results to get $\max S^4$,
 9: end for
10: ——————————Growing——————————
11: for each pair of simple nodes $u$ and $v$ do
12:     Compute similarity score $S^5 = \alpha S_p^5 + \beta S_a^1$
13: end for
14: loop
15:     Pick a node $u$ with the BFS algorithm,
16:     Find the best match nodes $v$ with $\max S^5$
17: end loop
```

2. For nodes in the same group, we calculate the dissimilarity, which equals the differences among birth times and death times. We have,

$$S^3 = \begin{cases} -\infty & \text{if } birth_{G^A} > birth_{G_B}, \\ & \text{or } death_{G^A} > death_{G_B} \\ -(\Delta birth + \Delta death) & \text{otherwise} \end{cases} \tag{7.6}$$

Then each possible mapping pair has a similarity score.

3. Begin with a mapped pair with the highest S^3, we iteratively try to map other nodes to get the maximum S^4, so we have,

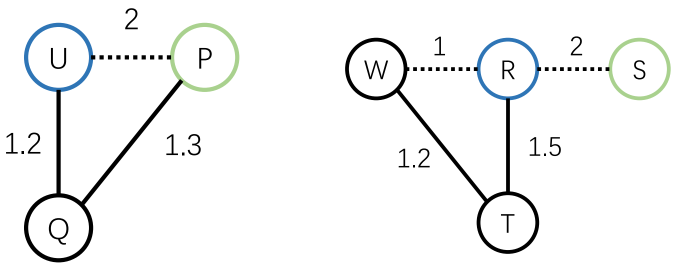
$$S^4 = \sum_{all\ pairs} S^3 \cdot \exp^{-w} \tag{7.7}$$

where w is the distance weight.

4. Then we change the initial mapping of step 3 to another pair, choosing the pair with the second highest S^3, as we get the maximum S^4. We repeat this step and record the best initial mapping and the following mapping.

Because the persistent structures with different sizes have a low probability to of showing the same group of users, we divide the super nodes into groups in step 1. Step 2 ensures that two persistent structures have a probability of mapping together when they have similar birth times and death times, but the existence periods are not required to be exactly the same. Hence, even if some relationship information is not expediently collected by the adversary, our mapping algorithm still has the chance to map the persistent structures together. Note that if the adversary has incorrect information of edge addition or deletion, which happens in the true case of an OSN, we do not map the persistent structures together.

The intuition behind step 3 is that we need to take the distance between persistent structures into consideration. The farther the distance, the less impact the structure has upon central mapping. However, the best S^4 calculated in step 3 is only meaningful to the initial mapping. It may be locally optimal result. Hence, in step 4 we iteratively change the initial mapping pair and get the best mapping result. In real cases, the initial mapping pairs may be the persistent structures with the same sizes and existence periods. So we need to test all possibilities when initial mapping has $S^3 = 0$. Although the final result may not be the globally optimal result, different initial pairs help our heuristic algorithm get better performance without an exhaustive search.



(a) Graph G^L (b) Graph G^R

Fig. 7.4.: Example of growing mapping with weighted edges

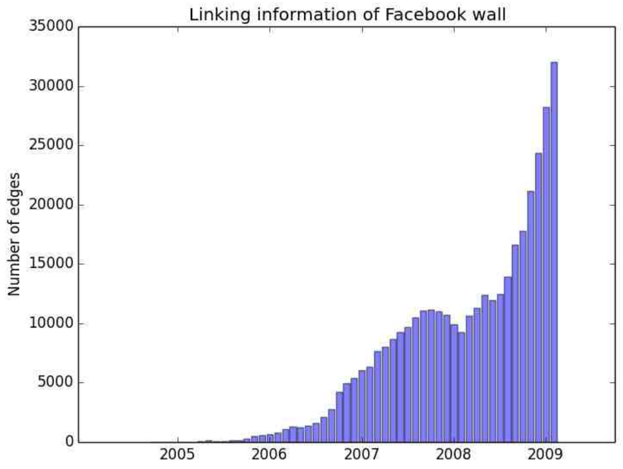
7.2.3 Match growing

In the match-growing process, we need to map the simple nodes with the help of persistent structures. Moreover, we need to differentiate the nodes inside each persistent structure. First, the whole graph $\widetilde{G^A}$ should be recovered, which means the simple nodes, solid edges, and the connectivity weights are back.

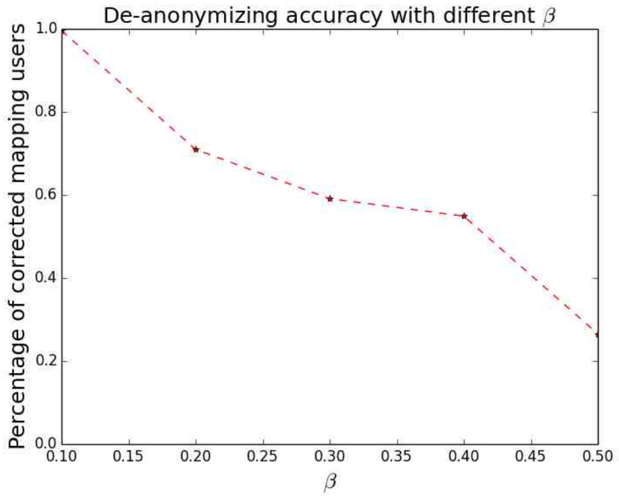
Compared the match-growing process discussed in Section 7.2.1, the match growing here needs to consider the weights, i.e., the connectivities, between the simple nodes and the super nodes. Each pair of simple nodes has a similarity score $S^5 = \alpha S_p^5 + \beta S_a^1$. The attribute similarity is the same as the one in Equation 7.4, and the topology similarity is given by,

$$S_s^5(u,v) = -\frac{\sum_{N(u),N(v)} \Delta w}{\sum_{N(u)} w + \sum_{N(v)} w} - \frac{||N0(u)| - |N0(v)||}{\max(N0(u),(N0(v)))} \tag{7.8}$$

The first item, based on Δw, shows the dissimilarity of weights between mapped seeds. The second item is the same as the one in Equation 7.5, which shows the unmapped neighbors. For example, in Figure 7.4, we have two pairs of already mapped super nodes, $U - R$ and $P - S$. However, when calculating S_p^5 between nodes Q and T, only the pair $U - R$ is mapped seeds. Δw between U and R is

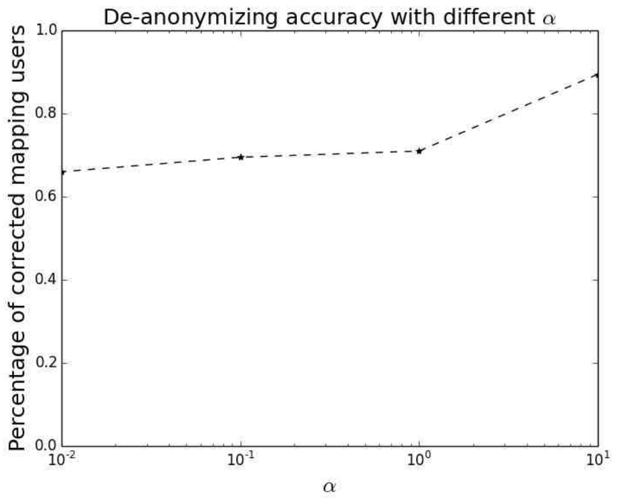


(a) Number of edges in each month



(b) Mapping accuracy with different β, $\alpha = 1$

$d_{RT} - d_{UQ} = 0.3$. Moreover, Q and T also have unmapped or potential mapped neighbors. Then $\sum_{N(u),N(v)} \Delta w = (d_{RT} - d_{UQ}) + d_{PQ} + d_{WT} = 2.8$, $\sum_{N(u)} w + \sum_{N(v)} w = (d_{UQ} + d_{PQ}) + (d_{WT} + d_{RT}) = 5.2$, so we have $S_p^5(u,v) = -\frac{2.8}{5.2} = -0.54$



(c) Mapping accuracy with different α, $\beta = 0.2$

Fig. 7.5.: Evaluation result

7.3 Experiment

In this section, we evaluate our de-anonymization algorithm with a real-world dataset, Facebook wall network [86, 141]. This dataset collects users' posts to other users' walls on Facebook from 2005 to 2009. The nodes of the network are Facebook users, and each edge means one post. Since users may write multiple posts on a single wall, the dataset collects each post and its timestamp. Figure 7.5(a) shows the number of edges in each month.

In our evaluation, we consider a post as a linking relationship between two users. And we combine time slices into time periods. If two users do not post anything between one time period, we consider their relationship to be broken in this time period. Here we set the length of time period to three months. Then, the dataset contains 17 time periods, which means 17 static OSN graphs. There are 46k users and 274k edges in the dataset among all time slices. We combine the whole dataset into a dynamic graph to capture the birth and death information of persistent structures.

Figure 7.5(b) and (c) show the mapping accuracy of our de-anonymization algorithm. When we have a fixed $\alpha = 1$, we get the highest mapping accuracy when $\beta = 0.1$, and the mapping accuracy is 99.58%. When β equals 0.2, 0.3, 0.4, and 0.5, the mapping accuracy is 70.98%, 59.11%, 54.90%, and 26.56%, respectively. We find that the error amount in adversaries' background knowledge largely impacts the de-anonymization ability of the adversaries. If the adversaries can capture the true information of edge addition/deletion, our algorithm is able to capture the persistent structures and de-anonymize the users.

When we have a fixed $\beta = 0.2$, we get the highest mapping accuracy when $\alpha = 10$, and the mapping accuracy is 89.44%. When alpha equals 0.01, 0.1, 1, and 10, the mapping accuracy is 65.99%, 69.49%, 70.98%, and 89.44%, respectively. We find that the latency of the adversaries' ability to capture information does not affect the mapping accuracy very much. Since most of the late edge addition/deletion in G^B will be corrected in the following time periods, our algorithm checks similarity between persistent structures in different time periods. Unlike other algorithms, which are required to precisely map edges in each time, our algorithm has the ability to capture the similarities among different time periods. Hence, our algorithm is robust to late edge addition/deletion.

7.4 Conclusion

In this chapter, we propose a new de-anonymization algorithm to deal with the dynamic OSNs. We introduce the persistent structures to capture the edge addition/deletion among different time periods. Our algorithm can map two similar persistent structures without having the same edge building time. The evaluation result shows the effectiveness of our algorithm, especially when the adversaries have less incorrect information.

This chapter analyze the de-anonymization performance when the error and latency challenges the attacker. However, the attacker may also face other challenges. One major problem is that the entities in the published data are different from the target persons. Previous researchers proposed some schemes to de-anonymize the data when the two group of entities have small non-overlap. In the following chapter, we may discuss the de-anonymization attack when there is dramatical difference between the two groups and/or the attacker has obvious error information.

8. DE-ANONYMIZATION WITH NOVEL METHOD - GENERATING-BASED ATTACK

Several problems may challenge the de-anonymization performance, including latency, error, and non-overlap of users. Chapter 7 and many existing attacks focus on the first and the second problems. However, these attacks are based on the assumption that the users in the published data have an approximated one-to-one mapping with the target users. If the assumption is not true, existing attacks which are based on mapping may have bad performance. This chapter studies the second and the third problems with a different attack route.

In this chapter, we introduce machine learning methods to analyze the graph structures. While existing de-anonymization mechanisms collect information by mapping users from adversary's background knowledge to published data, the proposed scheme directly generates a graph containing the link information and attribute information of targets. In particular, the property of the Generative Adversarial Network (GAN) ensures that the generated graph is undistinguishable with the published graph. The adversaries' background knowledge is embedded as conditional information into the GAN. We also adopt the specially designed graph auto-encoder and graph neural network to extract graph features. The evaluation on real-world OSN datasets proves that the new de-anonymization scheme can generate new graphs similar to the original OSN, de-anonymize the edge information with high accuracy, and enhance the existing user de-anonymization schemes.

The major technical contributions of this chapter are the following:

1. Unlike existing mapping-based attacks, we raise a new idea, which is to generate the graph in a de-anonymization attack.
2. We apply the CGAN model in new graph generation. We extract graph features and feed them as conditions into the CGAN.

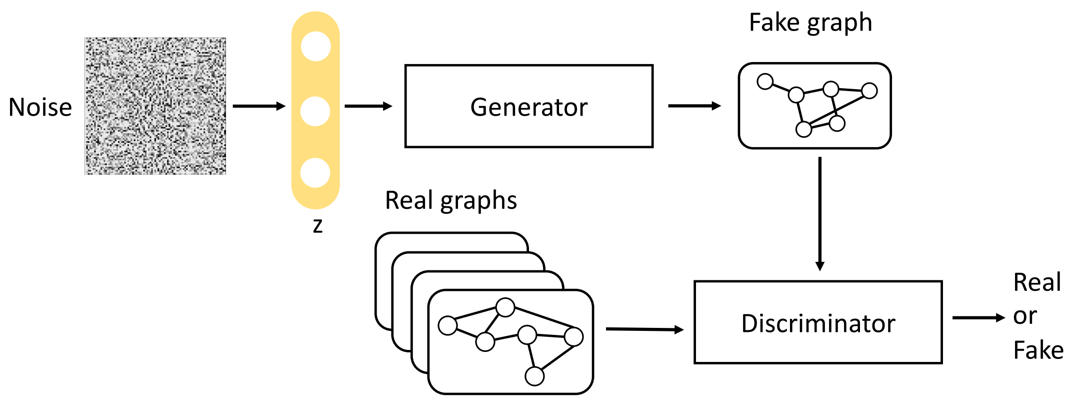


Fig. 8.1.: Structure of GAN

3. We carefully design the generator and the discriminator model to make it appropriate with the graph format data.

8.1 Preliminaries

In this chapter, the OSN is modeled as a graph $G = (V, E, A)$. V is the set of vertices and each vertex is a user in the original OSN. E is the set of edges and each edge is a relationship between two users in the OSN. A is the set of attributes and each attribute is an entry in a user's profile, e.g., name, age, gender. Before attacking, the adversary holds two graphs, one is his/her background-knowledge graph G_b concluded by the adversary himself/herself, the other is the published graph G_p retrieved from the OSN service provider or third parties.

In order to make up the incomplete information in G_b, we apply the GAN model to build a new graph. GAN is a deep neural network architecture comprised of two networks, a generator and a discriminator [62]. The detailed structure of GAN is shown in Figure 8.1. The generator is the neural network that generates a synthetic sample from the input noise vector z. The discriminator is the neural network that discriminates the generated sample with the real sample x. Typically, the optimization goal for the generator is to minimize the probability that the generated (fake) samples are caught by the discriminator.

$$\min_{Gen} E_{z \sim p_z}[log(1 - Dis(Gen(z))], \tag{8.1}$$

where Gen is the generator model and Dis is the discriminator model. p_z is a random noise distribution.

The optimization goal for the discriminator is to maximize the probability that real samples are classified as real, and fake samples are classified as fake.

$$\max_{Dis} \{E_{x \sim p_{real}}[\log Dis(x)] + E_{z \sim p_z}[log(1 - Dis(Gen(z))]\}, \tag{8.2}$$

where p_{real} is the distribution of real data. In the OSN de-anonymization problem, p_{real} is retrieved from the published graph G_p.

Then, we can combine the two goals, from generator and discriminator, into a minimax game [62]. The overall optimization goal is,

$$\min_{Gen} \max_{Dis} \{E_{x \sim p_{real}}[\log Dis(x)] + E_{z \sim p_z}[log(1 - Dis(Gen(z))]\}. \tag{8.3}$$

While the generator wants to minimize the objective function, the discriminator wants to maximize it.

In GAN, the generator and the discriminator compete with each other in a zero-sum game framework. After the overall architecture gets coverage, the two players are expected to achieve a Nash-equilibrium [130]. It should be hard for the discriminator to differentiate the generated graphs with real graphs. Moreover, the output graph should contain more information than G_b, which enhances an adversary's knowledge.

8.2 Problem statement

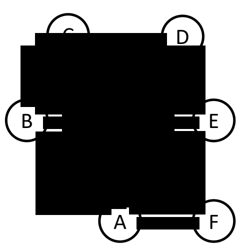
The main purpose of this chapter is to design a new de-anonymization method, which allows the adversary to discover the true linking relationships among the target area. The tools of the adversary are the published graph data and some knowledge of the target users. For example, in a company, the attacker wants to know which users are friends on Facebook. Facebook service providers may release some network data to third parties, which eventually becomes accessible to the attacker. There are three different scenarios of published data:

1. The published data contains all the target persons. However, after ID removal, the attacker needs to find a mapping between the published data and the target person. Moreover, the attacker only knows part of the linking relationships and attributes, which makes the mapping difficult.
2. The published data partially covers target persons in the company. The attacker needs to carefully discover which users are not in the published data.
3. The published data is not about the target persons at all. The attacker needs to infer linking relationships and attributes from other parts of the OSN.

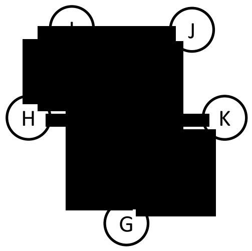
While most existing de-anonymization schemes focus on the first scenario, some schemes can handle the second scenario but have limited performance [136]. The third scenario is a new problem to de-anonymization attacks. Usually, the attacker has no idea which scenario characterizes the published data. It is not realistic to restrict the problem. Hence, in this chapter, we design a universal scheme which is suitable to all scenarios.

8.3 Scheme

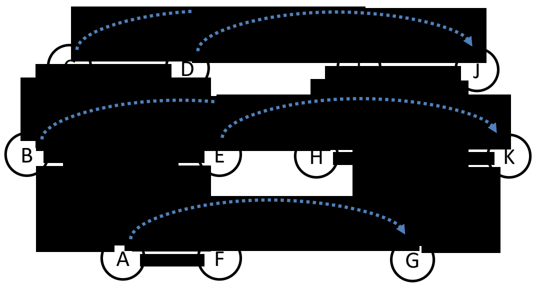
Before introducing our scheme, we would like to give a comparison between the mapping-based attack and the generating-based attack, with the example shown in Figure 8.2. In both of the two types of attacks, the input data is the published graph G_p and the background-knowledge graph G_b. Because the information in G_b is very limited, the attacker can use the information in G_p to greatly improve his/her knowledge about the target users.



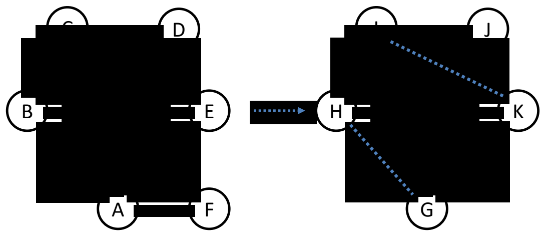
(a) G_p



(b) G_b



(c) Mapping-based attack



(d) Generating-based attack

Fig. 8.2.: Example of two attacks

In mapping-based de-anonymization schemes, the attacker focuses on mapping the nodes in G_p with the nodes in G_b. In Figure 8.2(c), the attacker first builds mapping links between the two graphs, e.g., A-G and B-H. Then, the attacker knows there should be edges between G-H and I-K in the background-knowledge graph. This information is the result of the de-anonymization attack.

In generating-based de-anonymization schemes, the attacker focuses on building a new graph. The new graph should contain information both in G_b and G_p. In Figure 8.2(d), the generated graph is shown on the right side. Although the attacker does not map users, he/she can still obtain the knowledge that there are edges between G-H and I-K. In conclusion, the two de-anonymization schemes can have the same result in this example.

However, there is still a big challenge, to inject information from G_b and G_p into the generated graph. In this scheme, we deploy a CGAN model to deal with the two kinds of information. Particularly, the information in G_p is marked as real data when training the GAN structure. The information in G_b is injected as the conditional information to the CGAN model. In the following sections, we will give technique details about the following aspects.

1. CGAN structure: We give the overview of the structure of our CGAN model.
2. Background knowledge data embedding: We apply the CGAN model to inject the conditional information.
3. Published data embedding: We apply mini-batch against model collapse.
4. Graph classifier: We apply Graph Neural Network (GNN) and Graph Auto-Encoder (GAE) to collect the graph information.

8.3.1 CGAN structure

In this section, we give the overview of the de-anonymization process in our scheme, which is based on the CGAN structure. Similar to the description of GAN in Section 8.1, the CGAN structure also has two components, the generator and the discriminator. In this section, we separately introduce the two parts in our scheme to clarify our design.

The structure of the generator is shown in Figure 8.3. Typically, the input to the basic generator is a noise vector z, which is an independent and identically distributed noise vector drawn from the uniform distribution $[0, 1)$. However, in this chapter, the

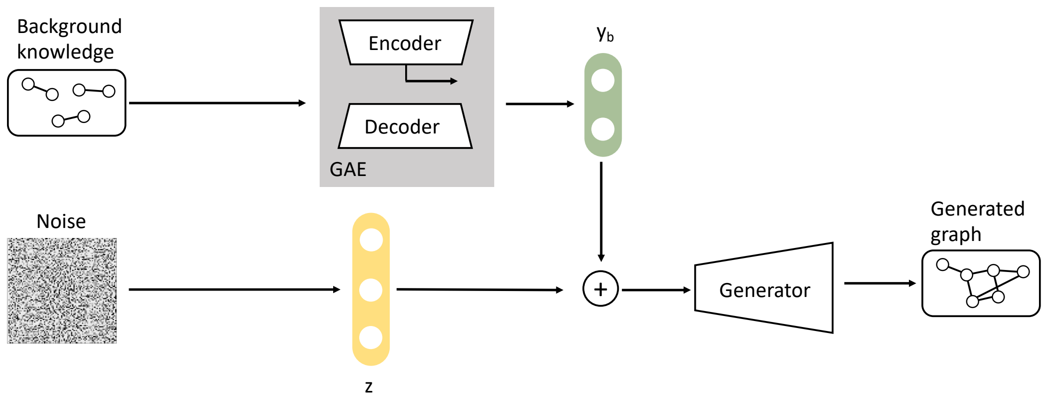


Fig. 8.3.: Generator structure

scheme also adopts the background knowledge of the adversary as the input to the generator. The background knowledge first composes the graph G_b. Then G_b is fed into the GAE, an encoder-decoder structure. The encoder in the GAE is utilized to derive the vector representation y_b of the background knowledge. GAE training takes the whole published graph as the input since it is the most complete graph the adversary holds. After training, GAE embeds graph structural information into the graph vector. Then the adversary collects the graph vector as well as the noise vector, and feeds them together into the generator to get the generated graph.

The structure of the discriminator is shown in Figure 8.4. The input of the discriminator can be divided into two categories, the real samples and the fake (generated) samples. The adversary first divides the published graph into a group of subgraphs. Each subgraph has a number of nodes equal to the node number of the target area. Then the adversary samples the subgraphs with his/her estimation of background knowledge. For example, if the adversary thinks he/she can collect 50% of the edges, then he/she samples 50% of the edges from the published graph to build the condition-information graph.

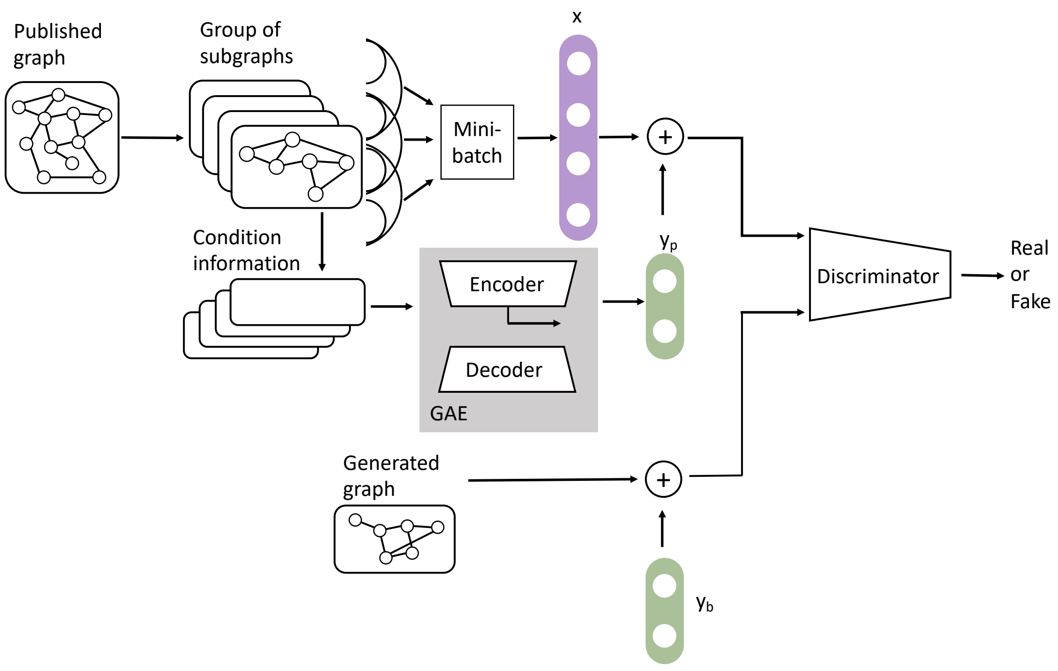


Fig. 8.4.: Discriminator structure

Thereafter, the same well-trained GAE is applied to get the graph vector. For each real graph, GAE here derives the vector representation y_p of the sampling of the published graph. For each fake graph, GAE in the generator has already derived the vector representation y_b of the background knowledge graph. The graph and its corresponding conditional vector are combined together and fed into the discriminator. Before feeding, the scheme also deploys the technique called mini-batch to avoid model collapse. We combine the feeding data into small batches, so that the generated graph batch should have a similar distribution with the input real graph batch.

The purpose of the generator is to fool the discriminator that all generated graphs are real, while the purpose of the discriminator is to correctly discriminate real or fake graphs. After the two networks are well trained, our scheme utilizes the generator to generate new graphs. The graph contains information both from G_p and G_b.

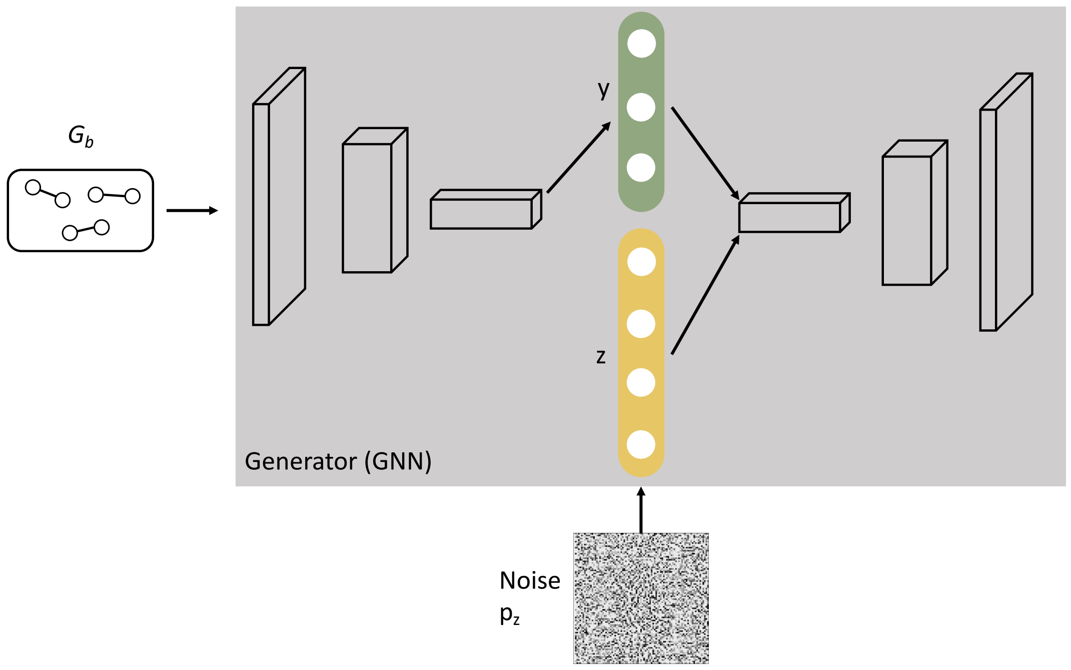


Fig. 8.5.: Generator with conditional information

8.3.2 Background-knowledge data embedding

Embedding the attacker's background knowledge into the final generated graph is one of the most important parts of the de-anonymization scheme. If the final generated graph contains no specific requirements about the target area, one may find it is easy to generate graphs similar to the published graph. For example, every subgraph cut from G_p is 'similar' to the graph G_p. However, generating edges fully reliant upon other parts of the published graph is nonsense to the adversary. The purpose of the adversary is to collect the information in the target area, not a subgraph that can show the statistical properties of the network.

There are several methods to inject the background information, which is not only the restriction but also the critical input, into the generated graph. Considering the generation process in GAN, we introduce three different approaches in the scheme.

Generation from both noise and information. In a traditional GAN framework, as demonstrated in Figure 8.1, the input of the generator is a noise vector z from the prior distribution p_z. The randomness introduced by z improves the robustness of the generated data, because we want the GAN to generate similar data instead of directly duplicating the input. Then, there is a direct approach in which we generate the data from not only the random noise, but also the prior information.

Following this idea, we modify the structure of the generator so that it is appropriate with G_p. The new structure is shown in Figure 8.5. Because the size of the vector representation of G_p is significantly larger than the size of the noise vector z, directly adding G_p may reduce the impact of z, which will decrease the robustness of the generated results. Hence, we first apply a GAE to calculate the vector projection y_1 for G_p, whose detailed structure is introduced in Section 8.3.4. After that, we combine the two vectors, y_1 and z, together, as the input to the generator. The new objective function of the GAN is given by,

$$\min_{Gen} \max_{Dis} \{E_{x \sim p_{real}}[\log Dis(x)] + E_{z \sim p_z}[log(1 - Dis(Gen(z|y_1))]\} \tag{8.4}$$

Punishment in discriminator. Besides injecting information in the generation process, we can also inject the prior knowledge in the discrimination process. A direct approach is to assign a big loss value if the generated graph does not have the information specified in G_b. However, when training the discriminator, we also need to add such punishment. Specifically, the adversary should build a synthetic G_b for each subgraph of G_p before training. The synthetic G_b is built upon the attacker's estimation of his/her power to collect information, e.g., 50% edges and attributes. If the final generated graph does not cover the information in G_b, the loss of generation is increased, whenever it is the training phase with the synthetic G_b or it is the testing phase with the real G_b.

Similarly, we denote the vector representation of the synthetic prior knowledge as y_2. When the discriminator classifies fake or real graphs, it considers the real graph samples x, the generated graph, and the (synthetic) prior knowledge y_2. The new objective function of the GAN is given by,

$$\min_{Gen}\max_{Dis}\{E_{x\sim p_{real}}[\log Dis(x|y_p)] + E_{z\sim p_z}[log(1 - Dis(Gen(z))]\} \tag{8.5}$$

Conditional generative adversarial network. As described above, we analyze the two information injection methods dealing with the generator or the discriminator, respectively. Afterwards, we consider applying a combined model to impact both the generator and the discriminator. CGAN is first introduced to generate objects conditioned on class labels [101]. CGAN has been widely adopted in domain transfer, e.g., from drawing and sketching to photos and from text to images [72, 158, 160]. Similar to the image translation from sketched images to real photos, the prior knowledge graph G_b is another kind of sketch to the original graph, and our de-anonymization purpose is to regenerate the original graph. Hence, CGAN is appropriate for application to our OSN de-anonymization scheme.

The structure of CGAN is shown in Figure 8.6. To each piece of input data, including the noise vector, generated data, and real data, we give it a condition (label). After this CGAN model is well trained, it can generate new samples under the specific condition. In the OSN de-anonymization problem, the condition is related with the prior knowledge. Thus the CGAN model can generate graphs according to adversary's knowledge.

The method to extract the conditional vectors y_b and y_p is similar to the method we discussed in the previous two approaches. In the generator part, G_b is transformed into y_b with the help of GAE. In the discriminator part, the adversary first estimates his/her power to collect information. Then this adversary samples the attributes and links to generate the subgraph. Finally the CGAN applies the same GAE to extract y_p from the published graph. The objective function of the CGAN is given by,

$$\min_{Gen}\max_{Dis}\{E_{x\sim p_{real}}[\log Dis(x|y_p)] + E_{z\sim p_z}[log(1 - Dis(Gen(z|y_b))]\} \tag{8.6}$$

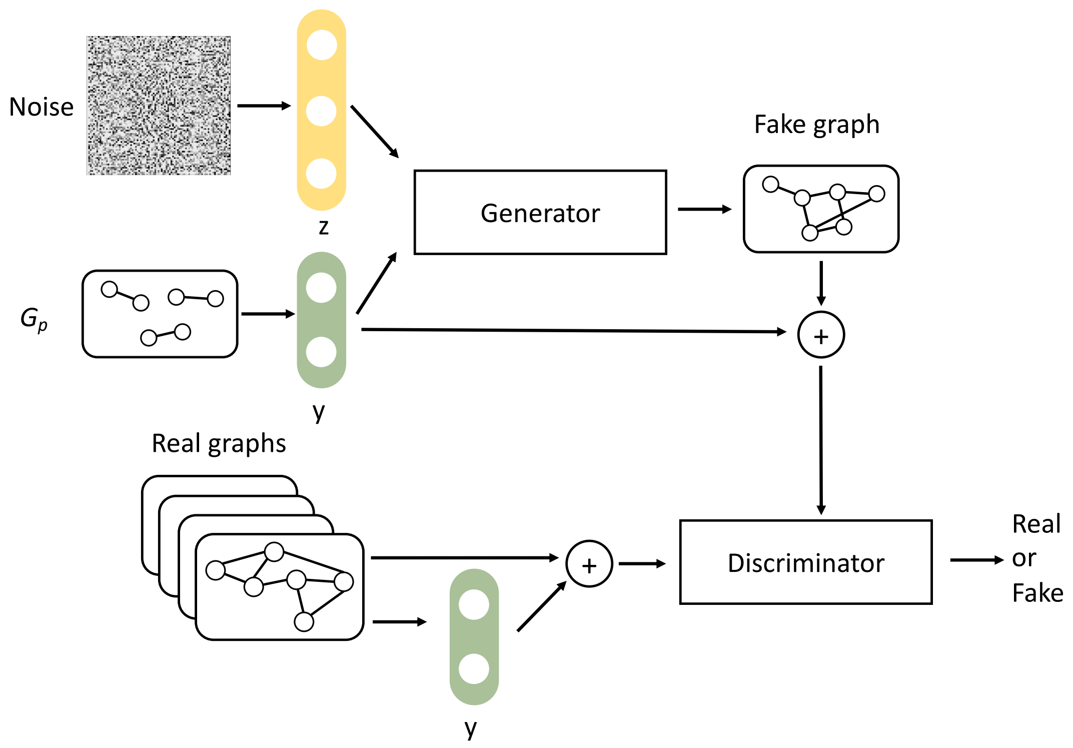


Fig. 8.6.: Conditional generative adversarial network

8.3.3 Published data embedding

In our CGAN model, the real samples are retrieved from the published graph G_p. Although the basic GAN architecture has the design to feed real samples, directly using G_p as the input may cause several problems, including graph size mismatch and GAN model collapse.

Graph size mismatch. Usually the attack target graph only contains several users, but the published graph may have thousands of other users. When the output graph tries to imitate the input graph, the size mismatch may inject unnecessary information into the output graph. Some properties that only exist in large OSNs may also be injected into the small output graph. In the proposed scheme, the method to deal with the size mismatch problem is subgraph sampling. Assuming the adversary has n_t users as the target user, he/she samples a group of subgraphs from G_p, where each subgraph contains n_t nodes.

GAN model collapse. The aim of training GAN is to find a Nash equilibrium between the two-player game. However, it is a non-convex problem and the two neural networks both have high-dimensional parameters. Hence, researchers typically trained the GAN with some gradient descent methods, which search the minimum value of the loss function [130]. These methods may cause the failure of GAN models when the parameters of the generator collapse to a specific vector. Then the generator always produces the same results. The collapse problem happens because of the imbalance between the generator and the discriminator. When the generator has relatively lower power, the noise inputs do not result in different outputs. Instead, the generator is forced to output samples that are highly similar to one of the real samples.

In the OSN de-anonymization problem, our purpose in applying GAN is to generate a graph that contains the properties of the input graphs. When the input is a group of subgraphs, we want the output to be a graph similar to all these graphs, instead of restricted to one of the samples. Hence, the proposed scheme introduces the mini-batch method to defend against the model collapse problem [130]. The mini-batch method combines real graphs, as well as generated graphs, into small batches. Rather than isolated feeding, we feed a batch of graphs at a time into the discriminator. When the real graphs in a batch are expected to be diverse, the batch of generated graphs would be classified as a fake batch if the graphs lack diversity. Instead of training a GAN to duplicate some specific inputs, applying mini-batch ensures the GAN learns the distribution of real graphs, and outputs the generated graphs following that distribution.

8.3.4 Graph classifier

In this section, we need to design the classification model for graphs. Specifically, there are two problems in this area: Which information (features) should be extracted from the graph? What is the structure of the classification model?

Input information. The information in an OSN graph or other kinds of networks can be divided into two categories: structural information and attribute information. To represent structural information, the adjacency matrix is one of the most direct and simple formats. The other representation of graph structure, including latent features, random walk vectors, and Katz index, are correlated with each other, and they are all based on the adjacency matrix. For example, we can directly extract the PageRank score matrix from only the adjacency matrix. Hence, we only feed adjacency matrix into the graph classifier to reduce input redundancy.

Another reason for choosing the adjacency matrix is that our scheme needs to generate the graph with incomplete knowledge. When the adversary only holds partial edge information in his/her prior knowledge, the loss of information may greatly influence these high-level feature results. For example, the adversary can hardly apply the random walk algorithm when 50% of edges are missing. On the contrary, the adjacency matrix directly shows the impact of adding or deleting one edge, which simplifies the structure of our learning process. Therefore, to a graph with $|V|$ users, our scheme uses a $|V| * |V|$ adjacency matrix to represent the structural information, and a $|V| * |A|$ matrix to represent the attribute information. $|A|$ is the number of attributes, which includes gender, age, location, etc., in OSNs.

Classification model. Generally speaking, we design three kinds of classifier, the auto-encoder, the generator, and the discriminator. Both the auto-encoder and the discriminator take a graph, i.e., the $|V| * |V|$ adjacency matrix and the $|V| * |A|$ attribute matrix, as the input.

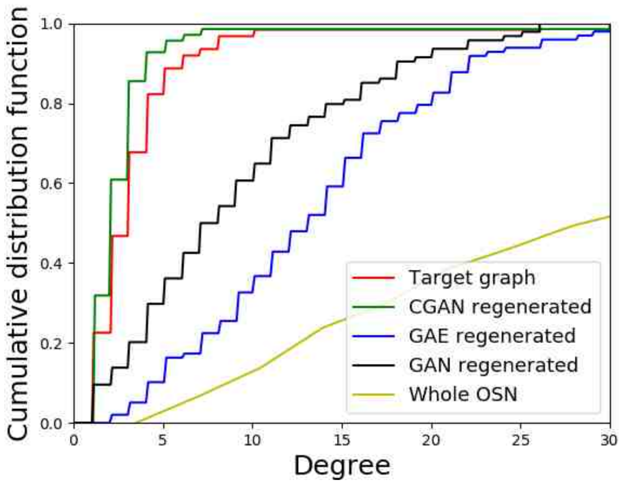
The structure of the discriminator follows the research on Graph Neural Network (GNN) to learn over graphs. A GNN typically contains graph convolution layers and graph aggregation layers. The convolution layers quickly extract local substructure features for each individual node, without designating weights to each neuron. The aggregation layers aggregate node-level features together. After summarizing the node-level features, the aggregation layer gets the graph features and finally outputs the classified results, i.e., fake or real.

The structure of the generator reverses the structure of the discriminator. Specifically, the generator first has several reversed-aggregation layers. These layers are fully connected with each other to generate graph features. Then, the graph features are fed into the de-convolution layers to transform the graph-level features to node-level features. Finally, the generator outputs a generated $|V| * |V|$ adjacency matrix and a $|V| * |A|$ attribute matrix.

The structure of the Graph Auto-Encoder (GAE) is similar to the discriminator because they have similar input. GAE also has some convolution layers to extract the substructure features. However, GAE should be an unsupervised learning model to learn these substructure features, and these features are the final output. Specifically, we follow the design of GAE described in [83]. The GAE first has an encoder to extract the features, then it also has a decoder to regenerate the graph based on the extracted features. Here, the decoder has the inverse structure of the encoder, which is also some convolution layers with reversed dimensions. The loss of the encoder-decoder model is the dis-similarity between the original graph and the generated graph,

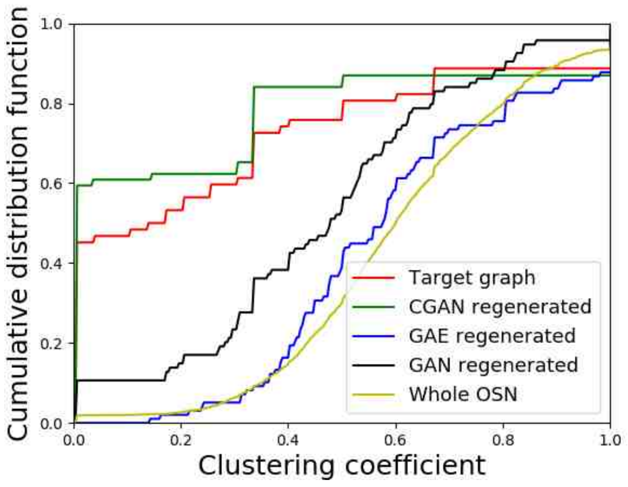
$$Loss = E_{q(x'|y)}[logp(y|x)] - D_{KL}(q(x'|y)||p(x)) \tag{8.7}$$

where x is the input adjacency matrix and feature matrix, x' is the regenerated adjacency matrix, and y is the extracted substructure features. D_{KL} shows the Kullback–Leibler divergence between two domains p and q: p is the domain of the input graph, and q is the domain of the regenerated graph [85].



Although the GAE and the GAN are both about regenerating graphs, their targets are different. The generator-discriminator structure aims to generate graphs ‘similar’ to the input graph, i.e., the generated graphs have a similar probability distribution with the original graph. The noise in the generator and the mini-batch method forbid the overall system from generating samples exactly the same as a single input graph. However, the encoder-decoder structure aims to copy. The greater the similarity of the generated graph, the more precisely the encoder extracts structure features.

Therefore, the input of GAE does not require diversity between graphs. The process of dividing the graph into subgraphs in training GAN may lead to the GAE having more difficulty in structure learning. In our scheme, we choose to feed the original published graph G_p without any segmentation. To an attacker, the original G_p contains almost all the information about the graph structures. The GAE is first trained with G_p. Then for each graph in G_p or G_b, the attacker applied the trained GAE model to extract the structure feature vector y. Finally, this y is used as the condition in the CGAN model.



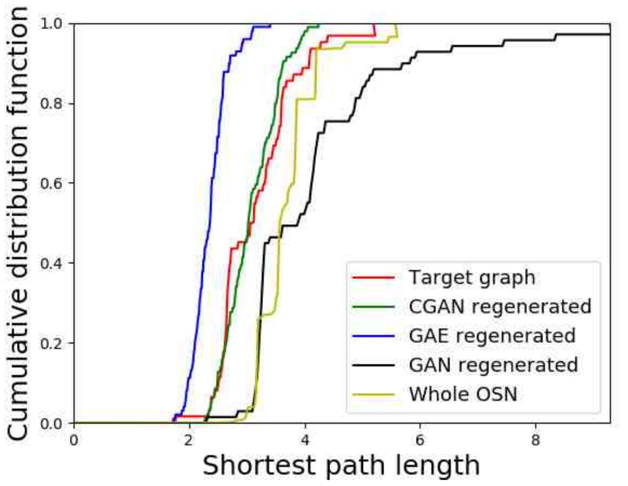
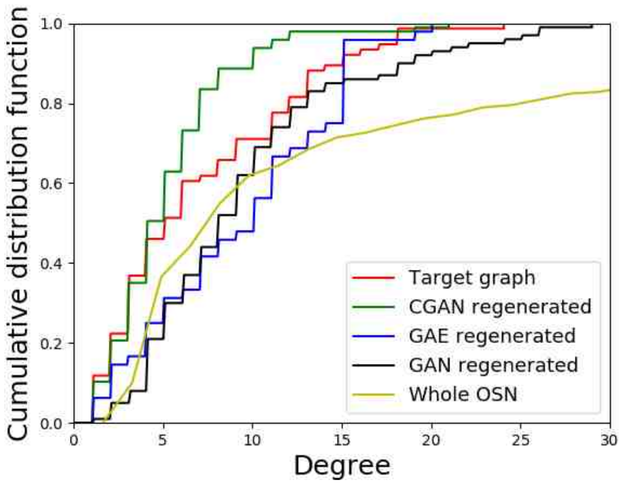
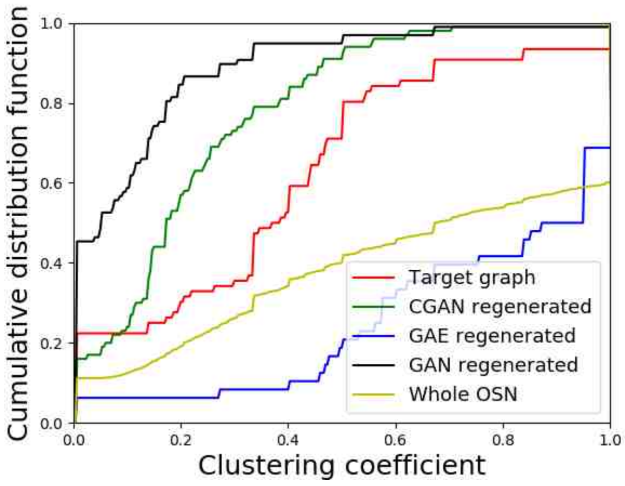


Fig. 8.7.: Utility metrics comparison between regenerated graphs, Facebook

8.4 Evaluation

In this section, we first describe the evaluation settings, including the datasets and CGAN structural design. Then we demonstrate the effectiveness of our de-anonymization attacks under two aspects, similarity and accuracy. In the end, we use a case study to show the capability of our generation based attack when combined with a traditional mapping based attack.





8.4.1 Evaluation settings

Our evaluation is simulated on two real-world OSNs, the Facebook dataset and the ca-HepPh dataset [89]. Facebook is a commercial OSN, while ca-HepPh is a collaboration network showing co-author relationships between authors in the domain of high energy physics.

As described in Section 8.2, the attacker may face three kinds of scenarios. Here we evaluate the performance of our scheme under different scenarios. Firstly, we test the second scenario. The target subgraph and the published graph are sampled

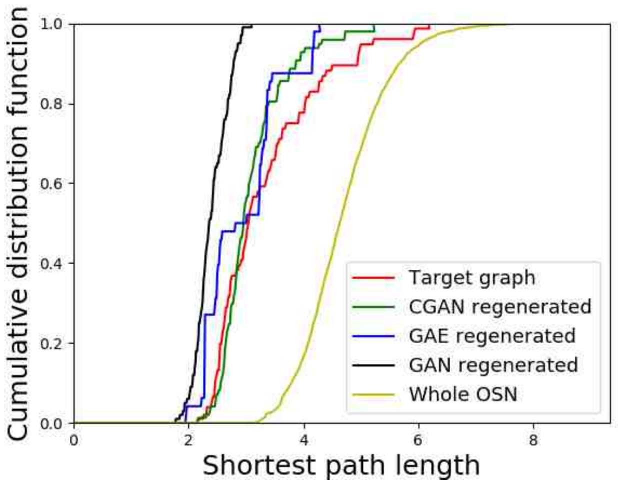


Fig. 8.8.: Utility metrics comparison between regenerated graphs, ca-HepPh

from the same graph, the whole OSN. In particular, we sample a 100-node subgraph and set it as the attacker's target area. The input training graphs are also sampled from the original OSN, each having 100 nodes, as described in graph-size mismatch problems in Section 8.3.3. We sampled 1200 subgraphs in total and fed them into the CGAN to learn the structure of an OSN, e.g., Facebook. Secondly, we test the third scenario. We first cut the whole OSN into two parts, one for target subgraph sampling and one for published graph sampling. Then the same sampling procedure is applied to the two parts of graphs but the only difference is that there is no overlap between the target area and the published graph.

Our CGAN was implemented on TensorFlow based on previous research [62, 122]. The encoder-decoder structure was also built on TensorFlow based on graph auto-encoders [83].

8.4.2 Similarity analysis

The final purpose of our scheme is to regenerate graphs that are similar to the ones in the original OSN, but unseen to attackers. However, it is difficult to directly compare two graphs since there are several edges and nodes. Here we present two methods to compare the regenerated graph with the original graph: one is similarity and the other is accuracy. The first is using several utility metrics that are widely used in comparing the similarities between graphs. The second is considering each pair of nodes as an instance. When there is an edge, it is a true instance, otherwise it is a false instance. In this section, we do the similarity analysis.

Here, we choose three utility metrics, including degree distribution, clustering coefficient distribution, and shortest path length distribution, to show the similarities between two graphs. We compare our CGAN regenerated graph with the original graph (target). For comparison purposes, we also compare our results with two other regeneration methods. The first is generated with GAE, which is the encoder-decoder structure embedded in our CGAN structure [83]. We reimplement the decoder to decode the vector representation and regenerate the graph. The second is generated with GAN, which is the base of our CGAN structure [62]. We apply the most basic GAN structure and do not add any conditional information into it. Since GAN can only learn information from the published graph but no information from the prior knowledge graph, we compare the GAN regenerated graph with the whole published graph, in degree distribution and other metrics. To conclude, there are five graphs for comparison: the whole published graph, the original target graph, the CGAN generated graph, the GAE generated graph, and the GAN generated graph.

All five graphs have 100 nodes (the size of the target area) except the whole published graph. The whole Facebook graph has 4039 nodes and 88234 edges, while the whole ca-HepPh graph has 9877 nodes and 25998 edges. We compare the degree distribution, the clustering coefficient, and the shortest path length. The utility metrics comparison results are shown in Figure 8.7 and Figure 8.8.

In Figure 8.7, the results from the Facebook dataset show that our CGAN regenerated graph is the most similar graph to the target graph, under all three utility metrics. The average degree of the target graph, the CGAN regenerated graph, the GAE regenerated graph, the GAN regenerated graph, and the whole OSN is 3.39, 2.70, 13.43, 9.09, and 43.69, respectively. The average clustering coefficient of the target graph, the CGAN regenerated graph, the GAE regenerated graph, the GAN regenerated graph, and the whole OSN is 0.27, 0.22, 0.60, 0.46, and 0.60, respectively. The average shortest path length of the five graphs is 3.14, 3.06, 2.35, 4.13, and 3.69, respectively.

In Figure 8.8, the results of ca-HepPh dataset also show our CGAN regenerated graph outperforms other schemes. The average degree of the target graph, the CGAN regenerated graph, the GAE regenerated graph, the GAN regenerated graph, and the whole OSN is 6.92, 5.13, 9.17, 9.42, and 21.00, respectively. The average clustering coefficient of the target graph, the CGAN regenerated graph, the GAE regenerated graph, the GAN regenerated graph, and the whole OSN is 0.36, 0.22, 0.75, 0.11, and 0.62, respectively. The average shortest path length of the five graphs is 3.32, 3.08, 2.93, 2.39, and 4.67, respectively.

It is notable that the purpose of the CGAN regenerated graph, the GAE regenerated graph, is to imitate the target graph, while the purpose of the GAN regenerated graph is to imitate the whole OSN. When our CGAN regenerated graph achieves the task of impression, the GAE and the GAN regenerated results are not solid. The problem of GAN may come from the size mismatch between the input graph and the output graph, e.g., 4039 of the whole Facebook dataset and 100 of the regenerated graph. Then it is a bit hard to mimic all the properties of the whole OSN in such a 100-node graph. Fortunately, our CGAN scheme introduces the conditional information to formalize the properties included. The CGAN regenerated graph does not need to cover all properties of the whole graph, but just some specific properties in the subgraphs that have similar conditional information as the target graph.

The problem of GAE may come from its regeneration power. Although previous research demonstrates its encoding power, especially in node categorization and vector representation, the performance of decoding does not have enough proof [83, 159]. From the regenerated graph we can also find that the GAE regenerated graph has properties similar to the whole OSN, e.g., the degree distribution and clustering coefficient distribution of the Facebook dataset. This result implies that GAE can have a good view of the whole network (recall that the training of GAE takes the whole published graph as the input). In this chapter, GAE is also applied to learn the graph structure and extract vector representation. Instead of the encoder-decoder structure, our CGAN scheme is based on the generator-discriminator structure. Invoking the adversary in learning enables the CGAN for better generalization of graph information and better generation of new graphs.

8.4.3 De-anonymization accuracy analysis

As discussed before, de-anonymization accuracy is another factor in demonstrating the effectiveness of our scheme. While existing de-anonymization methods focus on de-anonymizing the user, i.e., correctly mapping the nodes, our scheme focuses on de-anonymizing the relationship, i.e., correctly building the edges. Here we set the original target graph as the ground truth, and compare it with our regenerated graph. If two users have edges in the original graph as well as the regenerated graph, we view this instance as True Positive (TP). We also set the True Negative (TN), False Positive (FP), and False Negative (FN) following a similar rule.

We test the de-anonymization accuracy under two scenarios. Firstly, we test the performance of the second scenario, i.e., the published data partially cover the target graph. We test the recall and accuracy of our de-anonymization attack, which are defined as follows:

$$\begin{aligned} Recall &= \frac{TP}{TP+FN} \\ Accuracy &= \frac{TP+TN}{TP+TN+FP+FN} \end{aligned} \tag{8.8}$$

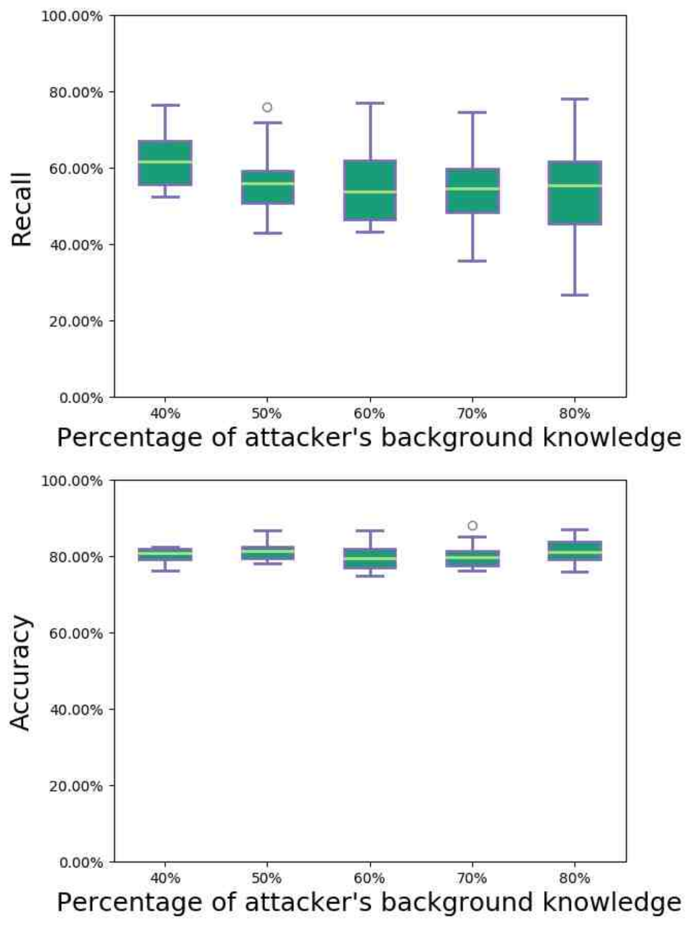


Fig. 8.9.: Recall and accuracy of our edge de-anonymization attack, published data partially covers target persons

The result of this scenario is shown in Figure 8.9, in the format of box plot [151]. When the attacker has 40%, 50%, 60%, 70%, and 80% information about the target graph, the average recall is 0.62, 0.65, 0.64, 0.73, and 0.71, respectively; the average accuracy is 0.87, 0.89, 0.88, 0.89, 0.89. These results demonstrate the effectiveness of our de-anonymization attack. When the scheme maintains high accuracy (nearly 90%), the recall keeps higher than 60% and it increases when the attacker has more information about edges.

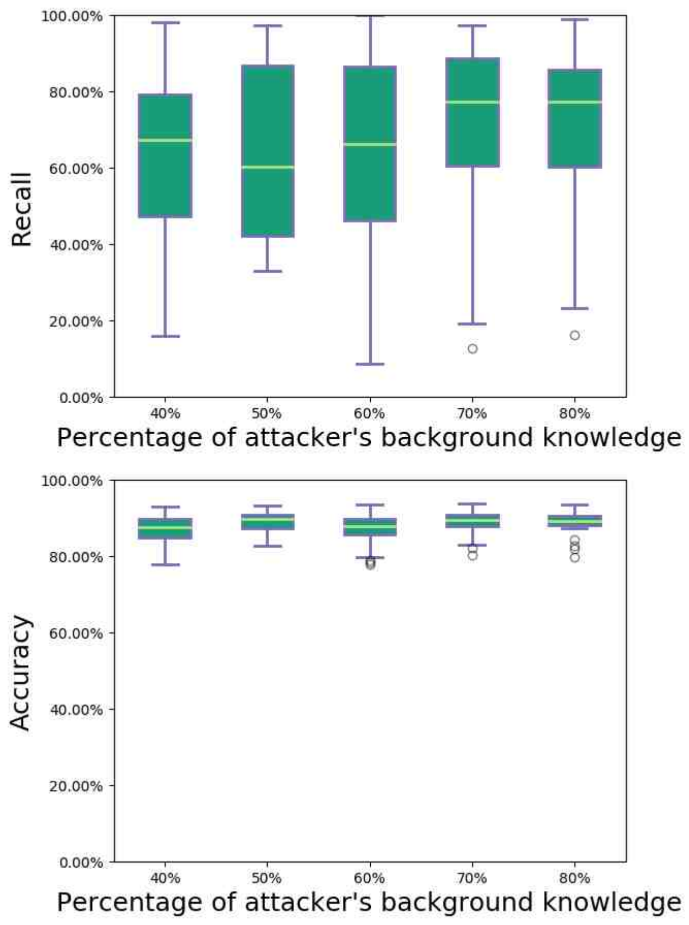


Fig. 8.10.: Recall and accuracy of our edge de-anonymization attack, published data does not cover target persons

Secondly, we test the performance of the second scenario, i.e., the published data does not cover target persons. Under this scenario, we separate the whole OSN into two parts, training and testing, at the very beginning. Similarly, we test the recall and accuracy, and the results are shown in Figure 8.10. When the attacker has 40%, 50%, 60%, 70%, and 80% information about the target graph, the average recall is 0.63, 0.56, 0.56, 0.55, and 0.55, respectively; the average accuracy is 0.80, 0.81, 0.80, 0.80, 0.81. These results show that our CGAN scheme can partially collect some information even if the target persons are not related with the published data. However, compared to the results in first scenario, the performances downgrade in

Table 8.1.: User (node) de-anonymization success rate

Prior knowledge	Attack success rate		
	Using original graph	Using original graph & regenerated graph	
		Not cover	Partially cover
40%	5.26%	15.79%	17.89%
50%	7.37%	21.05%	20.00%
60%	9.47%	24.21%	47.37%
70%	10.53%	29.47%	55.79%
80%	17.89%	57.89%	65.26%

both recall and accuracy. The comparison implies that if the attacker knows graph structures closer to the target area, he/she may extract more useful information. After the separation, the target area may be far away from the released graph, both in the sense of physical distance and in the sense of similarity distance. Then the recall and accuracy of de-anonymization become lower.

Someone may argue that we miss the first scenario described in Section 8.2. In the first scenario, there are two cases. First, when an adversary divides the published graph into subgraphs, one subgraph coincidentally exactly matches the target area. Second, no subgraph exactly matches. We argue that the probability of the first case is so negligible that the adversary can hardly have that luck. The second case is usual. However, our experiment shows that the attack performance in this case is similar to the one in Figure 8.9. When existing de-anonymization mainly focuses on this case, our CGAN scheme cannot fully take advantage of overlap between the published graph and the target area. Graph division may break the area in the published graph, which corresponds to the target area, into several parts. However, previous experiments prove that our CGAN scheme is suitable with the case that there is partial overlap or no overlap. Since the attacker cannot guarantee to collect releasing data containing the target area, the proposed scheme can be generally applied.

8.4.4 Case study: user de-anonymization enhancement

As discussed in previous sections, our generating-based de-anonymization works well when the published graph partially covers or does not cover target persons. However, when the attacker is confident that the published graph contains all the target persons, the proposed scheme cannot fully take advantage of this message. This problem, published data containing all target users, was widely studied, and researchers proposed several user de-anonymization methods based on mapping [65, 106, 113, 132]. These mapping-based de-anonymization attacks suffer from several problems, including, the attacker does not have enough information to apply the mapping. And our generating-based de-anonymization scheme is a good supplement to them.

In the first attack scenario described in Section 8.2, the published data contains all target users. Then we can view the big published graph in two parts, with or without target users (there may be some overlap). The first part contains all the users and there may be some other users. Our de-anonymization scheme can hardly cut the subgraph with the exact target users. Hence, we can only add some graphs into our training dataset that partially cover the targets. The second part contains no targets or very few targets. When this part is very useful to our generating-based de-anonymization attack, this part of data is redundant information to a mapping-based de-anonymization attack. Hence, the two de-anonymization attacks, generating-based and mapping based, are orthogonal with each other. If we can combine the two attacks together, we can get better de-anonymization performance. It is notable that the two attacks have different purposes: edge de-anonymization or node de-anonymization. It is simpler to first apply a generating-based attack to complete the graph than to apply a mapping-based attack to de-anonymize users.

In this case study, we implement a basic user de-anonymization scheme proposed in [106], with the help of SALab framework [65]. In this experiment, the attacker holds three graphs: (a) a prior knowledge graph, which is a 100-node graph containing part of the edges, e.g., 40%-80% edges; (b) a published graph, which is also a 100-node graph containing the exact target persons; (c) an auxiliary graph, which is a big published graph that partially covers or does not cover target persons. In the original user de-anonymization attack, the attacker directly maps users between the prior knowledge graph and the published graph, but the auxiliary graph makes no contribution. In the enhanced user de-anonymization attack, we feed the prior knowledge graph (as conditional information) and the auxiliary graph (as training data) into our CGAN to get the regenerated graph. Then we map users between the CGAN regenerated graph and the published graph.

When we successfully map one user in the two graphs, we mark it as a successful de-anonymization attack. In this case study, we sample the published graph into five versions of prior knowledge graphs, with 40%, 50%, 60%, 70%, and 80% edges left. The de-anonymization attack performance is shown in Table 8.1. Similarly, the auxiliary graph can be divided into two scenarios: this graph partially covers the target persons or this graph covers no target person. These results show that when we directly map users, and when the attacker has little prior knowledge, the attack has poor performance. However, after applying the CGAN, we can easily map users from the regenerated graph to the published graph. There is an over three-time success rate improvement of generation-and-mapping attack when comparing with the mapping-only attack. These results prove that our CGAN structure can extract information from the auxiliary graph and add the information into the regenerated graph. Finally, the added structure information improves the attack performance.

Besides the basic de-anonymization scheme, we also try other mapping-based de-anonymization algorithms and find some improvement in attack success rate. However, some of these de-anonymization algorithms are not suitable with the case that the attacker only holds 40%-80% of edge information. Their basic attack performances are poor. For example, the seed and grow de-anonymization scheme needs the core area, the seeds, and their 1-hop neighbors, to have a structure as complete

Table 8.2.: Recall and accuracy of our edge de-anonymization attack, published data is anonymized

Scheme	Accuracy	Recall	Scheme	Accuracy	Recall
HRG, $\epsilon = 1$	61.2%	58.6%	HRG, $\epsilon = 0.1$	67.3%	56.5%
dK-3, $\epsilon = 1$	40.2%	43.0%	dK-3, $\epsilon = 5$	47.0%	43.3%
PHDP, $\epsilon = 1$	46.7%	39.2%	PHDP, $\epsilon = 10$	53.3%	44.5%
Sketching, $\epsilon_a = 1.01$	58.6%	56.8%	Sketch, $\epsilon_a = 0.92$	57.9%	56.4%

as possible [113]. Then the algorithm can successfully map the seeds and grow the mapping on other areas. In the future, we would like to extend our generating-based de-anonymization scheme to match with these application areas.

8.4.5 Case study: anonymized graph de-anonymization

Attackers may suffer two kinds of problems in their de-anonymization attack. The first problem is that the published data partially covers or does not cover target persons, which is evaluated in Section 8.4.3. The second problem is that the published data was anonymized. Ji et al. studied this problem with several kinds of anonymization techniques and de-anonymization techniques [76]. In their study, all the de-anonymization techniques they evaluated have poor performance under specific anonymization techniques.

Although our generating-based de-anonymization attack is not designed for the anonymized graphs, this kind of technique can be adapted to attack the anonymized graphs. In our CGAN model design, we feed the published data as the 'real' data, which is the target of the generated data. When the published data is anonymized, the generating may have the wrong target. The CGAN model still utilizes the information to generate the graph. However, the generated data should be influenced by the correct information and the incorrect information in the published data.

The edge de-anonymization accuracy is shown in Table 8.2. Comparing with the result in Figure 8.9, we find that the misinformation has an essential influence on the de-anonymization attack performance. As discussed before, the anonymized data set the wrong target to the attackers. Then the attacker can hardly use the CGAN model to get the right information. Table 8.2 shows that the attacker may only know about 40%-60% edge information with the anonymized data.

Besides the edge de-anonymization attack, we also evaluate the node de-anonymization attack based on the user de-anonymization scheme proposed in [106]. However, the anonymization schemes with the dK-3 and HRG models eliminate the user identities when they perturb the graph. This elimination does not affect anonymization, but it destroys the ground truth of the user de-anonymization attack. Hence, we only evaluate the PHDP scheme and the sketching scheme. With the PHDP anonymized data, the attacker can successfully de-anonymize 11.49% of users. With the sketching anonymized data, the attacker can successfully de-anonymize 17.24% of users. The results show that the noise in the anonymized data misleads the attack and downgrade the attack performance.

8.5 Conclusion

In this chapter, we propose a new kind of attack, which is based on generating graphs, to de-anonymize OSN data. We design the CGAN model to smartly contain different sources of graph information into the newly generated graph. This generating-based attack overcomes several limitations of existing mapping-based attacks. The evaluation results prove that our new de-anonymization scheme can not only de-anonymize edge information with high accuracy, but also enhance existing user de-anonymization attacks. In the future, we would like to extend the application area of our generating-based de-anonymization scheme, to other user de-anonymization schemes enhancement and to other data distribution tasks.

In the previous two chapters, we study the data de-anonymization problem from the attackers' point of view. In our study, we find that the anonymization, which results in the error of attackers' information, does restrict the de-anonymization performance. Moreover, the attackers also struggle with other problems, e.g., latency and non-overlap. However, the persistent structure extraction methods and the machine learning methods may support the attackers to deal with these problems. In the future researches, one topic should be preventing the attackers get information from these novel methods.

9. SUMMARY AND FUTURE WORK

This dissertation presents a study of the anonymization and de-anonymization of online data sharing, especially the OSN data. In the anonymization defenses, the dissertation aims to strike a better balance between privacy and utility in the published data. The first concern (Chapter 3) is that existing global differential privacy criterion is too strict to preserve graph information. Hence, we use local differential privacy as the new privacy criterion to relax the privacy level. We embed group-based anonymization together with local differential privacy to enhance privacy in the published graph.

However, when analyze our anonymization scheme, we find that some utility loses because of graph abstraction instead of the privacy criterion. Hence, our second angle (Chapter 4) is reducing the utility loss in graph abstraction. We design a comprehensive graph abstraction model which combines three levels of dK model together to extract different levels of information from OSNs. We solve the conflicts between different dK models.

When Chapter 3 and Chapter 4 are both based on existing utility measurement, we raise questions about the graph utility metric. Existing utility metrics are disjoint with each other and each of them can only reveal part of the graph utility. Then, our third design (Chapter 5) is to introduce a comprehensive graph utility measurement called persistent homology. In that chapter, we design a OSN anonymization scheme which can preserve both differential privacy and persistent homology.

Chapter 3, 4, and 5 are all based on differential privacy. When the proposed anonymization schemes have clear bound of privacy, the utility impact to the data is unclear. Therefore, the forth idea (Chapter 6) is about design an anonymization scheme that we can adjust both the privacy level and the utility preservation in the anonymized graph. Our design is based on ADS and we solve some privacy problems including low positive and second round ADS attack.

In the de-anonymization attacks, this dissertation aims to extract more meaningful information from the anonymized data. In dynamic OSNs, existing researches did not fully consider the time evolving information. In existing de-anonymization attacks, seed-mapping is the essential step because it influences the following mapping result. So our first design (Chapter 7) is introduce persistent structures in seed-mapping of dynamic OSNs.

The second concern (Chapter 8) is that de-anonymization attackers may have pool result when the published data is not about the targets. To learn the hidden information in the published data, we introduce the deep learning model called CGAN. We design a generating-based de-anonymization attack which generates new knowledge based on existing background knowledge and published data.

In summary, studying the anonymization and the de-anonymization problems gives us a more comprehensive view from both the attackers and the defenders. The shortages in one side may become the opportunities in the other side. The novel introduced techniques, e.g., persistent structures extraction, machine learning, and sketch, help one group get advantages, but soon benefit the other group, and eventually raise the bars of both groups. When the competition between attackers and defenders never ends, the following works may be essential in the future.

- To the attackers, the utility information emphasized by the anonymization technique, e.g., degree information of dK method, may be opportunity. The smart attackers should consider the specific properties of anonymization schemes when designing their de-anonymization scheme.

- To the defenders, the previous argument does not mean restrictions in preserving all kinds of information. The opportunity is the different goals between the attackers and the benign third-party users. For example, the attackers consider more about the target individuals while the third parties consider more about the overall statistic information. While attackers seek the homogenous information to de-anonymize, the defenders should focus on the heterogeneous information and play their roles.

- To both the attackers and the defenders, novel introduced techniques are important, especially the ones already used in the other side. For example, machine learning has been widely used in data analysis. The defenders need to update their notion about utility and privacy under machine learning. The attackers need to search new attacks with machine learning.

REFERENCES

REFERENCES

[1] Robert J Adler, Omer Bobrowski, Matthew S Borman, Eliran Subag, Shmuel Weinberger, et al. Persistent homology for random fields and complexes. In *Borrowing strength: theory powering applications–a Festschrift for Lawrence D. Brown*, pages 124–143. Institute of Mathematical Statistics, 2010.

[2] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pages 37–48, Rio de Janeiro, Brazil, May 2013. ACM.

[3] Takuya Akiba and Yosuke Yano. Compact and scalable graph neighborhood sketching. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 685–694, San Francisco, CA, Aug 2016. ACM.

[4] Dalal Al-Azizy, David Millard, Iraklis Symeonidis, Kieron O-Hara, and Nigel Shadbolt. A literature survey and classifications on data deanonymisation. In *International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 36–51, Lesvos Island, Greece, Jun 2015. Springer.

[5] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of Android malware in your pocket. In *Proceedings of ISOC Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb 2014.

[6] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *International Conference on Principles of Security and Trust (POST)*, pages 164–186, Uppsala, Sweden, Apr 2017. Springer.

[7] Lars Backstrom, Cynthia Dwork, and Jon 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)*, pages 181–190, Banff, Canada, May 2007. ACM.

[8] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting mobile 3g using wifi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys)*, pages 209–222, New York, NY, Jun 2010. ACM.

[9] Yinfeng Ban, Juhua Pu, Yujun Chen, and Yuanhong Wang. Negan: Network embedding based on generative adversarial networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Rio, Brazil, Jul 2018. IEEE.

[10] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89, 2014.

[11] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems (NIPS)*, pages 585–591, Vancouver, Canada, Dec 2002.

[12] Daniel S Berger, Ramesh K Sitaraman, and Mor Harchol-Balter. Adaptsize: Orchestrating the hot object memory cache in a content delivery network. In *NSDI*, pages 483–498, Boston, MA, Mar 2017.

[13] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Persistent homology for path planning in uncertain environments. *IEEE Transactions on Robotics*, 31(3):578–590, 2015.

[14] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 695–704, Long Beach, CA, Jun 2019.

[15] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 609–618, Stockholm, Sweden, Jul 2018.

[16] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[17] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

[18] Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali. *Content delivery networks*, volume 9. Springer Science & Business Media, 2008.

[19] Zhipeng Cai, Zaobo He, Xin Guan, and Yingshu Li. Collective data-sanitization for preventing sensitive information inference attacks in social networks. *IEEE Transactions on Dependable and Secure Computing*, 15(4):577–590, 2018.

[20] Erik Carlsson, Gunnar Carlsson, and Vin De Silva. An algebraic topological method for feature identification. *International Journal of Computational Geometry & Applications*, 16(04):291–314, 2006.

[21] Gunnar Carlsson, Vin De Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the 25th annual symposium on Computational geometry (SoCG)*, pages 247–256. ACM, 2009.

[22] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology (Crypto)*, pages 199–203, Santa Barbara, CA, Aug 1983. Springer.

[23] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology (CRYPTO)*, pages 319–327, Sydney, Australia, Jan 1990. Springer.

[24] Rui Chen, Benjamin CM Fung, S Yu Philip, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 23(4): 653–676, 2014.

[25] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 199–208, Paris, France, Jun 2009. ACM.

[26] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD)*, pages 459–470, Indianapolis, Indiana, Jun 2010. ACM.

[27] Carla-Fabiana Chiasserini, Michel Garetto, and Emili Leonardi. De-anonymizing clustered social networks by percolation graph matching. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(2):21, 2018.

[28] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Structural inference of hierarchies in networks. In *Statistical network analysis: models, issues, and new directions*, pages 1–13. Springer, 2007.

[29] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.

[30] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.

[31] Edith Cohen. All-distances sketches, revisited: Hip estimators for massive graphs analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27 (9):2320–2334, 2015.

[32] Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches. In *Proceedings of the 26th annual ACM symposium on Principles of distributed computing (PODC)*, pages 225–234, Portland, OR, Aug 2007. ACM.

[33] Edith Cohen, Daniel Delling, Fabian Fuchs, Andrew V Goldberg, Moises Goldszmidt, and Renato F Werneck. Scalable similarity estimation in social networks: Closeness, node labels, and random edge lengths. In *Proceedings of the first ACM conference on Online social networks (COSN)*, pages 131–142, Boston, MA, Oct 2013. ACM.

[34] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, CIKM '14, pages 629–638, Shanghai, China, Nov 2014. ACM.

[35] CoinMarketCap. Cryptocurrency market capitalizations rankings, charts, and more. https://coinmarketcap.com/, 2017. [Online; accessed 30-October-2017].

[36] Jorge Cortés, Geir E Dullerud, Shuo Han, Jerome Le Ny, Sayan Mitra, and George J Pappas. Differential privacy in control and network systems. In *IEEE 55th Conference on Decision and Control (CDC)*, pages 4252–4272, Las Vegas, NV, Dec 2016. IEEE.

[37] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1115–1124, Stockholm, Sweden, Jul 2018.

[38] Wei Dai. b-money. http://www.weidai.com/bmoney.txt, 1998. [Online; accessed 30-April-2018].

[39] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. A sketch-based distance oracle for web-scale graphs. In *Proceedings of the 3rd ACM international conference on Web search and data mining (WSDM)*, pages 401–410, New York, NY, Aug 2010. ACM.

[40] Charo I Del Genio, Hyunju Kim, Zoltán Toroczkai, and Kevin E Bassler. Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PloS one*, 5(4):e10012, 2010.

[41] Xuan Ding, Lan Zhang, Zhiguo Wan, and Ming Gu. De-anonymizing dynamic social networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, Houston, TX, Dec 2011. IEEE.

[42] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 1–19, Xi'an, China, Apr 2008. Springer.

[43] Salah-Eddine Elayoubi, Antonia Maria Masucci, J Roberts, and Berna Sayrac. Optimal d2d content delivery for cellular network offloading. *Mobile Networks and Applications*, 22(6):1033–1044, 2017.

[44] Paul Erdos and Tibor Gallai. Graphs with prescribed degree of vertices (hungarian). *Matematikai Lapok*, 11(1):264–274, 1960.

[45] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[46] Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, and Ainuddin Wahid Abdul Wahab. A review on feature selection in mobile malware detection. *Digital Investigation*, 13(C):22–37, 2015.

[47] Tianchong Gao and Feng Li. Preserving graph utility in anonymized social networks? a study on the persistent homology. In *The 14th International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Orlando, FL, October 2017.

[48] Tianchong Gao and Feng Li. Studying the utility preservation in social network anonymization via persistent homology. *Computers & Security*, 77:49–64, 2018.

[49] Tianchong Gao and Feng Li. Sharing social networks using a novel differentially private graph model. In *16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4, Las Vegas, NV, Jan 2019. IEEE.

[50] Tianchong Gao and Feng Li. PHDP: preserving persistent homology in differentially private graph publications. In *IEEE International Conference on Computer Communications (INFOCOM)*, Paris, France, April 2019.

[51] Tianchong Gao and Feng Li. De-anonymization of dynamic online social networks via persistent structures. In *2019 IEEE International Conference on Communications (ICC)*, Shanghai, P.R. China, May 2019.

[52] Tianchong Gao and Feng Li. Efficient content delivery via interest queueing. In *IEEE International Conference on Communications (ICC): Mobile and Wireless Networks Symposium (ICC)*, Shanghai, P.R. China, May 2019.

[53] Tianchong Gao and Feng Li. Privacy-Preserving sketching for online social network data publication. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Boston, USA, June 2019.

[54] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. Preserving local differential privacy in online social networks. In *International Conference on Wireless Algorithms, Systems, and Applications (WASA)*, pages 393–405, Guilin, China, Jun 2017. Springer.

[55] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. Local differential privately anonymizing online social networks under hrg-based model. *IEEE Transactions on Computational Social Systems*, 5(4):1009–1020, 2018.

[56] Tianchong Gao, Wei Peng, Devkishen Sisodia, Tanay Kumar Saha, Feng Li, and Mohammad Al Hasan. Android malware detection via graphlet sampling. *IEEE Transactions on Mobile Computing*, 1(1):1–15, 2018.

[57] Hugo Gascon, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. Structural detection of Android malware using embedded call graphs. In *Proceedings of ACM Workshop on Artificial Intelligence and Security (AISec)*, Berlin, Germany, Nov 2013.

[58] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008.

[59] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice.* Chapman and Hall/CRC, 1995.

[60] Minas Gjoka, Bálint Tillman, and Athina Markopoulou. Construction of simple graphs with a target joint degree matrix and beyond. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1553–1561, Hong Kong, China, Apr 2015. IEEE.

[61] Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)*, 21(1):3:1–3:30, 2018.

[62] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, Montreal, Canada, Dec 2014.

[63] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151(1):78–94, 2018.

[64] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 855–864, San Francisco, CA, Aug 2016. ACM.

[65] Gábor György Gulyás, Benedek Simon, and Sándor Imre. An efficient and robust social network de-anonymization attack. In *Proceedings of the 2016 ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 1–11, Vienna, Austria, Oct 2016. ACM.

[66] Bo Han, Pan Hui, and Aravind Srinivasan. Mobile data offloading in metropolitan area networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 14(4):28–30, 2011.

[67] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, Barcelona, Spain, Jul 2018. De Gruyter.

[68] Jordi Herrera-Joancomartí. Research and challenges on bitcoin anonymity. In *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, pages 3–16. Springer, 2015.

[69] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 603–618, Dallas, TX, Oct 2017. ACM.

[70] Danijela Horak, Slobodan Maletić, and Milan Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034, 2009.

[71] Weiyu Huang and Alejandro Ribeiro. Persistent homology lower bounds on network distances. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4845–4849, Shanghai, China, Mar 2016. IEEE.

[72] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1125–1134, Honolulu, HI, Jul 2017.

[73] Sergei Ivanov and Panagiotis Karras. Harvester: Influence optimization in symmetric interaction networks. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 61–70, Montreal, Canada, Oct 2016. IEEE.

[74] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, and Raheem Beyah. Structural data de-anonymization: Quantification, practice, and implications. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1040–1053, Scottsdale, AZ, Nov 2014. ACM.

[75] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem A Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *Network and Distributed System Security (NDSS)*, San Diego, CA, Feb 2015.

[76] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem A Beyah. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *24th USENIX Security Symposium*, pages 303–318, Washington, DC, Aug 2015.

[77] Shouling Ji, Weiqing Li, Shukun Yang, Prateek Mittal, and Raheem Beyah. On the relative de-anonymizability of graph data: Quantification and evaluation. In *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, Apr 2016. IEEE.

[78] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[79] Peter Kairouz. Local differential privacy. http://web.stanford.edu/kairouzp/privacy_google.pdf, 2017. [Online; accessed 29-October-2018].

[80] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems (NIPS)*, pages 2879–2887, Montreal, Canada, Dec 2014.

[81] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[82] Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 439–453, Dallas, TX, Nov 2017. ACM.

[83] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv e-prints*, page arXiv:1611.07308, 2016.

[84] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, Toulon, France, Apr 2017.

[85] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[86] Jérôme Kunegis. KONECT – The Koblenz Network Collection. In *Proceedings of the international conference on world wide web (WWW)*, pages 1343–1350, Rio de Janeiro, Brazil, May 2013.

[87] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th international conference on World wide web (WWW)*, pages 591–600, Raleigh, NC, Apr 2010. ACM.

[88] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection, June 2014.

[89] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. [Online; accessed 12-January-2019].

[90] Huaxin Li, Qingrong Chen, Haojin Zhu, Di Ma, Hong Wen, and Xuemin Sherman Shen. Privacy leakage via de-anonymization and aggregation in heterogeneous social networks. *IEEE Transactions on Dependable and Secure Computing*, 1(1):1–1, 2017.

[91] Xinghua Li, Meixia Miao, Hai Liu, Jianfeng Ma, and Kuan-Ching Li. An incentive mechanism for k-anonymity in lbs privacy protection based on credit mechanism. *Soft Computing*, 21(14):3907–3917, 2017.

[92] Yujia N Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[93] Qin Liu, Guojun Wang, Feng Li, Shuhui Yang, and Jie Wu. Preserving privacy with probabilistic indistinguishability in weighted social networks. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1417–1429, 2017.

[94] Jun Long, Lei Zhu, Zhan Yang, Chengyuan Zhang, and Xinpan Yuan. Temporal activity path based character correction in heterogeneous social networks via multimedia sources. *Advances in Multimedia*, 2018(2058670):1–16, 2018.

[95] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. Systematic topology analysis and generation using degree correlations. *ACM SIGCOMM Computer Communication Review*, 36(4):135–146, 2006.

[96] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. Cryptology ePrint Archive, Report 2016/1054, 2016. URL https://eprint.iacr.org/2016/1054.pdf. Accessed: 2017-06-29.

[97] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 455–471, Dallas, TX, Oct 2017. ACM.

[98] Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra. Classifying android malware through subgraph mining. In Joaquin Garcia-Alfaro, Georgios Lioudakis, Nora Cuppens-Boulahia, Simon Foley, and William M. Fitzgerald, editors, *Data Privacy Management and Autonomous Spontaneous Security*, pages 268–283, Berlin, Heidelberg, 2014. Springer.

[99] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, Providence, RI, Oct 2007. IEEE.

[100] Bo Mei. *Improving Social Network Inference Attacks via Deep Neural Networks*. PhD thesis, The George Washington University, 2018.

[101] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv e-prints*, page arXiv:1411.1784, Nov 2014.

[102] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.

[103] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations (ICLR)*, pages 1–11, San Juan, Puerto Rico, May 2016.

[104] Shahid Mumtaz and Jonathan Rodriguez. *Smart device to smart device communication*. Springer, 2014.

[105] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. [Online; accessed 18-June-2019].

[106] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *30th IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, CA, May 2009. IEEE.

[107] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 2014–2023, New York, NY, Jun 2016.

[108] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[109] Lucky Onwuzurike, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version). *ACM Transactions on Privacy and Security (TOPS)*, 22(2):14, 2019.

[110] Pedram Pedarsani, Daniel R Figueiredo, and Matthias Grossglauser. A bayesian method for matching two similar graphs without seeds. In *51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1598–1607, Champaign, IL, Oct 2013. IEEE.

[111] Wei Peng, Feng Li, Xukai Zou, and Jie Wu. Seed and grow: An attack against anonymized social networks. In *9th Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 587–595, Seoul, Korea, Jun 2012. IEEE.

[112] Wei Peng, Feng Li, Xukai Zou, and Jie Wu. The virtue of patience: Offloading topical cellular content through opportunistic links. In *2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, pages 402–410, Hangzhou, China, 2013 2013. IEEE.

[113] Wei Peng, Feng Li, Xukai Zou, and Jie Wu. A two-stage deanonymization attack against anonymized social networks. *IEEE Transactions on Computers*, 63(2):290–303, 2014.

[114] Wei Peng, Tianchong Gao, Devkishen Sisodia, Tanay Kumar Saha, Feng Li, and Mohammad Al Hasan. Acts: Extracting android app topological signature through graphlet sampling. In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 37–45, Philadelphia, PA, Oct 2016. IEEE.

[115] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 701–710, New York, NY, Aug 2014. ACM.

[116] Giovanni Petri, Martina Scolamiero, Irene Donato, and Francesco Vaccarino. Topological strata of weighted complex networks. *PloS one*, 8(6):e66506, 2013.

[117] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *draft version 0.5*, 9:14, 2016.

[118] N Pržulj, Derek G Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[119] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, and Linlin Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9, San Francisco, CA, Apr 2016. IEEE.

[120] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, Linlin Chen, Taeho Jung, and Junze Han. Social network de-anonymization and privacy inference with knowledge graph model. *IEEE Transactions on Dependable and Secure Computing*, 16(4):679–692, 2017.

[121] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 425–438, Dallas, TX, Oct 2017. ACM.

[122] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1511.06434, 2015.

[123] Mahmudur Rahman, Mansurul Bhuiyan, and Mohammad Al Hasan. GRAFT: An approximate graphlet counting algorithm for large graph analysis. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, Maui, HI, Oct 2012.

[124] Mahmudur Rahman, Mansurul Alam Bhuiyan, Mahmuda Rahman, and Mohammad Al Hasan. GUISE: a uniform sampler for constructing frequency histogram of graphlets. *Knowledge and information systems*, 38(3):511–536, 2014.

[125] Vaibhav Rastogi, Yan Chen, and Xuxian Jiang. Droidchameleon: evaluating android anti-malware against transformation attacks. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (Asia CCS)*, pages 329–334, Hangzhou, China, May 2013. ACM.

[126] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. *Security and privacy in social networks*, 2:197–223, 2013.

[127] Sara Retal, Miloud Bagaa, Tarik Taleb, and Hannu Flinck. Content delivery network slicing: Qoe and cost awareness. In *IEEE International Conference on Communications (ICC)*, pages 1–6, Paris, France, May 2017. IEEE.

[128] Huan Rong, Tinghuai Ma, Meili Tang, and Jie Cao. A novel subgraph k$^+$-isomorphism method in social network based on graph similarity detection. *Soft Computing*, 22(8):2583–2601, 2018.

[129] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the ACM SIGCOMM conference on Internet measurement conference (IMC)*, pages 81–98, Berlin, Germany, Nov 2011. ACM.

[130] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems (NIPS)*, pages 2234–2242, Barcelona, Spain, Dec 2016.

[131] Vikas Kumar Sihag. A clustering approach for structural k-anonymity in social networks using genetic algorithm. In *Proceedings of the CUBE International Information Technology Conference*, CUBE '12, pages 701–706, New York, NY, 2012. ACM.

[132] Benedek Simon, Gábor György Gulyás, and Sándor Imre. Analysis of grasshopper, a novel social network de-anonymization algorithm. *Periodica Polytechnica Electrical Engineering and Computer Science*, 58(4):161–173, 2014.

[133] Hassan Sinky and Bechir Hamdaoui. Cloudlet-aware mobile content delivery in wireless urban communication networks. In *Global Communications Conference (GLOBECOM)*, pages 1–7, Austin, TX, Dec 2016. IEEE.

[134] Alberto Speranzon and Shaunak D Bopardikar. An algebraic topological perspective to privacy. In *American Control Conference (ACC)*, pages 2086–2091, Boston, MA, Jul 2016. IEEE.

[135] Rade Stanojevic, Mohamed Nabeel, and Ting Yu. Distributed cardinality estimation of set operations with differential privacy. In *IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 37–48, Washington, DC, Aug 2017. IEEE.

[136] Ariel Stolerman, Rebekah Overdorf, Sadia Afroz, and Rachel Greenstadt. Breaking the closed-world assumption in stylometric authorship attribution. In *International Conference on Digital Forensics (IFIP)*, pages 185–205, Vienna, Austria, Jan 2014. Springer.

[137] Matthias Studer and Gilbert Ritschard. What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 179(2):481–511, 2016.

[138] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[139] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. *arXiv e-prints*, page arXiv:1709.02753, 2017.

[140] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, Apr 2018.

[141] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks (WOSN)*, pages 37–42, Barcelona, Spain, Aug 2009.

[142] Guojun Wang, Qin Liu, Feng Li, Shuhui Yang, and Jie Wu. Outsourcing privacy-preserving social networks to a cloud. In *The 32nd Annual IEEE International Conference on Computer Communications (INFOCOM)*, pages 2886–2894, Turin, Italy, Apr 2013. IEEE.

[143] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the 32rd Conference on Artificial Intelligence (AAAI)*, pages 2508–2515, New Orleans, LA, Feb 2018.

[144] Jinbao Wang, Zhipeng Cai, Yingshu Li, Donghua Yang, Ji Li, and Hong Gao. Protecting query privacy with differentially private k-anonymity in location-based services. *Personal and Ubiquitous Computing*, 22(3):453–469, 2018.

[145] Meiqi Wang, Qingfeng Tan, Xuebin Wang, and Jinqiao Shi. De-anonymizing social networks user via profile similarity. In *IEEE 3rd International Conference on Data Science in Cyberspace (DSC)*, Guang zhou, China, Jun 2018. IEEE.

[146] Rui Wang, Xi Peng, Jun Zhang, and Khaled B Letaief. Mobility-aware caching for content-centric wireless networks: modeling and methodology. *IEEE Communications Magazine*, 54(8):77–83, 2016.

[147] Yazhe Wang and Baihua Zheng. Preserving privacy in social networks against connection fingerprint attacks. In *IEEE 31st International Conference on Data Engineering (ICDE)*, pages 54–65, Seoul, South Korea, Apr 2015. IEEE.

[148] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy*, 6(2):127, 2013.

[149] Wikipedia. Facebook–cambridge analytica data scandal — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal, 2018. [Online; accessed 21-July-2018].

[150] Wikipedia. List of android app stores. https://en.wikipedia.org/wiki/List_of_Android_app_stores, 2019. [Online; accessed 22-May-2019].

[151] David F Williamson, Robert A Parker, and Juliette S Kendrick. The box plot: a simple visual method to interpret data. *Annals of internal medicine*, 110(11): 916–921, 1989.

[152] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151(1):1–32, 2014.

[153] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 911–920, New York, NY, Aug 2014. ACM.

[154] Chen Xu, Caixia Gao, Zhenyu Zhou, Zheng Chang, and Yunjian Jia. Social network-based content delivery in device-to-device underlay cellular networks using matching theory. *IEEE Access*, 5:924–937, 2017.

[155] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, May 2019.

[156] Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. Modeling and discovering vulnerabilities with code property graphs. In *Proceedings of IEEE Symposium on Security and Privacy*, San Jose, CA, May 2014.

[157] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 332–348, Toronto, Canada, Oct 2018. ACM.

[158] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision (ECCV)*, pages 517–532, Amsterdam, Netherlands, Oct 2016. Springer.

[159] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*, pages 5694–5703, Stockholm, Sweden, Jul 2018.

[160] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5907–5915, Venice, Italy, Oct 2017.

[161] Mu Zhang, Yue Duan, Heng Yin, and Zhiruo Zhao. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1105–1116, Scottsdale, AZ, Nov 2014. ACM.

[162] Zijian Zhang, Zhan Qin, Liehuang Zhu, Jian Weng, and Kui Ren. Cost-friendly differential privacy for smart meters: Exploiting the dual roles of the noise. *IEEE Transactions on Smart Grid*, 8(2):619–626, 2017.

[163] Min Zheng, Mingshen Sun, and John Lui. Droid Analytics: A signature based analytic system to collect, extract, analyze and associate Android malware. In *Proceedings of IEEE Trust, Security and Privacy in Computing and Communications (TrustCom)*, Melbourne, Australia, Jul 2013.

[164] Ziyun Zhu and Tudor Dumitras. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 767–778, Vienna, Austria, Oct 2016. ACM.

[165] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

[166] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.

[167] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, pages 1–15, New Orleans, LA, May 2019.

[168] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2847–2856, London, United Kingdom, Aug 2018.

VITA

VITA

Tianchong Gao received his B.S. in Information Science and Technology from Southeast University, China, in 2014. He received his M.S. in Electrical and Computer Engineering from University of Michigan-Dearborn, USA, in 2015. He joined Purdue University in August, 2015 to pursue his Ph.D. degree in computer engineering under the supervision of Dr. Feng Li.