UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA - VARESE

DiSTA

Dipartimento di Scienze Teoriche e Applicate

P H D T H E S I S

to obtain the title of

Doctor of Science

Specialty : Computer Science

Defended by

Anh-Tu Hoang

PRIVACY-PRESERVING PUBLISHING OF KNOWLEDGE GRAPHS

Advisor: Prof. Barbara Carminati

Advisor: Prof. Elena Ferrari

defended on December 03, 2020

To my wonderful mother and father.

Thank you for raising me to be who I am today. Without your love and encouragement, I will not have enough confidence and freedom to do all the things I want in my life.

Acknowledgments

First of all, I would like to express my deep and sincere gratitude to my advisers, Professor Elena Ferrari and Professor Barbara Carminati, for their invaluable guidance and support. Their immense knowledge, vision, patience, and sincerity greatly enrich my research skills and motivate me to finish this study. It is a great honour for me to work and study under their guidance.

Furthermore, I would like to thank the reviewers of my thesis, i.e., Professor Dan Lin and Professor Xukai Zou. Their detail comments and hard questions enrich my knowledge and widen my research from various perspectives.

Last but not least, I would like to give my thanks to all of my laboratory members: Pietro Colombo, Gökhan Sağırlar, Bikash Chandra Singh, Zulfikar Alom, Christian Rondanini, Engin Deniz Tümer, Federico Daidone, Ha Xuan Son, Ahmed Lekssays, Mauro Santabarbara, and Roberta Viola. They helped me to get used to life in Italy and supported my research throughout these years. We have passed great moments together. Without them, my PhD life would not be so enjoyable and memorable.

Abstract

Online social networks (OSNs) attract a huge number of users sharing their data every day. These data can be shared with third parties for various usage purposes, such as data analytics and machine learning [26]. Unfortunately, adversaries can exploit shared data to infer users' sensitive information [49]. Various anonymization solutions [26] have been presented to anonymize shared data such that it is harder for adversaries to infer users' personal information. Whereas OSNs contain both users' attributes and relationships, previous work only consider anonymizing either attributes, illustrated in relational data [33, 36, 49, 53] or relationships, represented in directed graphs [10, 60].

To cope with this issue, in this thesis, we consider the research challenge of anonymizing knowledge graphs (KGs), due to their flexibility in representing both attributes' values and relationships of users. The anonymization of KGs is not trivial since adversaries can exploit both attributes and relationships of their victims [45–47]. In the era of big data, these solutions are significant as they allow data providers to share attributes' values and relationships together. Over the last three years, we have done important research efforts which has resulted in the definition of different anonymization solutions for KGs for many relevant scenarios, i.e., anonymization of static KGs, sequential anonymization of KGs, and personalized anonymization of KGs.

Since KGs are directed graphs, we started our research by investigating anonymization solutions for directed graphs [10, 60]. As anonymization algorithms proposed in the literature (i.e., [10, 60]) cannot always anonymize graphs, we first presented the Cluster-Based Directed Graph Anonymization Algorithm (CDGA). We proved that CDGA can always generate anonymized directed graphs. We analyzed an attacking scenario where an adversary can exploit attributes' values and relationships of his/her victims to re-identify these victims in anonymized KGs. To protect users in this scenario, we presented the k-Attribute Degree (k-ad) protection model to ensure that users cannot be re-identified with a confidence higher than $\frac{1}{k}$. We proposed the Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA) to anonymize KGs for this scenario. CKGA has been designed for a scenario where KGs are statically anonymized. Unfortunately, the adversary can still re-identify his/her victims if he/she has access to many versions of the anonymized KG. To cope with this issue, we further presented the k^w-Time-Varying Attribute Degree to give users the same protection of k-ad even if the adversary gains access to w continuous anonymized KGs. In addition, we proposed the Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm to anonymize KGs while allowing data providers

to insert/re-insert/remove/update nodes and edges of their KGs. However, users are not allowed to specify their privacy preferences which are crucial to for those users requiring strong privacy protection, such as influencers [54,59]. To this end, we proposed the Personalized k-Attribute Degree to allow users to specify their own value of k. The effectiveness of the proposed algorithms has been tested with experiments on real-life datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the popularity of online social networks (OSNs), many people share their information through these networks. For instance, 30% of internet users use Facebook more than once a day, generating 4 petabytes of data per day.[1] These data contain not only users' attributes (e.g., *education*, *birthday*), but also their relationships (e.g., *follows*). Even though the data contain sensitive information, OSNs' providers (e.g., *Twitter*, *Facebook*) sell or share them with other companies or researchers, and this may result in serious privacy breaches (e.g., *Cambridge Analytica*, *Gnip*)[2].

A typical data sharing scenario starts with a data provider publishing the data of the users it manages to a data recipient. There are two paradigms for how the data is released: interactive and non-interactive. In interactive setting, the provider receives queries from the recipient and returns users' data satisfying these queries. According to this paradigm, the provider can monitor what type of data the recipient requests and modify the returned data such as to protect users' sensitive information. Unfortunately, this paradigm requires huge effort from the providers, and therefore most of them do not support such option. For instance, most popular data sharing repositories (i.e., UCI Machine Learning Repository[3], Stanford SNAP[4], and Kaggle[5]) only allow users to download their datasets directly from their websites. Twitter also provides APIs to allow data recipients to download its users' public data. In these APIs, Twitter does not modify the data to protect its users' privacy but let the users decide which data are public. Therefore, non-interactive settings in which the provider modifies data to hide users' sensitive information and sends the modified data to the recipient is more popular.

The most straightforward way to hide users' sensitive information is to remove users' identities from the published data. However, in 2002, Sweeney et al. [49] proved that adversaries can still infer users' sensitive information from them. She showed that 87% of

[1] https://www.brandwatch.com/blog/facebook-statistics/

[2] https://www.forbes.com/sites/kalevleetaru/2018/12/15/what-does-it-mean-for-social-media-platforms-to-sell-our-data/

[3] http://archive.ics.uci.edu/ml/index.php

[4] http://snap.stanford.edu/data/index.html

[5] https://www.kaggle.com/datasets

the population in the United States has unique values for attributes: *ZIP*, *gender*, and *date of birth* and these attributes can be associated with healthcare data to infer users' diseases even though the data do not contain users' identities. In 2008, Narayanan et al. [41] re-identified users in the Netflix Prize dataset even though Netflix removed identities of their users in the dataset. For this reason, data providers must find a better way to anonymize their data such that users' sensitive information is undiscoverable.

Sweeney et al. [49] presented the first protection model, namely k-anonymity, to prevent users in an anonymized relational dataset from being re-identified with a confidence higher than $\frac{1}{k}$, where k is a positive integer provided by data providers. By increasing or decreasing values of k, the providers can increase or decrease the protection level of their users' privacy. Since [49], different extensions of k-anonymity, such as l-diversity [36] and t-closeness [33] have been proposed to protect users' sensitive information from different attacks and background knowledge. Unfortunately, these work only consider users' attributes.

Since the target of this thesis is protecting data shared on social networks, we need to focus on anonymization models also considering users' relationships. In general, there are two types of user-to-user relationships: undirected (e.g., *friendship*) and directed (e.g., *follow*) relationships. These relationships can be represented in undirected and directed graphs, respectively.

Many k-anonymity extensions [12, 34, 61, 64] have been presented to anonymize undirected graphs. However, due to the popularity of directed relationships in OSNs, they are not efficient enough to protect users in OSNs. Recently, the Paired k-degree [10] and the K-In&Out-Degree Anonymity [60] have been proposed to anonymize directed graphs. Similar to k-anonymity [49], both of them ensure that users in the anonymized directed graphs cannot be re-identified with a confidence higher than $\frac{1}{k}$, where k is a positive number provided by data providers. Unfortunately, adversaries can still infer users' sensitive information if they have access to many versions of the anonymized data [7, 39].

Privacy preserving sequential publishing has been widely addressed in the context of relational data (e.g., [3, 7, 55, 63]). Anonymization solutions for sequentially publishing of undirected [37, 39, 50] and directed [60] graphs have also been presented. However, since the work do not anonymize attributes' values and relationships together, a sequential anonymization solution for KGs is still missing.

Another issue of the previous work is that they have ignored a significant fact, that is, different users may have different privacy protection requirements [59]. Many anonymization solutions for relational data [35, 54, 56] have been extended to allow users to specify their own level of privacy protection. Similarly, various personalized anonymization solutions for undirected graphs have been presented in [27, 59]. Nevertheless, there are no anonymization solutions for directed graphs and KGs.

The key limitation of the previous anonymization work [3, 7, 10, 12, 33–37, 39, 49, 50, 54–56, 59–61, 63, 64] is related to the data model they support. Neither relational data nor graphs can illustrate both attributes' values and relationships together. As a result, the work cannot anonymize both users' attributes and their relationships.

Recently, after the announcement of Google Knowledge Graph in 2012, knowledge

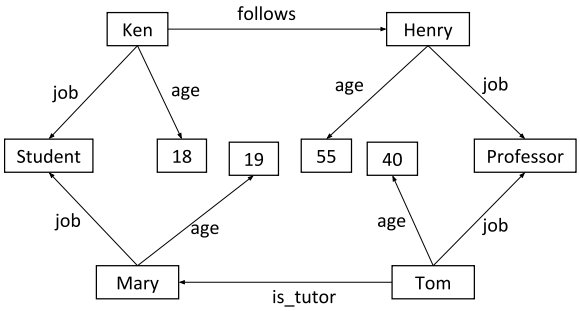


Figure 1.1: An example of knowledge graph.

graphs (KGs) are taking a huge consideration from many big companies, such as Wikipedia, Amazon, and Microsoft. The huge popularity of KGs comes from their flexibility in modelling users' data. A KG is a directed graph which models two types of users' information: attributes' values and relationships. Figure. 1.1 illustrates an example of using a KG to model attributes (e.g., *age* and *job*) and relationships (e.g., $follows$) of four users (e.g., Ken, $Henry$, $Mary$, and Tom). By using KGs, data providers can represent and share both types of their users' information together. However, as KGs contain much more information compared to relational data and traditional graphs, users' sensitive information in anonymized KGs is more vulnerable to attacks [45–47]. Unfortunately, previous work [10, 12, 33, 34, 36, 49, 60, 61, 64] cannot be used to anonymize KGs as they do not consider both users' attributes and their relationships at the same time. Consequently, the providers still need new anonymization approaches for KGs that protect their users by considering both their attributes and relationships and this is the overall goal of the research work described in this thesis.

This thesis presents our efforts on preserving users' privacy in anonymized KGs. At this purpose, this thesis considers three scenarios: static publishing of anonymized KGs, sequentially publishing of anonymized KGs, and personalized anonymization of KGs. The first scenario allows data providers to publish their anonymized KGs once. The second one extends the first to allow the providers to insert/re-insert/update/delete their data and publish new anonymized versions of their KGs. The final one let users specify their own privacy protection levels and anonymize KGs such that all users are protected under their own levels.

1.1 Contributions

This thesis aims at presenting anonymization solutions for KGs. Figure. 1.2 illustrates the associations between the main research contributions of this thesis, which are summarized in what follows.

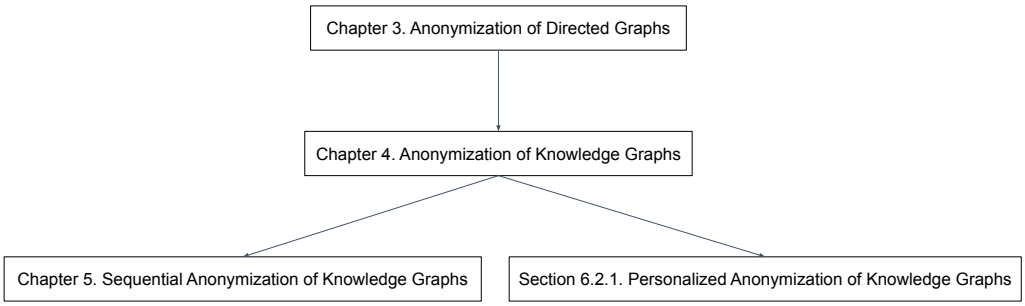


Figure 1.2: Graphical representation of key contributions of this thesis.

1.1.1 Anonymization of Directed Graphs

Since KGs are directed graphs, we first focused on previously defined protection models tailored for these graphs, namely the Paired k-degree [10] and the K-In&Out-Degree Anonymity [60]. While these models are sound, the proposed anonymization algorithms cannot always generate anonymized graphs (see Chapter 3 for more details). Thus, in Chapter 3, as a first step, we focused on proposing a more reliable anonymization algorithm, i.e., the Cluster-Based Directed Graph Anonymization Algorithm (CDGA), to generate anonymized directed graphs satisfying the requirements of both the Paired k-degree and the K-In&Out-Degree Anonymity. The intuition of the algorithm is to gather users into clusters that have at least k users and add/remove users' relationships to make the out-/in-degree of users in the same cluster identical. To prevent our algorithm from modifying too many relationships, we defined the Degree Information Loss Metric (DM) to measure the amount of information users lose when their out-/in-degree are identical. Then, our algorithm inserts users into clusters such that the loss of their information calculated by DM is minimized. To show the effectiveness of our algorithm, we tested it on three real-life datasets. The results of this work have been published in [21] and will be presented in Chapter 3.

1.1.2 Anonymization of Knowledge Graphs

After dealing with directed graphs, we shift our research to anonymize KGs containing both types of users' information: attributes' values and relationships. Adversaries can re-identify users in anonymized KGs by combining both information types of their victims [45–47]. Unfortunately, previously mentioned protection models, i.e., the Paired k-degree [10], and the K-In&Out-Degree Anonymity [60], cannot be applied as they only anonymize one relationship type of these users.

To cope with this issue, we proposed k-Attribute Degree (k-ad), an extension of k-anonymity [49], the Paired k-degree [10], and the K-In&Out-Degree Anonymity [60], to protect users' identities in anonymized KGs. k-ad prevents users in anonymized KGs from being re-identified with a confidence higher than $\frac{1}{k}$ even if adversaries exploit both attributes' values and relationships of these users, where k is a positive integer provided by

data providers. The higher k is, the harder is for adversaries to re-identify users. However, it also makes the anonymized KG lose its information.

To measure the loss of users' information in anonymized KGs, we presented two information loss metrics. The first one is the Attribute Information Loss Metric (AM) that measures the amount of information users lose when we anonymize their attributes' values. However, AM does not consider the truthfulness of these values. Indeed, by adding fake user-to-attribute edges for data anonymization, user attributes' values can be untruthful. As an example, when anonymizing attributes' values of a user whose *age* is *18*, we can accidentally add a fake edge which models his/her *job* as $Professor$. Since the user whose *age* is *18* is not likely to be a $Professor$, an adversary can infer that either *age* or *job* of the user is fake and try to remove them to infer his/her real attributes' values. To address this issue, we present the second metric, namely the Attribute Truthfulness Information Loss Metric (ATM), to measure how truthful attributes' values of a user are. ATM uses an indicator to decide whether two attributes' values are truthful when a user has these values at the same time. We trained the indicator by using a dataset of truthful and untruthful attributes' values. To generate the dataset, we follow the Closed World Assumption [29] which states that two attributes' values are untruthful if the original KG does not contain any user that have these values at the same time. If data providers do not have access to the indicator, they can use AM to measure users' information loss. In addition, we combine AM and ATM with DM to measure the loss of not only attributes' values but also relationships of users in anonymized KGs.

Moreover, we designed the Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA) to anonymize KGs, according to the proposed k-Attribute Degree (k-ad) protection model. To allow for more flexibility, we aimed at allowing data providers to specify which clustering algorithm they want to use to generate clusters. To reach this goal, our algorithm performs an additional step that generates data points for users in the given KG such that the Euclidean distance of two points is almost equal to the information loss of making the attributes' values and out-/in-degrees of users corresponding to these points identical. Since most state-of-the-art clustering algorithms (e.g., k-means [14], HDBSCAN [9]) can take as input these points [14], our algorithm allows data providers to use most clustering algorithms to generate clusters. To prevent the generated clusters from having less than k users, we proposed the k-Means Partition Algorithm (KP) to ensure that all of the generated clusters have from k to $2 \times k - 1$ users. The carried out experiments on five real-life datasets showed that CKGA outperforms previous algorithms (i.e., [10, 21, 60]). We have published results of this work in [22] and they will be described in Chapter 4.

1.1.3 Sequential Anonymization of Knowledge Graphs

Although k-ad prevents adversaries from re-identifying users in an anonymized KG, they can still re-identify these users if they exploit many versions of the anonymized KG. Therefore, we developed the k^w-Time-Varying Attribute Degree (k^w-tad) protection model, to protect identities of users appearing at least once in w continuous anonymized KGs, where

w and k are two positive numbers provided by data providers. The providers can use w to control how many continuous anonymized KGs to monitor and k to control the confidence of re-identifying their users in these KGs. Moreover, we proposed the Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm (CTKGA) to generate anonymized KGs satisfying requirements of k^w-tad even if the providers insert, re-insert, update, and delete the users in their KGs. Different from CKGA, CTKGA removes the need of generating data points for users in anonymized KGs. More precisely, given an information loss metric measuring the amount of information two users lose when making their attributes' values and out-/in-degrees identical (i.e., AM, DM, ATM), CTKGA generates the distance matrix whose distance between two users are calculated using the information loss metric. Then, CTKGA can use any clustering algorithm that supports distance matrix to generate the clusters. To prove the efficiency of our algorithm, we compared it to previous work (i.e., [10, 21, 22]) by running experiments on six real-life datasets (see Appendix B for these datasets' description). The results of this work have been accepted to be published in [23] and will be presented in Chapter 5.

1.1.4 Personalized Anonymization of Knowledge Graphs

While each user may have different concerns on the privacy protection level [54, 59], k-ad applies the same protection level for all users in the anonymized KGs. Therefore, in this ongoing work, we plan to present the Personalized k-Attribute Degree (p-k-ad), an extension of k-ad, to protect users with their own values of k. However, anonymizing users whose k value is too high with those whose k value is too low can result in high information loss. The following example demonstrates this scenario.

Example 1. Let Ken and $Laura$ be two users in the original KG. Suppose Ken's k is 20, thus he requires to have at least 19 other users have the same attributes' values and out-/in-degrees with him. Suppose that k value of $Laura$ is only 2. She thus only requires another user to have the same attributes' values and out-/in-degrees with her. If we make their attributes and out-/in-degrees identical, $Laura$ will lose a huge amount of information. The reason is that instead of anonymizing her attributes and relationships with those of another user, we must anonymize hers with those of 19 other users.

To remedy this problem, we are working on a cluster-based anonymization algorithm which considers different k values of users. We will present the intuitive idea of the algorithm in Section 6.2.1.

1.2 Thesis Organization

This thesis contains seven chapters and three appendices whose content is briefly described in what follows:

- **Chapter 2**. We review advantages and disadvantages of state-of-the-art anonymization approaches for relational data and undirected/directed graphs in this chapter. Moreover, we review recent attacking models to infer users' sensitive information in

KGs. Related work related to sequential and personalized anonymization of data are also presented in this chapter.

- **Chapter 3**. In this chapter, we first show the limitation of previous work [10, 60] on anonymizing directed graphs. Then, we present the Cluster-Based Directed Graph Anonymization Algorithm (CDGA) to ensure that users' relationships modelled in directed graphs can always be anonymized.

- **Chapter 4**. This chapter describes k-Attribute Degree (k-ad), a protection model which protects users' identities in anonymized KGs even if adversaries take advantages of not only attributes' values but also relationships of these users. In addition, we present the Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA), an extension of CDGA described in Chapter 3, to anonymize both attributes' values and relationships of users such that the anonymized KGs satisfy k-ad.

- **Chapter 5**. In this chapter, we demonstrate k^w-Time-Varying Attribute Degree (k^w-tad), an extension of k-ad, that we have developed to ensure that adversaries cannot re-identify users even if they exploit w continuous versions of the anonymized KG. Furthermore, we present the Cluster-Based Time-Varying Knowledge Graph Anonymization Algorithm (CTKGA) which generates anonymized KGs satisfying k^w-tad even though data providers insert/re-insert/update/delete users of their KGs.

- **Chapter 6**. This chapter summarizes the results of the research described in this thesis and presents our future work (i.e., the personalized anonymization solution and other research directions).

- **Appendix A**. In this appendix, we describe abbreviations and notations that we used in this thesis.

- **Appendix B**. This appendix contains the description of real-life datasets that we used in all of our experiments.

- **Appendix C**. We summarize our publications and their abstract in this appendix.

1.3 Related Publications

Results of research activities described in this thesis brought to the following publications (see Appendix C for more details about these publications):

- Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Clusters-Based Anonymization of Directed Graphs". Proceedings of the IEEE International Conference on Collaboration and Internet Computing, 2019.

- Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Clusters-Based Anonymization of Knowledge Graphs". Proceedings of the International Conference on Applied Cryptography and Network Security, 2020.

- Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Privacy-Preserving Sequentially Publishing of Knowledge Graphs". Proceedings of the IEEE International Conference on Data Engineering, accepted, 2021.
- Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Personalized Anonymization of Knowledge Graphs". Under preparation.

Chapter 2

Related Work

2.1 Introduction

As described in the introduction, a KG is a directed graph that models two information types of users (i.e., attributes' values and relationships) by using its edges. Figure. 1.1 reports a sample KG illustrating attributes (e.g., *age*, *job*) and relationships (e.g., $follows$) of four people (e.g., Ken, $Henry$, $Mary$, and Tom). By using KGs, a data provider can share both types of information together. Although the information KGs convey can leverage the development of many data-driven applications, such as machine learning, adversaries can exploit both information types to infer sensitive information of their victims.

Our research has mainly focused on proposing models for protecting users' identities in anonymized KGs and anonymization algorithms for generating these KGs. As KGs contain two types of users' information, we need to consider previous proposals for anonymizing both of them. Therefore, in this chapter, we first review anonymization approaches for users' attributes, which have been proposed for relational data (Section 2.2). Then, in Section 2.3, we review the main approaches proposed so far for anonymizing relationships in graphs. To the best of our knowledge, we are the first aiming at preserving users' privacy in published KGs. We therefore survey in Section 2.4 different attacks on users' privacy that can be done by using information on KGs.

Since the thesis also addresses privacy issues arising from the sequential publishing of KGs, Section 2.5 is devoted to a review of the techniques proposed in the literature to deal with this issue, for both relational data and graphs. Finally, we analyze personalized anonymization solutions for relational data and graphs in Section 2.6.

2.2 Anonymization of Relational Data

Relational data organize attributes' values of users as a table of rows and columns. Each column models an attribute and a row represents attributes' values of a user. When tables contain personal information, attributes can be classified into four types: explicit identifiers, quasi-identifiers, sensitive attributes, and non-sensitive attributes. Explicit identifiers (e.g.,

name) model information that can explicitly identify the real-life identity of users. Quasi-identifiers (QIDs) (e.g., *age*, *job*) are attributes that adversaries can exploit to re-identify users. Sensitive attributes (e.g., *salary*, *disease*) model users' sensitive information. Non-sensitive attributes are attributes that do not belong to any of the other types. To share users' attributes, data providers often remove explicit identifiers of users from the table. However, in 2002, Sweeney et al. [49] showed that even though explicit identifiers of patients in a health insurance dataset were removed, she can re-identify some these patients. Her re-identification is done by linking the health insurance dataset with a twenty-dollar voter registration list using three QIDs: *age*, *birthdate*, and *sex*. By using these QIDs, she reported that she can identify rows of the voters in the health insurance dataset and infer the *disease* of these voters.

k-anonymity [49] is the first model that protects users in anonymized relational data from being re-identified even if adversaries exploit QIDs of these users. The model requires that the original data must be modified such that the QIDs' values of each user are indistinguishable from those of $k-1$ other users, where k is a parameter. Therefore, the adversaries cannot re-identify any user with a confidence higher than $\frac{1}{k}$ even if they exploit these QIDs. Such modifications downgrade the quality of anonymized data satisfying requirements of k-anonymity. The higher value of k is, the more the data lose their quality. Therefore, anonymization algorithms must preserve the quality of k-anonymous relational data as much as possible.

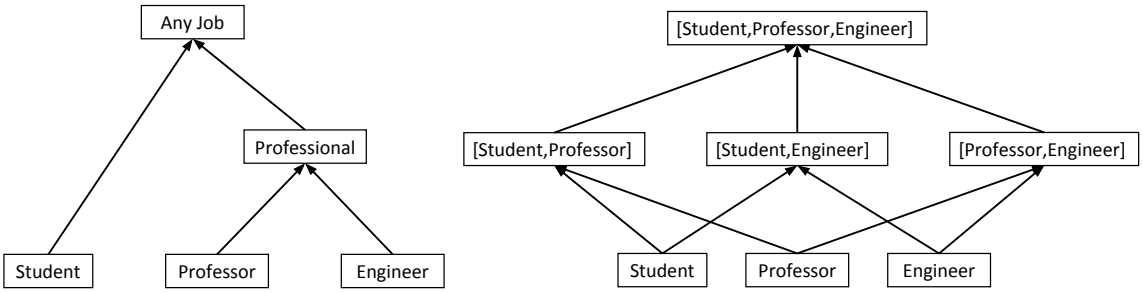
Many anonymization algorithms [1, 15, 31, 44, 52] have been presented to generate high-quality k-anonymous relational data. Unfortunately, the complexity of searching for the highest quality k-anonymous data is NP-hard [15, 52]. To cope with this issue, all of these algorithms use different heuristics to balance between the quality of anonymized data and their utility. For instance, LeFevre et al. [31] presented a practical framework for implementing k-anonymity anonymization algorithms. Then, Aggrawal et al. [44] proposed an $O(k)$-approximation anonymization algorithm which was further improved to $O(\log k)$ in [1]. In addition, Wang et al. [52] presented the Bottom-Up Generalization Algorithm (BUG) which adapts interactive bottom-up generalization from data mining to anonymize relational data. Then, Fung et al. [15] proposed the Top-Down Specialization (TDS) Algorithm which extends BUG to handle both categorical and continuous QIDs. Hoang et al. [20] further presented the Indexed Bottom-Up Generalization Algorithm to improve performance of BUG and TDS. To obtain k-anonymized relational data, the above mentioned algorithms use different heuristics to keep anonymizing the original data until for every user, his/her QIDs' values are identical to those of other $k-1$ other users. A serious weakness of these algorithms, however, is that they anonymize relational data in the same way without considering the data's distribution [42]. They can generate good quality anonymized data for some data and bad quality one for other data. As a result, data providers must have knowledge of all anonymization algorithms to choose the appropriate algorithm for their data or keep trying different algorithms until the anonymized data are good enough and this is a quite difficult task.

This led [2, 6, 11, 42] to use cluster-based approaches to generate k-anonymized relational data. Their key idea is to tailor existing clustering algorithms (i.e., k-means,

DBSCAN [14]) to generate clusters that have at least k users. Then, these algorithms modify the original data such that QIDs' values of users in the same clusters are identical. These algorithms use the information loss of modifying QIDs' values of two users as the distance between these users. The higher information these users lose, the higher their distance. Since the clustering algorithms gather users whose distance is close to the same cluster, these anonymization algorithms can minimize the information loss of users in the anonymized data. [2, 6, 42] developed extensions of the traditional clustering algorithm, i.e., k-means [14], to anonymize relational data. An extended version of DBSCAN [14] for anonymizing relational data is proposed in [11]. Not only k-means and DBSCAN [14] but also more recent clustering algorithms can be easily tailored to anonymize data. Since these algorithms rely on clustering algorithms, data providers can choose the appropriate anonymization algorithm based on their knowledge of clustering algorithms. For instance, if the providers know that k-means is good to perform clustering on their dataset, they can choose the k-means based anonymization algorithms to generate the anonymized version of the dataset. Moreover, this approach is flexible because it allows data providers to change the distance metric to generate high quality anonymized data.

All anonymization algorithms achieving k-anonymity generalize QIDs' values to generate anonymized relational data. Generalization replaces the original values with more general ones, by leveraging a user-defined taxonomy tree to map values of categorical QIDs with their more general versions. A taxonomy tree for a QID is a tree where all values of the QID are its leaf nodes and their parent nodes illustrate their more general values. Figure. 2.1a shows a user-defined taxonomy tree for *Job*. In this figure, $Professor$ and $Engineer$ can be generalized to $Professional$. For continuous QIDs, exact values can be generalized with an interval that covers these values. For instance, value 18 of *age* can be generalized to $[18, 25]$. If taxonomy trees for these continuous QIDs are provided, the generalization of their values is similar to that of the categorical ones. In case these taxonomy trees are missing, [15] analyzes the original data to generate the taxonomy trees for continuous QIDs at runtime. The biggest issue of generalization is that data providers must specify a taxonomy tree for each categorical QID in their original data. Furthermore, designing taxonomy trees that fit different usage purposes of data recipients is hard. To cope with these limitations, Nergiz et al. [42] presented the Natural Domain Generalization Hierarchy (NDGH). NDGH automatically generates a taxonomy tree for each QID by searching for all values of that QID in the original data and combining these values together. Figure. 2.1 illustrates a user-defined taxonomy tree and NDGH for *Job*. The main difference between these trees is that each leaf node in NDGH can have multiple parent nodes. As a result, anonymization algorithms using NDGH can choose different ways to generalize QIDs' values of users. Therefore, they are more flexible and their resulting anonymized data can fit many different usage purposes.

k-anonymity [49] fails to prevent adversaries from inferring users' sensitive values in particular settings. In particular, let an equivalence class be a set of users whose QIDs' values are identical. If an anonymized table satisfies k-anonymity, each user in the table belongs to an equivalence class that have at least k users. However, if all users in an equivalence class have the same value for their sensitive attributes, the adversaries can



(a) User-Defined Taxonomy Tree (b) Natural Domain Generalization Hierarchy

Figure 2.1: Taxonomy tree of *Job* attribute.

easily infer the sensitive values of these users. This attack, known as Homogeneity Attack, led Machanavajjhala et al. [36] to develop l-diversity to protect users' sensitive values. To this end, l-diversity requires that for every equivalence class in an anonymized table, it contains at least l distinct values for each sensitive attribute, where l is a positive number provided by data providers, with $l \leq k$. As a result, adversaries cannot infer users' sensitive values with a confidence higher than $\frac{1}{l}$. Here, l is always less than or equal to k. [36] also introduced the Entropy l-diversity and Recursive $(c-l)$-diversity to protect the sensitive values when the adversaries exploit the entropy and frequency of these values. Unfortunately, l-diversity cannot prevent adversaries from inferring users' sensitive values if these values are skewed or semantically similar [33]. For instance, if sensitive values of an equivalence class are in Stomach diseases category, the adversaries can infer that all users in the equivalence class have stomach related diseases even though they do not know users' real disease. t-closeness [33] addressed this limitation by ensuring that, for every equivalence class, the distance between the distribution of its sensitive values and that of the whole data is no more than t, where t is a positive number provided by data providers. A similar protection model was presented in [53], namely (α, k)-anonymity, which requires that every equivalence class has at least k users and the relative frequency of its sensitive values is less than or equal to α, where α is a positive number provided by the providers. These models prevent the adversaries from exploiting the distribution of sensitive values.

The quality of an anonymized relational dataset is evaluated based on its usage purposes (e.g., data mining, classification). For instance, if the data is used to train a classifier, [4, 15, 25, 52] evaluate its quality by measuring the difference between the Classification Error (CE) of classifiers trained by using it and its original version. Unfortunately, data recipients can use the data in many other ways such as clustering, data analysis. In these situations, CE cannot evaluate the quality of anonymized data. Therefore, most of previous work use the General Information Loss Metric (LM) [25] to evaluate the quality of their anonymized relational data or use both LM and CE at the same time. The intuition of LM is that the higher users' generalized value is in the taxonomy tree, the more information they lose.

Also, the Ambiguity Metric (AM') is presented in [42] for general-purpose anonymiza-

tion but it considers the number of tuples whose attributes' values are equal to those of a generalized tuple. The higher the number of tuples is, the more information the anonymized data loses. Unfortunately, LM and AM are not good enough if data providers know the usage purpose of their anonymized data [25]. Therefore, the Classification Metric (CM) [25] has been presented to improve the quality of anonymized data for training a classifier. CM penalizes a tuple if its class is different from the majority class of tuples in its group. Similarly, Bayardo et al. [4] presented the Discernibility Metric (DM) for anonymizing relational data for training usage but DM penalizes a tuple based on how many tuples in the anonymized data are indistinguishable from it. The data providers thus can choose an appropriate information loss metric based on the usage purpose of their anonymized data.

Although k-anonymity and its variants can protect different types of users' sensitive information, it cannot be applied if data providers cannot identify QIDs. In practice, it is hard for data providers to specify QIDs because the background knowledge of adversaries is unknown. Choosing too many attributes as QIDs results in high information loss of anonymized data, whereas too few QIDs can compromise users' privacy [15]. Dwork et al. [13] presented ϵ-Differential Privacy (ϵ-DP) to remedy this limitation, where ϵ is a positive real number provided by the data providers. Consider a table and an algorithm that analyzes the table and generates statistical outputs about it. ϵ-DP protects a user's privacy by ensuring that whether or not he/she is in the table, the output of the algorithm is pretty similar. The similarity between these outputs are controlled by ϵ. The lower the value of ϵ, the more indistinguishable the outputs. Data providers can decrease ϵ if they want to increase the privacy protection of users. While k-anonymity protects users based on the assumption about the background knowledge of adversaries about the users, ϵ-DP does it by assuming what type of analysis these adversaries can use. Therefore, ϵ-DP is good for interactive paradigm since data providers can restrict supported analysis types which satisfy requirements of ϵ-DP. However, it cannot protect anonymized data in non-interactive settings, because after releasing the data, the providers lose their control on how the data are analyzed. Since we focus on KGs, which are new and there are many types of analysis that can be performed over there (e.g., graph analysis, deep learning), this limitation can prevent researchers from developing data-driven applications. As a result, in this thesis, we focus on using k-anonymity to anonymize KGs and leave the DP approach for KGs for future work.

2.3 Anonymization of Graphs

Graphs use nodes and edges to represent users and their relationships, respectively. Unfortunately, protecting users' privacy in these graphs is more challenging than the protection of relational data, because adversaries can exploit not only victims' information but also information about those having relationships with the victims.

Liu et al. [34] was among the first to present a protection model, namely k-degree (k-da), to protect users' identities in undirected graphs against *degree attack*. *Degree attack* assumes that adversaries can know the number of relationships their victim has. Thus,

by comparing the number of relationships with degrees (i.e., the number of edges that are incident to a node) of nodes in the anonymized graphs, they can identify which node represents their victim. To this end, k-da ensures that for every node in the anonymized graph, its degree is indistinguishable from that of $k-1$ other nodes.

Another protection model, i.e. k-neighborhood (k-na), was presented in [61] to prevent users from being re-identified by *neighborhood attack*. A *Neighborhood attack* assumes that adversaries know neighbors of their victim. These adversaries then can compare the neighbors of their victim and neighbors of nodes in the anonymized graph to re-identify which node is their victim. k-na prevents this attack by ensuring that for every node in the anonymized graph, its neighbors are indistinguishable from those of $k-1$ other nodes.

Moreover, Hay et al. [18] presented their anonymization solution against *subgraph attack* and *hub fingerprint attack*. *Subgraph attack* assumes that adversaries can use any subgraph containing the node of their victim to re-identify the victim. *Hub fingerprint attack* relies on the hub fingerprint of the victim. A hub is a node that has high degree and high betweenness centrality (e.g., the proportion of shortest paths in the network that include the node). As it is hard for adversaries to collect neighbors' information of many users, [18] stated that a node can only reach a hub if the distance between them is less than the maximum distance. Then, a hub fingerprint of a node is the shortest path length from the node to a set of reachable hubs. To resist these attacks, [18] splits nodes into partitions and generates the anonymized graph which contains super nodes, one for each partition, and super edges representing relationships between these super nodes.

All the above mentioned work [18,34,61] led Zou et al. [64] to develop k-automorphism (k-auto) which protects users' identities in undirected graphs against *degree attack* [34], *neighborhood attack* [61], *subgraph attack* [18], and *hub fingerprint attack* [18]. A similar work is presented in [12], namely k-isomorphism. Their intuition is that for every node in the anonymized graph, its structural information should be isomorphic with that of at least $k-1$ other nodes in the graph.

Recently, [10] presented the Paired k-degree, an extension of k-da, to protect users' identities in directed graphs from *degree attack*. This attack relies on the background knowledge about the number of outgoing and incoming relationships to re-identify the victim. Then, by comparing the number of outgoing and incoming relationships of the victim with the out- and in-degree (i.e., the number of outgoing and incoming edges, respectively) of nodes in an anonymized directed graph, adversaries can re-identify the victim's node. The Paired k-degree prevents this attack by requiring that for every node in an anonymized graph, its out- and in-degree are indistinguishable from those of $k-1$ other nodes in the graph. Zhang et al. [60] developed the K-In&Out-Degree Anonymity (K-IODA) whose requirement is identical to that of the Paired k-degree. Similar to relational data, these models also require anonymization algorithms to modify the original graph's structure to change out- and in-degree of its nodes.

Different anonymization algorithms have been presented to generate anonymized undirected [34, 61, 64] and directed [10, 60] graphs satisfying requirements of the protection models described above. Liu et al. [34] proposed a two-step anonymization algorithm for k-da [34]. The algorithm first finds the sequence of nodes' degrees in the original undirected

graph. Then, it modifies the generated sequence such that for every degree in the sequence, there are at least $k-1$ other identical degrees. Finally, it adds and removes edges to ensure the degrees of nodes in the graph are identical to those in the sequence. Unfortunately, this algorithm does not always generate anonymized graphs satisfying k-da [34].

Another two-step anonymization algorithm for k-na is presented in [61]. Its first step extract neighborhoods of nodes in the original graph. Since each neighborhood is a subgraph of the original one, the algorithm represents it by using a code, i.e., the set of edges in the subgraph. The code is generated such that nodes whose neighborhood are identical have the same code. Then, the second step gathers users that have similar code to the same group until all groups have at least k users and adds fake edges to ensure all users in the same group have the same code.

The anonymization algorithm of k-auto [64] ensures that for every user, his/her neighborhood is isomorphic with the neighborhood of at least $k-1$ other users. To this end, it first partitions the original graph into blocks and gather these blocks into groups such that each group has at least k blocks. Then, for each group, it adds fake edges to make blocks in the same group isomorphic.

There are a few other approaches [40, 48] to anonymize graphs. Stokes et al. [48] extended k-means [14] to anonymize undirected graphs. A random walk based anonymization algorithm has been presented in [40]. It randomly modifies edges of nodes to hide their real edges. Nevertheless, these algorithms only support undirected graphs.

There are two proposals we are aware of dealing with the anonymization of directed graphs. Casas et al. [10] presented the Directed Graph Anonymization Algorithm (DGA), an extension of the algorithm in [34], to anonymize directed graphs satisfying Paired k-da [10]. As the Paired k-da considers both out-degree and in-degree, DGA generated two degree sequences for out-degree and in-degree, respectively. To increase the chances of generating anonymized directed graphs, DGA extended edge addition/switch from [34] and presented their new technique, i.e. edge extension. [60] proposed DSNDG-KIODA, to anonymize directed graphs satisfying the requirements of K-IODA [60]. To preserve the community structure of anonymized graphs, DSNDG-KIODA only adds fake edges and nodes to anonymize the graphs. Unfortunately, as showed in Chapter 3, both DGA and DSNDG-KIODA cannot always generate anonymized directed graphs.

The difference between the structural properties of anonymized graphs and their original versions are used to measure the quality of anonymized graphs. The higher the difference is, the lower the quality of these graphs is. The most fundamental property is degree distribution. This property is measured by using the Total Degree Difference [12, 34, 64] (i.e., the sum of the difference between degrees of users in anonymized graphs and their original versions), the number of added fake edges [61], and the number of intersection edges [10] (i.e., the number of edges existed in both a directed graph and its anonymized version). If anonymization algorithms add fake users, the number of added fake users [60] are often used. Then, other properties of graphs such as Clustering Coefficient, Path Length are also used to measure the quality of anonymized graphs. These properties are measured by using metrics like the Average Clustering Coefficient, which is used to measure the average of local clustering coefficient of nodes, and the Average Path Length, which is the

average number of steps along the shortest paths for all possible pairs of network nodes, are used to measure the quality of anonymized graphs in [10, 12, 34, 60, 64].

Even though many metrics have been used to evaluate the quality of anonymized graphs, most of the previous work [10, 12, 34, 61, 64] use the difference between degrees of users in anonymized and original graphs as the information loss of these users. To minimize the difference, [10, 34] use the difference as a cost function to generate the degree sequence. Other work [12, 60, 61, 64] use different heuristics to minimize these differences. Zou et al. [64] uses graph partitioning and block alignment, to reduce the number of edges they need to copy in the last step of their algorithm.

Data providers often assign a label for each node to illustrate the sensitive value (e.g., *spammer*, *salary*, *disease*) of its corresponding user. These node-labelled graphs are often used in graph analysis applications such as spammer detection and disease analysis. However, the above mentioned work [10, 18, 34, 60, 61, 64] cannot protect users' labels. To cope with this issue, Yuan et al. [58] presented k-degree-l-diversity (k-da-l-diver) to generate anonymized undirected graphs satisfying both k-da [34] and l-diversity [36]. k-da-l-diver ensures that for every node in the anonymized graph, there exist at least $k-1$ other nodes having the same degree in the anonymized graph and the nodes having the same degree contain at least l distinct labels. [58] anonymizes undirected graphs in two steps. In the first step, it generates a sequence of pairs of degree and label of nodes such that for every pair, its degree is indistinguishable from that of other $k-1$ other pairs, and pairs having the same degree contain at least l distinct labels. The second step adds fake edges to ensure that the degree of every node in the anonymized graph is equal to that of the node in the generated sequence.

We will show in Chapter 3 that our Cluster-Based Anonymization Algorithm (CDGA) always generated anonymized directed graphs satisfying both the Paired k-da [10] and K-In&Out-Degree Anonymity [60]. Different from DGA [10], we present a new technique, the Degree Decrement, to decrease the out- or in-degree of users in the anonymized graph. Also, we prove that our algorithm can always generate an anonymized graph satisfying both the Paired k-da [10] and K-IODA [60].

Table. 2.1 summarizes the graph anonymization proposals we have reviewed so far.

Table 2.1: Analysis of existing graph anonymization work. (UG=Undirected Graph, DG=Directed Graph, Iden=Identity, Attr=Attribute, Deg=Degree, Neg=Neighborhood, Subg=Subgraph, HubFig=Hub Fingerprint, EA=Edge Addition, ED=Edge Deletion, NA=Node Addition, ND=Node Deletion)

	Type		Proctection		Background Knowledge				Generalization			
	UG	DG	Iden	Attr	Deg	Neg	SubG	HubFig	EA	ED	NA	ND
k-da [34]	✓		✓		✓				✓	✓		
k-na [61]	✓		✓			✓			✓			
Hay et al. [18]	✓		✓				✓	✓	✓			
k-iso [12]	✓		✓		✓	✓	✓	✓	✓			
k-auto [64]	✓		✓		✓	✓	✓	✓	✓			
Stokes et al. [48]	✓		✓		✓				✓	✓		
Paired k-da [10]		✓	✓		✓				✓			
K-IODA [60]		✓	✓		✓				✓		✓	
k-da-l-diver [58]			✓	✓	✓				✓	✓	✓	✓

2.4 De-anonymization of Knowledge Graphs

Many work [45–47, 51, 62] have focused on investigating attacks to the privacy of users in KGs. These proposals showed that adversaries can re-identify users in naive-anonymized KGs generated by removing users' identities. Qian et al. [47] presented their two-step attack on KGs to re-identify and infer sensitive values of users in KGs. In this work, they aimed at re-identifying users in the KGs by exploiting two information types of the users (i.e., attributes' values and relationships). To this end, they presented the Node Similarity Metric that measures the similarity between two users based on their nodes' attributes and relationships. Then, they exploit the metric to find the similarity between their victims and users in the KGs. If the similarity is greater than the specified threshold, the user is the match of the victim in the anonymized KG. In the first step, they pick a random victim and find a matched user in the anonymized KG. Then, for each neighbor of the victim, they find his/her matched user by looking into the neighbors of the matched user. They keep doing these steps until all victims are matched. At this time, they can re-identify all of their victims. [45] extended this attack to improve its accuracy. The extension improves the algorithm to match neighbors of the victim. Furthermore, in [46], they investigated the correlation between the accuracy of their algorithm and the amount of information about the victims. Their conducted experiments revealed that the accuracy of de-anonymization is not necessarily monotone increasing with the amount of information about the victims.

In addition, DeepLink [62] leveraged deep learning techniques to link users of two KGs. One KG contains users' information (e.g., attributes and relationships) in an OSN and the other consists of the information of these users in another OSN. First, DeepLink learns a function to generate a vector for each user based on their information. Then, it learns another function to decide whether two vectors belong to the same user. These learned functions help DeepLink to generate vectors of unseen users based on their attributes' val-

ues and relationships. Then, by using these generated vectors, DeepLink decides whether two users are the same person. A similar work is presented in [51], namely DeepMatch. DeepMatch first defines a set of pre-defined users in a KG. Then, for every pre-defined user, it finds his/her corresponding user in other KG by using the same approach of DeepLink. For each pair of a user and his/her corresponding one, DeepMatch finds the corresponding users of these users' neighbors by checking neighbors of the user with those of its corresponding one. Since users often have the same neighbors when they use different OSNs, DeepMatch can increase the probability of re-identifying these neighbors instead of naive checking for all pairs of users in DeepLink [51].

Unfortunately, previous anonymization work for relational data [1, 15, 42, 44, 49, 52] and graphs [10, 12, 34, 60, 61, 64] cannot be applied as they are to anonymize KGs, since KGs consist of both users' attributes and relationships. Chapter 4 shows in details our anonymization approach for KGs. Our approach prevents adversaries from re-identifying users by exploiting attributes' values and relationships of the users.

2.5 Sequential Anonymization of Released Datasets

All of the work we have revised so far allow data providers to publish an anonymized version of their data. However, if they update their original data and publish the anonymized version of the updated dataset, adversaries can exploit these published versions to attack users' privacy. For instance, let us assume that a data provider publishes two anonymized datasets that satisfy 3-anonymity at time 1 and 2. The provider publishes the anonymized dataset at time 2 after updating information of its users. If adversaries know that a user, say Ken, is in both datasets and they only have access to one of them, they cannot re-identify Ken with a confidence higher than $\frac{1}{3}$. However, if they have access to both anonymized datasets and know that Ken has just changed his job from $Student$ to $Engineer$, they can look into these datasets to check how many users have the same change of their job. Ken can be the only user who changes his Job. In this situation, the adversaries can re-identify Ken in both datasets. Furthermore, adversaries can exploit not only update but also insertion, deletion, and re-insertion operations, as we will describe in more details in Chapter 5.

There is some work addressing this issue for relational data. Byun et al. [8] was among the first to protect users' identities and sensitive values when data providers insert new users into their original relational data and publish the anonymized version of the updated data. This work is an extension of k-anonymity [49] and l-diversity [36]. First, they add new users to equivalence classes of the previous anonymized data if QIDs' values of these users are equal to those of these classes. If they cannot find any equivalence class that can contain these users, they create a new equivalence class for these users. Then, they split equivalence classes that have more than $2 \times l$ distinct sensitive values. Finally, they modify the updated data to ensure that users in the same equivalence class have the same QIDs' values. Unfortunately, this work only allows data providers to insert new tuples.

To cope with this issue, m-invariance [55] was presented to allow the providers to insert

and remove users from their data, where m is a positive integer provided by data providers. Here, k and m have the same meaning. Both of them are used to control the maximum confidence that adversaries can re-identify users in anonymized data. Let the signature of an equivalence class be the set of sensitive values of users in the class. The signature of a user is the signature of his/her equivalence class. This work assumes that adversaries can have access to previous anonymized tables and they can exploit their victims' signatures. Then, an anonymized table is m-unique if each equivalence class has at least m users and all of these users have different sensitive values. Furthermore, a sequence of anonymized versions of a table is m-invariant if all of these versions satisfy m-uniqueness and for every user existing at least once in these versions, his/her signature in all the published versions is identical. This work generates anonymized tables in four steps: division, balancing, assignment, and split. Division step splits users in the original table into buckets containing users whose signature is identical. Balancing step ensures that all buckets are balanced, i.e. having the same number of users for each of their sensitive values. If they cannot find users to add to these buckets, they add fake users to these buckets. Assignment step adds the remaining users to these buckets if these buckets are still balanced after adding these users. Finally, split step splits these buckets to ensure that the resulting buckets have only m users. This work ensures that adversaries cannot re-identify any user and infer his/her sensitive values with a confidence higher than $\frac{1}{m}$ when the adversaries know QIDs' values of users in all published data. However, [55] does not consider the re-insertion scenario, that is, when a data provider re-inserts deleted users.

Anjum et al. [3] remedy this issue by proposing τ-safety to ensure that adversaries cannot infer the sensitive values of any user with a confidence higher than $\frac{1}{m}$ even though they know the event list τ of their victim. Here, τ describes the victim's events (i.e., inserted, re-inserted, updated, deleted) in the published datasets. τ-safety gives the same privacy protection of m-invariance even though data providers insert/update/delete/re-insert tuples on their data. They address the limitation of m-invariance by monitoring not only current users but also removed users. Then, when the providers re-insert a user, they ensure that his/her signature is identical to that of him/her when he/she is deleted. Unfortunately, Zhu et al. [63] showed that when the providers update the sensitive value of their users, the updated values change the signatures of these users. The same situation can happen when the providers re-insert users whose sensitive value is different from that of them when they are deleted. Since τ-safety requires users' signatures to be identical in all published data, these users' data cannot be published. Therefore, they presented τ-safe (l, k)-diversity [63] to cope with this problem, where l, k are two positive integers provided by data providers such that l is less than or equal to k. A series of anonymized tables satisfies τ-safe (l, k)-diversity if all of its table satisfies (l, k)-diversity (i.e., satisfying k-anonymity and l-diversity) and for every user existing at least once in the series, the intersection of all of his/her signatures in these tables are empty. Then, Zhu et al. [63] showed that τ-safe (l, k)-diversity can give the same privacy protection with τ-safety [3] and m-invariance [55] even in the new scenario.

The issue of dynamic release of anonymized data has also been addressed for undirected [37, 39, 50] and directed graphs [60]. Medforth et al. [39] presented the degree-trail attack

that helps adversaries to re-identify users in anonymized undirected graphs by monitoring users' degrees in these graphs and proposed their protection solution against this attack. They assume that a provider inserts new nodes to their graph and adversaries can have access to all anonymized graphs. By combining these graphs, the adversaries can re-identify their victims. To protect anonymized graphs from this attack, they propose to randomly add m edges and remove m-edges from the original graph, where m is a positive integer provided by data providers. However, randomly adding edges can accidentally add removed edges of the previous graph. Similarly, removing edges randomly can remove added edges of the previous one. Then, the adversaries can compare the current anonymized graph and its previous version to infer removed real edges and added fake ones. To address these issues, this work ensures that they only add and remove edges which do not exist in the previous graphs. Since the random edges modifies users' degrees, adversaries cannot re-identify their victims. However, this work does not formalize the underlying protection model.

Tai et al. [50] proposed the k^w-SDA model to protect users from being re-identified when adversaries monitor users' degrees in w continuous anonymized undirected graphs. They assume that data providers can insert new nodes and edges to their new graph and each user can have many labels which illustrates multiple values of his/her attribute. For instance, a user can have two labels: flu and $SARS$ at the same time. First, they define a k-shielding consistent group in w continuous anonymized graphs as the group consisting of at least k users whose degree is identical and the intersection of its users' labels has at least k distinct values. Then, their algorithm adds fake edges and nodes to generate anonymized undirected graphs such that all users appearing in w continuous anonymized graphs are in a k-shielding consistent group.

Macwan et al. [37] extended [50] to allow users to have many attributes. This work assumes that each node has a label representing its attributes' values. For instance, the label $< 21, M, US >$ models that *age*, *birthday*, and *location* of a user are 21, M, and US, respectively. They ensure that there are at least k nodes having the same label. Their algorithm first partitions the original graph into clusters, such that each cluster has at least k nodes and a label. If there are more than k new nodes, it creates a new cluster for them. Then, it merges clusters that have less than k users or add fake nodes to ensure that all resulting clusters have at least k users. Unfortunately, this work only protects users' labels and does not prevent adversaries from re-identifying these users [34]. Furthermore, this work does not consider anonymizing directed graphs.

Recently, Zhang et al. [60] presented the Dynamic Social Network Directed Graph K-In&Out-Degree Anonymity Model (DSNDG K-IODA) to protect users' identities when the adversaries gain access to continuous anonymized directed graphs. Similar to [10], this work first partitions the original graph into clusters that have at least k nodes. Then, it adds fake nodes and fake edges to ensure that the out-/in-degrees of users in the same clusters are identical. However, this work does not allow nodes to have attributes.

The above mentioned work [8, 37, 39, 50, 55, 60] cannot protect users in anonymized KGs. The first issue is that none of the work considers anonymizing two information types: attributes' values and relationship types at the same time. Furthermore, anonymization

solutions for undirected [37, 39, 50] and directed [60] graphs only support the impractical scenario where data providers only add new edges and nodes to their graphs. Table. 2.2 summarizes the sequentially anonymization solutions for graphs we have discussed so far.

Table 2.2: Analysis of existing approaches for sequentially anonymization of graphs. (UG=Undirected Graph, DG=Directed Graph, Iden=Identity, Attr=Attribute, Deg=Degree, EA=Edge Addition, ED=Edge Deletion, NA=Node Addition, ND=Node Deletion)

	Type		Proctection		Background Knowledge		Generalization			
	UG	DG	Iden	Attr	Degree	Attr	EA	ED	NA	ND
Medforth et al. [39]	✓		✓				✓	✓		
Tai et al. [50]	✓		✓	✓	✓	✓	✓		✓	
Macwan et al. [37]	✓			✓	✓	✓	✓		✓	
Zhang et al. [60]		✓	✓		✓		✓		✓	

We present a solution to cope with these issues in Chapter 5.

2.6 Personalized Anonymization

Users can have different privacy protection requirements [54]. For example, a famous user may require high level of privacy protection. In contrast, a normal one who does not know technologies may not care about his/her privacy and he/she may choose to have low levels of protection. However, all of the anonymization solutions that have been discussed before [3, 10, 12, 33, 34, 36, 49, 55, 60, 61, 63, 64] apply the same protection level to all users in their data. This limitation does not allow data providers to protect users who require strong privacy protection levels, without taking the risks of generating low-quality anonymized data. The low-quality data are generated by setting strong protection levels for all users in their data.

Xiao et al. [54] presented the first personalized anonymization approach for relational data. This work allows users to specify a guarding value for their sensitive values. A guarding value is an intermediate node in the taxonomy tree of a sensitive attribute. They ensure that the anonymized sensitive value of a user cannot be the successors of his/her guarding value. For example, with reference to Figure Figure. 2.1a if a user choose his/her guarding value as $Professonal$, his/her anonymized sensitive value cannot be $Professor$ or $Engineer$. Anonymized data are generated such that all users' sensitive values are not the successors of their guarding values. Next, Ye et al. [56] proposed the personalized version of (α, k)-privacy [53] to allow users to specify their own values of α and k. Then, Liu et al. [35] combined solutions in [54] and [56] to allow users to specify not only the guarding value for their sensitive values but also their values of k and l.

The first personalized anonymization approach for undirected graphs has been presented in [59]. This work allows users to specify one of three supported protection levels. Level 1 for users who want to protect their identities when adversaries know their attributes' values.

Users whose level is 2 want to be protected from being re-identified when adversaries know both their attributes' values and degree. The final level, i.e. level 3, is for those who want to protect their identities when adversaries know their attributes' values, degree, and edges adjacent to their nodes. To anonymize the data, this work firstly applies k-anonymity [49] to protect users who requires protection level 1. Then, k-da [34] is used to anonymized users' relationships based on anonymized results from k-anonymity. Finally, it provides an additional protection for level 3 by modifying results of k-da. Unfortunately, this work anonymizes relationships based on anonymized attributes' values. This limitation results in a huge amount of information loss for users' relationships. In addition, Jian et al. [27] presented the personalized version of k-da-l-diver [58] to allow users to specify their values of k and l. Unfortunately, all the work [27, 59] only support one type of undirected relationships. Therefore, this work cannot be used to anonymize KGs.

In Section 6.2.1, we show our personalized anonymization approach for KGs.

Chapter 3

Anonymization of Directed Graphs

3.1 Introduction

Many protection models (e.g., k-degree [34], k-na [61], and k-auto [64]) have been presented to protect users' identities in anonymized undirected graphs. These models ensure that even if adversaries know structural information (e.g., nodes' degrees) of any user in the anonymized graph, the confidence of their prediction will be less than $\frac{1}{k}$, where k is a positive integer provided by data providers. However, these models only protect undirected graphs, while most of the popular relationships in social networks are directed. For example, Twitter, Facebook, and Instagram users have directed relationships (e.g., $follow$).

Recently, few protection models (e.g., the Paired k-degree [10] and K-in&out-Degree anonymity [60]) have been presented to protect users in directed graphs. Different from undirected graphs, the directed ones contain not only the outgoing but also the incoming edges. Therefore, these models require that the out- and in-degree of any user in the anonymized graph are indistinguishable from those of $k-1$ other users. Then, they can ensure that adversaries cannot discover the identities of any user in the anonymized graphs with the confidence higher than $\frac{1}{k}$ even if they know the out- and in-degree of the target users.

Although the Paired k-degree [10] and K-in&out-Degree anonymity [60] have the same privacy requirements, their authors propose different algorithms to anonymize the graphs. The Directed Graph Anonymization Algorithm (DGA) [10] modifies edges until the resulting graph satisfies the Paired k-degree [10] property. The Dynamic Social Network Directed Graph K-In&Out-Degree Anonymity Algorithm (DSNDG-KIODA) [60] adds fake edges and nodes to make the resulting graph satisfying the K-in&out-degree anonymity. Nevertheless, this approach decreases the quality of the anonymized graphs as the graphs contain not only fake edges but also fake nodes. Furthermore, none of them can prove that their algorithms always generate an anonymized graph satisfying the proposed privacy models [10, 60].

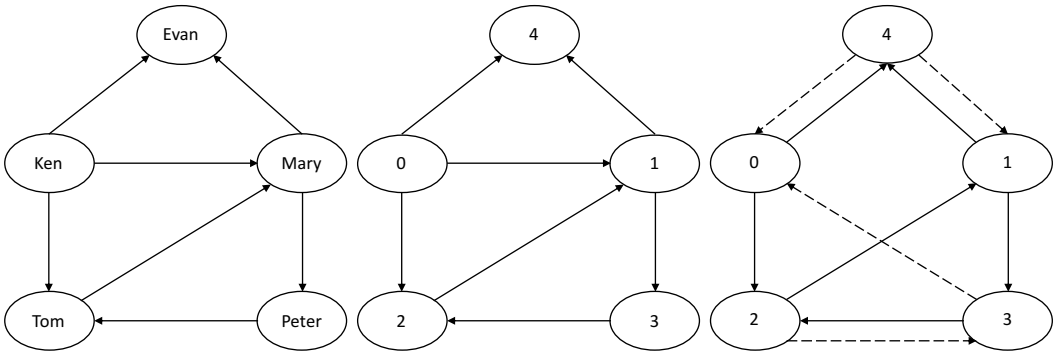
To cope with this issue, we present the Cluster-based Directed Graph Anonymization (CDGA) algorithm and prove that, with the appropriate parameters, our algorithm

can generate anonymized directed graphs satisfying both the Paired k-degree [10] and K-In&Out-Degree Anonymity [60]. CDGA groups users into clusters that have at least k users and modifies the original graphs in such a way that all users in the same cluster have the same out- and in-degree. We also present an additional modification technique, namely the Degree Decrement technique, to decrease the users' out- or in-degrees. We prove that CDGA always generates anonymized graphs satisfying both the Paired k-degree [10] and K-in&out-degree anonymity [60].

The remainder of this chapter is organized as follows. Section 3.2 illustrates the considered protection models for anonymizing directed graphs, whereas Section 3.3 describes our anonymization algorithm. Finally, we report the experimental results in Section 3.4.

3.2 Anonymizing Directed Graphs

In this chapter, we focus on anonymizing directed graphs representing users' relationships. Figure. 3.1a shows an example of directed graph. Let $\mathcal{G}(V, E)$ be a directed graph, where V is a set of nodes and E is a set of edges connecting these nodes. Each node in V represents a user. Each directed edge in E is a pair (u, v), $u, v \in V$, indicating that user u has a relationship with user v. Appendix A summarizes notations used in this chapter.



(a) Directed graph $\mathcal{G}$ (b) Directed graph $\overline{\mathcal{G}}_1$ (c) Directed graph $\overline{\mathcal{G}}_2$

Figure 3.1: An example of directed graph and its anonymized versions

3.2.1 Adversaries' Background Knowledge

Let $\mathcal{G}(V, E)$ be a directed graph and $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be its anonymized version. Similar to [10], we assume that the adversary background knowledge contains both the users' out- and in-degrees in the anonymized graph. The out-degree of a user refers to the number of edges outgoing from his/her node, whereas his/her in-degree represents the number of edges incoming to his/her node. More precisely, let u be a user in $\overline{V}$, we denote as $d_o(\overline{\mathcal{G}}, u) = |\{(u, v) \in \overline{E}\}|$ and $d_i(\overline{\mathcal{G}}, u) = |\{(v, u) \in \overline{E}\}|$ the user u's out- and in-degree

Table 3.1: Adversaries' background knowledge

		$\mathcal{G}$		$\mathcal{G}_1$		$\mathcal{G}_2$	
NodeID	Name	d_o	d_i	d_o	d_i	d_o	d_i
0	Ken	3	0	3	0	2	2
1	Mary	2	2	2	2	2	2
2	Tom	1	2	1	2	2	2
3	Peter	1	1	1	1	2	2
4	Evan	0	2	0	2	2	2

in the anonymized graph $\overline{\mathcal{G}}$, respectively. More formally, we define the background knowledge as follows:

Definition 1 (Adversaries' Background Knowledge [10])**.** *Let $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be an anonymized directed graph and u be a user in $\overline{V}$. The background knowledge that adversaries use to re-identity u is $I_d(\overline{\mathcal{G}}, u) = (d_o(\overline{\mathcal{G}}, u), d_i(\overline{\mathcal{G}}, u))$, where $d_o(\overline{\mathcal{G}}, u)$ and $d_i(\overline{\mathcal{G}}, u)$ are the user u's out- and in-degree extracted from the anonymized graph $\overline{\mathcal{G}}$.*

Example 2. Figure. 3.1 and Table. 3.1 illustrate two anonymized graphs: $\overline{\mathcal{G}}_1$, $\overline{\mathcal{G}}_2$ and the out- and in-degrees of all users in these graphs as well as in the original graph $\mathcal{G}$. To generate $\overline{\mathcal{G}}_1$, we only remove users' identities from $\mathcal{G}$. Then, we generate $\overline{\mathcal{G}}_2$ by modifying edges of $\overline{\mathcal{G}}_1$ such that the out- and in-degree of all users in $\overline{\mathcal{G}}_2$ are indistinguishable. $\overline{\mathcal{G}}_2$ satisfies the Paired 5-degree. Here, the adversaries can re-identify each user in $\overline{\mathcal{G}}_1$ as the background knowledge extracted from their structural data is unique. For instance, if the adversaries know Ken's out- and in-degree, they can identify his node as 0.

3.2.2 Anonymity of Directed Graphs

As the adversaries know the out- and in-degree of the victim, we need to consider both the degrees to protect users in the anonymized graphs. The Paired k-degree [10] and K-In&Out-Degree Anonymity [60] have been proposed to protect users from the knowledge of both the degrees. As their properties are identical, we only present the Paired k-degree as it is the first one being published.

Definition 2 (Paired k-degree)**.** *Let $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be an anonymized directed graph. $\overline{\mathcal{G}}$ satisfies the Paired k-degree if and only if for every user u in $\overline{V}$, there exists a set of users in $\overline{V}$, denoted $\zeta^{\overline{\mathcal{G}}}(u)$, such that: $\zeta^{\overline{\mathcal{G}}}(u) = \{v \in \overline{V} | I_d^{\overline{\mathcal{G}}}(v) = I_d^{\overline{\mathcal{G}}}(u)\}$ and $|\zeta^{\overline{\mathcal{G}}}(u)| \geq k$.*

The intuitive idea of the Paired k-degree is that the out- and in-degrees, i.e., $I^{\overline{\mathcal{G}}}$, of all users in the anonymized directed graph $\overline{\mathcal{G}}$ are indistinguishable from those of at least $k-1$ other whose information also appears in $\overline{\mathcal{G}}$.

Example 3. As showed in Example 2, even though $\overline{\mathcal{G}}_1$ does not contain any users' identifiers (e.g., *name*), the adversary can re-identify all the users. However, adversaries

cannot re-identify any user in $\overline{\mathcal{G}}_2$ with confidence higher than $\frac{1}{5}$, since all of the users have the same out- and in-degrees in $\overline{\mathcal{G}}_2$.

3.3 Cluster-based Anonymization

In this section, we first introduce the intuition of our Cluster-Based Directed Graph Anonymization Algorithm (CDGA), before presenting it in details.

3.3.1 Overview

We design CDGA to generate anonymized directed graphs satisfying the Paired k-degree while maximizing their quality. More precisely, given a directed graph $\mathcal{G}(V, E)$ and the positive number k, CDGA generates its Paired k-degree anonymized version $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ by modifying the structure of $\mathcal{G}$. Our algorithm has two main steps:

1. **Clusters generation**. The aim of this step is to generate a set of user clusters $C^{\overline{\mathcal{G}}} = \{c \subset \overline{V} || c| \geq k\}$. Our algorithm maximizes the quality of the generated clusters according to an information loss metric that will be discussed in details in Section 3.3.2.

2. **Directed graph generalization**. The goal of this step is to add and remove edges in $\mathcal{G}$ such that all users in the same cluster have the same out- and in-degree. More precisely, given the set of clusters $C^{\overline{\mathcal{G}}}$ generated in the previous step, we generate $\overline{\mathcal{G}}(\overline{V}, \overline{E})$, such that $\overline{V} = \bigcup_{c \in C^{\overline{\mathcal{G}}}} c$ and $\overline{E}$ satisfies the condition that $\forall c \in C^{\overline{\mathcal{G}}}$, $\forall u, v \in c$: $I_d(\overline{\mathcal{G}}, u) = I_d(\overline{\mathcal{G}}, v)$.

After performing these steps, we obtain $\mathcal{G}$'s Paired k-degree anonymized version $\overline{\mathcal{G}}$. In what follows, we first discuss the information loss metric we use to evaluate the quality of the anonymized graphs.

3.3.2 Information Loss Metric

To evaluate the loss of information on a user u, we measure the difference between his/her degree in the original and anonymized graph. Different from previous work [10, 60], we define two information loss metrics, namely DM'_o and DM'_i, to evaluate the differences between the users' out- and in-degrees in the anonymized and original graphs. This approach allows us to control the contributions of both the out- and in-degree information losses to the overall information loss DM.

More precisely, given a user u, DM'_o measures his/her out-degree information loss by calculating the difference between his/her out-degree in the original graph $\mathcal{G}$ and the anonymized one $\overline{\mathcal{G}}$. As the highest out-degree of user u is $|V|$, the maximum difference is also $|V|$. The metric then normalizes the difference to the range [0, 1], by dividing it by the maximum difference. We define DM'_o as follows:

Definition 3 (DM'_o)**.** *Let $\mathcal{G}(V, E)$ be a directed graph, $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be its anonymized version, and u be a user in $\overline{V}$. The out-degree loss metric (DM'_o) of anonymizing user u is:*

$$DM'^{\overline{\mathcal{G}}}_o(u) = \frac{|d_o(\overline{\mathcal{G}}, u) - d_o(\mathcal{G}, u)|}{|V|}$$

Similarly, we define DM'_i as follows:

Definition 4 (DM'_i)**.** *Let $\mathcal{G}(V, E)$ be a directed graph, $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be its anonymized version, and u be a user in $\overline{V}$. The in-degree loss metric (DM'_i) of anonymizing user u is:*

$$DM'^{\overline{\mathcal{G}}}_i(u) = \frac{|d_i(\overline{\mathcal{G}}, u) - d_i(\mathcal{G}, u)|}{|V|}$$

Since we aim to exploit these metrics to evaluate the information loss of making the out- and in-degree of two users identical, we introduce this further definition:

Definition 5 (DM)**.** *Let $\mathcal{G}$ be a directed graph, $\overline{\mathcal{G}}$ be its anonymized version, and u, v be two users in $\overline{V}$. The Out- and In-degree Information Loss Metric (DM) of making u and v having the same out- and in-degree is:*

$$DM^{\overline{\mathcal{G}}}(u, v) = \alpha \times DM^{\overline{\mathcal{G}}}_o(u, v) + (1 - \alpha) \times DM^{\overline{\mathcal{G}}}_i(u, v)$$

where:

$$DM^{\overline{\mathcal{G}}}_o(u, v) = \frac{DM'^{\overline{\mathcal{G}}}_o(u) + DM'^{\overline{\mathcal{G}}}_o(v)}{2}$$

$$DM^{\overline{\mathcal{G}}}_i(u, v) = \frac{DM'^{\overline{\mathcal{G}}}_i(u) + DM'^{\overline{\mathcal{G}}}_i(v)}{2}$$

and α is a number between 0 and 1 to control the contribution of out- and in-degree information losses.

While some directed graph analysis techniques (e.g., spam detection) rely on either outgoing or incoming edges, anonymizing the graph changes both users' out- and in-degrees. For example, as spammers follow as many users as they can to access the users' information [24], their out-degree is higher than those of regular users. Changing too many outgoing edges decreases the accuracy of the spam detection technique. Therefore, by using a parameter α, we allow data providers to control the changes of the users' out- and in-degree. If the providers want to prioritize preserving the out-degree, they can assign α to a value larger than 0.5. On the other hand, they can set it a value less than 0.5 if they want to prioritize minimizing the differences of the users' in-degrees. As we do not assume how the anonymized graphs generated by our algorithm will be used, we assign α to 0.5 to balance the out- and in-degree information losses.

Example 4. Let us assume $\alpha = 0.5$, and the anonymized graph $\overline{\mathcal{G}}_2$ illustrated in Figure. 3.1c. Here, we have $|V| = 5$. The out-degree information loss of user 0 and user 1

are $DM'^{\overline{\mathcal{G}}_2}_o(0) = \frac{|2-3|}{5} = 0.2$, $DM'^{\overline{\mathcal{G}}_2}_o(1) = \frac{|2-2|}{5} = 0$. Their in-degree information loss are $DM'^{\overline{\mathcal{G}}_2}_i(0) = \frac{|2-0|}{5} = 0.4$, $DM'^{\overline{\mathcal{G}}_2}_i(1) = \frac{|2-2|}{5} = 0$. Then, the out- and in-degree information loss of anonymizing users 0, 1 together is $DM^{\overline{\mathcal{G}}_2}_o(0,1) = \frac{0.2+0}{2} = 0.1$, $DM^{\overline{\mathcal{G}}_2}_i(0,1) = \frac{0.4+0}{2} = 0.2$; $DM^{\overline{\mathcal{G}}_2}(0,1) = 0.5 \times 0.1 + (1-0.5) \times 0.2 = 0.15$.

3.3.3 Algorithms

In the following, we present the algorithms for cluster generation and directed graph generalization.

Clusters generation

Given a directed graph $\mathcal{G}$, storing users' relationships and a positive number k, we aim at generating a set of clusters that have at least k users. Initially, the algorithm assigns each user to a separate cluster. Then, in each iteration, it merges a cluster that has less than k users with another one. This technique allows us to increase the number of users in each cluster. Consequently, the algorithm reduces the number of invalid clusters (i.e., containing less than k users) after each iteration. Our algorithm terminates when all clusters are valid, or it cannot merge clusters anymore. Finally, the algorithm only returns the set of valid clusters.

In general, the more clusters we merge, the more information we lost and the farther the clusters are, the more information the anonymized graphs lose after merging them. Therefore, by merging the closest clusters, our algorithm minimizes the information loss of the anonymized graphs. We calculate the distance between two clusters c_1 and c_2 as follows:

$$D^{\mathcal{G}}(c_1, c_2) = \max\{DM^{\mathcal{G}}(u, v) | \forall u, v, u \in c_1 \wedge v \in c_2\} \tag{3.1}$$

This distance metric measures the maximum information loss between all pairs of users in two clusters. We can minimize the information loss by merging only clusters that are not too far from each other. Thus, we can impose some threshold on the maximum information loss a merge operation might bring. Our algorithm restricts the maximum information loss of merging two clusters to a threshold τ. Then, our algorithm would not merge the clusters whose distances are greater than τ. This parameter gives a good trade-off between the quality and the anonymity of the anonymized graphs. Small values of τ prevent our algorithm from generating the anonymized graphs with high values of k; it, however, allows the algorithm to generate high-quality anonymized graphs.

If clusters have too many users, merging them will require us to add more edges to make their out- and in-degrees identical. Therefore, we limit the maximum number of users in each resulting cluster to ϵ. We will not merge clusters if the resulting one has more than ϵ users.

The algorithm aims at reducing the number of invalid clusters in each iteration by merging an invalid cluster with its nearest one. The naive approach generates all pairs

of one invalid cluster and its closest one and merges the pair whose distance is minimum. The complexity of this step is $O(m \times n)$, where m is the number of invalid clusters and n is the number of all clusters. We improve the performance of this step by only considering ω percentage of the invalid clusters in each iteration. As ω's values go from 0 to 1, the algorithm only considers $\omega \times m$ invalid clusters in each iteration. Thus, we can limit the complexity to $O(\omega \times m \times n)$. By using small values of ω, we can improve the performance of our algorithm, as our experiments will show.

Algorithm 1 Clusters Generation $(\mathcal{G}, k, \tau, \omega, \epsilon)$

Input: Graph $\mathcal{G}(V, E)$; positive number k; parameters τ, ω, and ϵ.
Output: A set of clusters $C^{\overline{\mathcal{G}}}$.

```
1: Let s be the set of clusters containing a single user in V
2: k_s ← min{|c||∀c ∈ s}
3: while k_s < k do
4:     s_next ← find_next_clusters(s, k, τ, ω, ε)
5:     if s_next = ∅ then
6:         break
7:     else
8:         s ← s_next
9:     end if
10:    k_s ← min{|c||∀c ∈ s}
11: end while
12: C^Ḡ ← {c|c ∈ s ∧ |c| ≥ k}
13: return C^Ḡ
```

Algorithm 1 takes as input the directed graph $\mathcal{G}$, a positive number k, the threshold τ, the percentage of invalid clusters ω to be considered at each iteration, and the maximum size of clusters ϵ. At the beginning, it initializes the set of clusters s by inserting each user in V as a distinct cluster (line 1). We denote as k_s the anonymity level of s, as the minimum cardinality among clusters in s (line 2). Then, the algorithm evaluates whether the anonymity level of s is less than k (line 3). If this is the case, Algorithm 1 calls function $find\_next\_clusters()$ (line 4) to find a new set of clusters, i.e., s_{next}. This is computed by merging clusters in s as it will be described in the following. Then, Algorithm 1 checks whether s_{next} is empty (line 5). If this is the case, the algorithm cannot merge clusters anymore. Therefore, it exits from the while loop (line 6). Otherwise, the algorithm assigns s_{next} to s (line 8) and updates k_s (line 10). Then, it goes to the next iteration till s does not contain any cluster that has less than k users or no more merge operations are possible. Finally, the algorithm adds the valid clusters in s to the final set $C^{\overline{\mathcal{G}}}$ (line 12).

Function $find\_next\_clusters()$. Given a set of clusters s, a positive number k, a threshold τ, the percentage of invalid clusters ω, and the maximum size of the merged cluster ϵ, the function finds a new set of clusters by merging two closest clusters, whose

Function 1 $find\_next\_clusters\ (s, k, \tau, \omega, \epsilon)$

1: $\psi_{invalid} \leftarrow \{c | c \in s \wedge |c| < k\}$
2: $\psi_{\omega} \leftarrow \omega \times |\psi_{invalid}|$*randomly selected clusters in* $\psi_{invalid}$
3: $\psi \leftarrow \{(c_1, c_2) | c_1 \in \psi_{\omega} \wedge c_2 \in s \wedge |c_1| + |c_2| \leq \epsilon\}$
4: $score \leftarrow +\infty$
5: **for each** $c'_1, c'_2 \in \psi$ **do**
6: $score' \leftarrow D^{\mathcal{G}}(c'_1, c'_2)$
7: **if** $score' \leq \tau$ **and** $score' < score$ **then**
8: $score \leftarrow score'$
9: $c_{1\_\text{selected}}, c_{2\_\text{selected}} \leftarrow c'_1, c'_2$
10: **end if**
11: **end for**
12: **if** $score \neq +\infty$ **then**
13: $c \leftarrow c_{1\_\text{selected}} \cup c_{2\_\text{selected}}$
14: $s_{next} \leftarrow (s \setminus c_1) \setminus c_2$
15: $s_{next} \leftarrow s_{next} \cup \{c\}$
16: **else**
17: $s_{next} \leftarrow \varnothing$
18: **end if**
19: **return** s_{next}

distance is smaller than or equal to τ. Moreover, the size of the resulting cluster must be less than or equal to ϵ. The function first forms the set of $\omega \times |\psi_{invalid}|$ clusters, randomly selected in the set $\psi_{invalid}$ of invalid clusters (lines 1-2). Then, the function creates the set ψ by finding all pairs consisting of an invalid cluster $c_1 \in \psi_{\omega}$ and a cluster $c_2 \in s$, such that the size of the merged cluster is less than or equal to ϵ (line 3). Then, for each pair c'_1, c'_2 in ψ, it assigns $score'$ based on the distance between c'_1, c'_2, calculated by using the distance metric $D^{\mathcal{G}}(c'_1, c'_2)$ (line 6). Next, it checks whether the distance $score'$ is less than or equal to τ and less than $score$ (which is initialized in line 4 as the largest positive number $+\infty$). If this is the case, the function updates $score$ with $score'$ (lines 7-10) and saves the two clusters with minimum $score$ in $c_{1\_selected}$, $c_{2\_selected}$. The loop terminates when all the pairs in ψ have been checked. Then, the function checks whether at least one pair $c_{1\_selected}$, $c_{2\_selected}$ is found. If this is the case, the new set of clusters s_{next} is created by removing $c_{1\_selected}$, $c_{2\_selected}$ from s and adding the merged cluster $c = c_{1\_selected} \cup c_{2\_selected}$ to s_{next} (lines 13-15). Otherwise, the function sets s_{new} empty (line 17).

In the following example, we illustrate how Algorithm 1 generates clusters.

Example 5. Suppose that we use Algorithm 1 to generate clusters of users in the original directed graph $\mathcal{G}$ (Figure. 3.1a) with the following parameters ω, k, ϵ, τ are equal to 0.5, 2, 4, 0.4, respectively. Initially, Algorithm 1 generates 5 clusters: $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}$. Then, since there are 5 invalid clusters that have less than 2 users, Function $find\_next\_clusters()$ randomly selects $\omega \times 5 = 0.5 \times 5 \approx 3$: $\{0\}, \{2\}, \{4\}$ invalid clusters. The function returns the

pair of clusters: $(\{2\},\{4\})$ since the distance between $\{2\},\{4\}$ is the smallest among all of the considered pairs (i.e., $(\{0\},\{1\})$, $(\{0\},\{2\})$, $(\{0\},\{3\})$, $(\{0\},\{4\})$, $(\{2\},\{1\})$, $(\{2\},\{3\})$, $(\{2\},\{4\})$, $(\{4\},\{1\})$, $(\{4\},\{3\})$) and the algorithm merges them to generate the cluster $\{2,4\}$. In the second iteration, the function randomly chooses 2 invalid clusters: $\{0\},\{3\}$. Next, it can find the pair of two closest clusters: $\{1\},\{3\}$. The algorithm merges the clusters and generates the cluster $\{1,3\}$. The cluster $\{0\}$ cannot be merged to $\{1,3\}$ and $\{2,4\}$ since its distances to these clusters are higher than $\tau = 0.4$. Finally, the final set of clusters $C^{\overline{\mathcal{G}}}$ is $\{\{1,3\}, \{2,4\}\}$.

It is straightforward to show that, given a directed graph and denoting with $C^{\overline{\mathcal{G}}}$ the set of clusters returned by Algorithm 1, executed with a positive number k, all clusters in $C^{\overline{\mathcal{G}}}$ have at least k users. Indeed, let c be an arbitrary cluster in $C^{\overline{\mathcal{G}}}$. Suppose that c has less than k users. As Algorithm 1 removes all clusters that have less than k users from $C^{\overline{\mathcal{G}}}$ (see line 12), $C^{\overline{\mathcal{G}}}$ cannot contain c.

Graph generalization

Given a directed graph $\mathcal{G}$ and the set of clusters $C^{\overline{\mathcal{G}}}$ generated by Algorithm 1, this step generates the anonymized graph $\overline{\mathcal{G}}$ by adding and removing edges in $\mathcal{G}$ such that all users in the same cluster have the same out and in-degrees. Here, all users whose cluster is in $C^{\overline{\mathcal{G}}}$ are included in the set of the anonymized users $\overline{V}$.

Let c be a cluster in $C^{\overline{\mathcal{G}}}$. According to Definition 2, to generalize $\mathcal{G}$, we need to make the anonymized out- and in-degrees of all users in c identical. We denote as $\Im_o^{\mathcal{G}}(c)$ and $\Im_i^{\mathcal{G}}(c)$ the biggest out- and in-degree among all users in c, respectively. We modify the set of edges in the graph to make the out- and in-degree of all users in c identical to the biggest out- and in-degree of users in c. In particular, let u be a user in the cluster c, we denote as $\delta_o(u) = \Im_o^{\mathcal{G}}(c) - d_o(\mathcal{G},u)$ and $\delta_i(u) = \Im_i^{\mathcal{G}}(c) - d_i(\mathcal{G},u)$ the difference between the user u's out- and in-degree in $\mathcal{G}$ and the biggest out- and in-degree of all users in c, respectively. If $\delta_o(u)$ and $\delta_i(u)$ are zero, user u's out- and in-degree are equal to $\Im_o^{\mathcal{G}}(c)$ and $\Im_i^{\mathcal{G}}(c)$.

To anonymize all users u in $\overline{V}$, we aim at adding/removing edges until $\delta_o(u)$ and $\delta_i(u)$ are equal to zero. Let $\Gamma_o = \{u | u \in \overline{V} \wedge \delta_o(u) > 0\}$ and $\Gamma_i = \{u | u \in \overline{V} \wedge \delta_i(u) > 0\}$ be the sets of users whose out- and in-degree are less than the biggest out- and in-degree among those of other users in the same cluster. If Γ_o and Γ_i are empty, the out- and in-degree of all users are the biggest out- and in-degree of their clusters. In the other words, all users in the same clusters have the same out- and in-degree. Therefore, the obtained $\overline{\mathcal{G}}$ satisfies the Paired k-degree [10].

The Directed Graph Anonymization Algorithm (DGA) [10] modifies edges to reduce the number of users in Γ_o and Γ_i until they are empty by using three techniques: *Edge addition*, *Edge switch*, and *Edge extension*. We formalize these techniques as follows:

1. **Edge addition** adds edges (u,v) satisfying the constraint $u \in \Gamma_o \wedge v \in \Gamma_i \wedge u \neq v \wedge (u,v) \notin \overline{E}$. This technique reduces $\delta_o(u)$ and $\delta_i(v)$.

2. **Edge switch** adds edges (u,v'), (u',v) and removes edges (u',v') satisfying the constraint $u \in \Gamma_o \wedge v \in \Gamma_i \wedge u \neq v \wedge (u,v),(u',v') \in \overline{E} \wedge (u',v),(u,v') \notin \overline{E}$. This

technique reduces $\delta_o(u)$ and $\delta_i(v)$ while preserving $I_o^{\overline{\mathcal{G}}}(u')$ and $I_i^{\overline{\mathcal{G}}}(v')$.

3. **Edge extension** adds edges (u', u), (u, v') and removes edges (u', v') satisfying the constraint $u \in \Gamma_o \wedge u \in \Gamma_i \wedge (u', v') \in \overline{E} \wedge (u', u), (u, v') \notin \overline{E}$. This technique reduces $\delta_o(u)$ and $\delta_i(u)$ while keeping $I_o^{\overline{\mathcal{G}}}(u')$ and $I_i^{\overline{\mathcal{G}}}(v')$ the same.

As these techniques require that both Γ_o and Γ_i are not empty, it cannot generate anonymized directed graphs when one of them is empty. To cope with this situation, we propose the Degree Decrement technique to decrease the highest out-degree (in-degree, respectively) of users when Γ_o (Γ_i, respectively) is empty. Our technique removes edges to reduce the number of outgoing or incoming edges. It chooses a cluster where the number of outgoing or incoming edges we need to add is highest among that of other ones. Then, it removes edges of all users whose out- or in-degree is highest among that of the remaining ones in the chosen cluster (see Function 2). After that, we can continue generalizing $\mathcal{G}$ by using the previously explained technique.

Algorithm 2 Graph generalization $(\mathcal{G}, C^{\overline{\mathcal{G}}})$

Input: A directed graph $\mathcal{G}(V, E)$; the set of clusters $C^{\overline{\mathcal{G}}}$.
Output: An anonymized directed graph $\overline{\mathcal{G}}(\overline{V}, \overline{E})$.

1: $\overline{V} \leftarrow \bigcup_{c \in C^{\overline{\mathcal{G}}}} c$
2: $\overline{E} \leftarrow E$
3: Initializes $\overline{\mathcal{G}}(\overline{V}, \overline{E})$
4: Let Γ_o and Γ_i be the set of users who need to increase their out- and in-degrees
5: **while** $|\Gamma_o| > 0$ or $|\Gamma_i| > 0$ **do**
6: $\quad \overline{\mathcal{G}} \leftarrow modify\_edges\_by\_DGA(\overline{\mathcal{G}}, C^{\overline{\mathcal{G}}})$
7: $\quad \overline{\mathcal{G}} \leftarrow decrease\_cluster\_degree(\overline{\mathcal{G}}, C^{\overline{\mathcal{G}}})$
8: $\quad$ Update Γ_o and Γ_i
9: **end while**
10: **return** $\overline{\mathcal{G}}$

Algorithm 2 illustrates our graph generalization approach. First, we calculate the set of users in the anonymized graph $\overline{\mathcal{G}}$ as the union of all clusters in $C^{\overline{\mathcal{G}}}$ (line 1). Also, we initialize the set of anonymized edges $\overline{E}$ with the set of original ones E (line 2). Then, we initialize the anonymized graph $\overline{\mathcal{G}}$ with $\overline{E}$ and $\overline{V}$ (line 3). The algorithm keeps generalizing by calling DGA [10] (line 6) and the Degree Decrement technique (line 7). It terminates when $|\Gamma_o| = 0$ and $|\Gamma_i| = 0$. Finally, the algorithm returns the anonymized graph $\overline{\mathcal{G}}$ (line 10).

Function *decrease_cluster_degree*(). The function starts by initializing δ_o and δ_i for all users in $\overline{V}$ (line 1). Then, it initializes Δ_o and Δ_i for all clusters c in $C^{\overline{\mathcal{G}}}$, where $\Delta_o(c)$ and $\Delta_i(c)$ are the number of outgoing and incoming edges we need to add to anonymize

all users in cluster c (lines 2-5). Next, it finds the clusters whose Δ_o and Δ_i are highest: c_o and c_i (lines 6-7). If $\Delta_o(c_o)$ is higher than $\Delta_i(c_i)$, we will decrease the highest out-degree of the cluster c_o by one (lines 8-15). To decrease the highest out-degree of the cluster c_o, we find the highest out-degree of all users in c_o: $\Im_o^{\overline{\mathcal{G}}}(c_0)$. Then, for every user u in c_o, if the user u's out-degree (i.e., $d_o(\overline{\mathcal{G}}, u)$) is equal to $\Im_o^{\overline{\mathcal{G}}}(c_0)$, we randomly remove an outgoing edge of u to decrease his/her out-degree (lines 10-15). Similarly, we use the same approach to decrease the highest in-degree of the cluster c_i by one (lines 17-25). Finally, the function returns the anonymized graph $\overline{\mathcal{G}}$.

Function 2 *decrease_cluster_degree* $(\overline{\mathcal{G}}(\overline{V}, \overline{E}), C^{\overline{\mathcal{G}}})$

1: Initializes δ_o and δ_i for all users in $\overline{V}$
2: **for** $c \in C^{\overline{\mathcal{G}}}$ **do**
3: $\quad \Delta_o(c) \leftarrow \sum_u^c \delta_o(u)$
4: $\quad \Delta_i(c) \leftarrow \sum_u^c \delta_i(u)$
5: **end for**
6: $c_o \leftarrow \arg\max_c^{C^{\overline{\mathcal{G}}}} \Delta_o(c)$
7: $c_i \leftarrow \arg\max_c^{C^{\overline{\mathcal{G}}}} \Delta_i(c)$
8: **if** $\Delta_o(c_o) > \Delta_i(c_i)$ **then**
9: $\quad \Im_o^{\overline{\mathcal{G}}}(c_o) \leftarrow \max_u^{c_o} d_o(\overline{\mathcal{G}}, u)$
10: $\quad$ **for** $u \in c_o$ **do**
11: $\quad\quad$ **if** $d_o(\overline{\mathcal{G}}, u) = \Im_o^{\overline{\mathcal{G}}}(c_o)$ **then**
12: $\quad\quad\quad v \leftarrow$ *randomly find* $v : (u, v) \in \overline{E}$
13: $\quad\quad\quad \overline{E} \leftarrow \overline{E} \setminus \{(u, v)\}$
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: **else**
17: $\quad$ **if** $\Delta_i(c_i) > 0$ **then**
18: $\quad\quad \Im_i^{\overline{\mathcal{G}}}(c_i) \leftarrow \max_u^{c_i} d_i(\overline{\mathcal{G}}, u)$
19: $\quad\quad$ **for** $u \in c_i$ **do**
20: $\quad\quad\quad$ **if** $d_i(\overline{\mathcal{G}}, u) = \Im_i^{\overline{\mathcal{G}}}(c_i)$ **then**
21: $\quad\quad\quad\quad v \leftarrow$ *randomly find* $v : (v, u) \in \overline{E}$
22: $\quad\quad\quad\quad \overline{E} \leftarrow \overline{E} \setminus \{(v, u)\}$
23: $\quad\quad\quad$ **end if**
24: $\quad\quad$ **end for**
25: $\quad$ **end if**
26: **end if**
27: **return** $\overline{\mathcal{G}}$

We illustrate Algorithm 2 in the following example.

Example 6. Suppose that we use Algorithm 2 to generate anonymized graph from the generated clusters in Example 5, i.e., $\{1, 3\}, \{2, 4\}$. Since $\Gamma_o = \{3, 4\}$ and $\Gamma_i = \varnothing$, DGA

cannot add, switch, or extend edges. Then, Function *decrease_cluster_degree*() calculates $\Delta_o(\{1,3\}) = 1$, $\Delta_o(\{2,4\}) = 1$, $\Delta_i(\{1,3\}) = 0$, $\Delta_o(\{2,4\}) = 0$ and $c_o = \{1,3\}$ and $c_i = \{2,4\}$. As $\Delta_o(\{1,3\}) > \Delta_i(\{2,4\})$, the function removes two edges $(3,2)$ and $(1,3)$. Finally, the algorithm can use DGA to add two fake edges $(3,2)$ and $(4,3)$ and the out-/in-degrees of users $1,3$ are identical. Similarly, those of user 2 are equal to those of user 4.

Algorithm 2 terminates when both set Γ_o and Γ_i are empty. In this case, all users in the same cluster have the same out- and in-degree.

Theorem 1. Let $\mathcal{G}$ be a directed graph, $\overline{\mathcal{G}}$ be its anonymized version generated by Algorithm 2, and $C^{\overline{\mathcal{G}}}$ be the set of clusters. If $|\Gamma_o| = 0$ and $|\Gamma_i| = 0$, then, for every cluster c in $C^{\overline{\mathcal{G}}}$, for every user u, v in c, $d_o(\overline{\mathcal{G}}, u) = d_o(\overline{\mathcal{G}}, v)$ and $d_i(\overline{\mathcal{G}}, u) = d_i(\overline{\mathcal{G}}, v)$.

Proof. Let c be an arbitrary cluster in $C^{\overline{\mathcal{G}}}$ and u, v be two arbitrary users in c. Suppose $d_o(\overline{\mathcal{G}}, u) \neq d_o(\overline{\mathcal{G}}, v)$ or $d_i(\overline{\mathcal{G}}, u) \neq d_i(\overline{\mathcal{G}}, v)$. Then, $d_o(\overline{\mathcal{G}}, u) < \Im_o^{\overline{\mathcal{G}}}(c)$ or $d_o(\overline{\mathcal{G}}, v) \neq \Im_o^{\overline{\mathcal{G}}}(v)$ or $d_i(\overline{\mathcal{G}}, u) \neq \Im_i^{\overline{\mathcal{G}}}(c)$ or $d_i(\overline{\mathcal{G}}, v) \neq \Im_i^{\overline{\mathcal{G}}}(v)$. Thus, $\delta_o(u) > 0$ or $\delta_o(v) > 0$ or $\delta_i(u) > 0$ or $\delta_i(v) > 0$. Then, $|\Gamma_o| > 0$ or $|\Gamma_i| > 0$. As c, u, and v were arbitrary, we can conclude that if $|\Gamma_o| = 0$ and $|\Gamma_i| = 0$ then for every cluster c in $C^{\overline{\mathcal{G}}}$, for every user u, v in c, $d_o(\overline{\mathcal{G}}, u) = d_o(\overline{\mathcal{G}}, v)$ and $d_i(\overline{\mathcal{G}}, u) = d_i(\overline{\mathcal{G}}, v)$. □

Theorem 2. Let $\mathcal{G}$ be a directed graph, $\overline{\mathcal{G}}$ be its anonymized version generated by Algorithm 2, and $C^{\overline{\mathcal{G}}}$ be the set of clusters. For all clusters c in $C^{\overline{\mathcal{G}}}$, for all users u, v in c, $I_d(\overline{\mathcal{G}}, u) = I_d(\overline{\mathcal{G}}, v)$.

Proof. Let c be an arbitrary cluster in $C^{\overline{\mathcal{G}}}$ and u, v be two arbitrary users in c. When Algorithm 2 terminates, $|\Gamma_o| = 0$ and $|\Gamma_i| = 0$ (see Algorithm 2, line 5). According to Theorem 1, $d_o(\overline{\mathcal{G}}, u) = d_o(\overline{\mathcal{G}}, v)$ and $d_i(\overline{\mathcal{G}}, u) = d_i(\overline{\mathcal{G}}, v)$. Then, according to Definition 1, $I_d(\overline{\mathcal{G}}, u) = I_d(\overline{\mathcal{G}}, v)$. As c, u, and v are arbitrary, we can conclude that for all clusters c in $C^{\overline{\mathcal{G}}}$, for all users u, v in c, $I_d(\overline{\mathcal{G}}, u) = I_d(\overline{\mathcal{G}}, v)$. □

Theorem 3. Let $\mathcal{G}(V, E)$ be a directed graph, $\overline{\mathcal{G}}(\overline{V}, \overline{E})$ be its anonymized version created by Algorithm 2, and k be an integer number. $\overline{\mathcal{G}}$ satisfies the Paired k-degree [10] and K-In&Out-Degree Anonymity [60].

Proof. Let $C^{\overline{\mathcal{G}}}$ be the set of clusters generated by Algorithm 1 and $C^{\overline{\mathcal{G}}}(u)$ be the cluster in $C^{\overline{\mathcal{G}}}$ containing the user u. For every user u in $\overline{V}$, $|C^{\overline{\mathcal{G}}}(u)| \geq k$. Moreover, as $\overline{\mathcal{G}}$ is generated by Algorithm 2, according to Theorem 2, for every user u, v in $\overline{V}$, if u and v are in the same cluster $C^{\overline{\mathcal{G}}}(u)$ then $I_d(\overline{\mathcal{G}}, u) = I_d(\overline{\mathcal{G}}, v)$. Then, for every user u in $\overline{V}$, there are at least $k-1$ other users having the same out- and in-degrees, i.e., I_d, to u. Therefore, we can conclude that $\overline{\mathcal{G}}$ satisfies the Paired k-degree. $\overline{\mathcal{G}}$ also satisfies K-In&Out-Degree Anonymity as its requirements are identical to those of the Paired k-degree. □

3.4 Experiments

In this section, we evaluate the quality of the anonymized directed graphs generated by our algorithm by varying the requested parameters (k, ω, and τ) and compare our algorithm

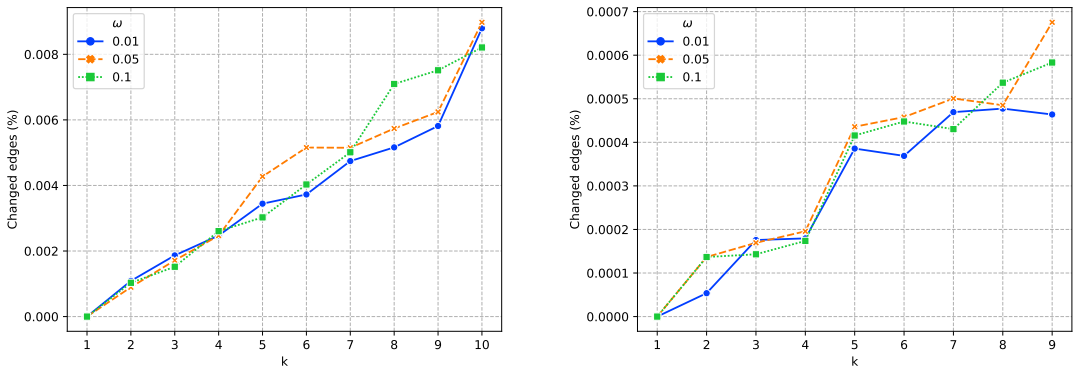
CDGA with previous proposals, namely DGA [10], and DSNDG-KIODA [60]. All experiments in this thesis were conducted on a Debian GNU/Linux server with 64 dual-core 2.00-GHz Intel(R) Xeon(R) processors and 128-GB RAM.

3.4.1 Datasets

We use three real-life networks to evaluate our algorithm: (1) *Email-temp* (986 users) [43], (2) *Bitcoin Alpha* (3,783 users) [30], and (3) *DBLP* (12,591 users) [32]. Appendix B summarizes properties of these datasets.

3.4.2 Tuning CDGA

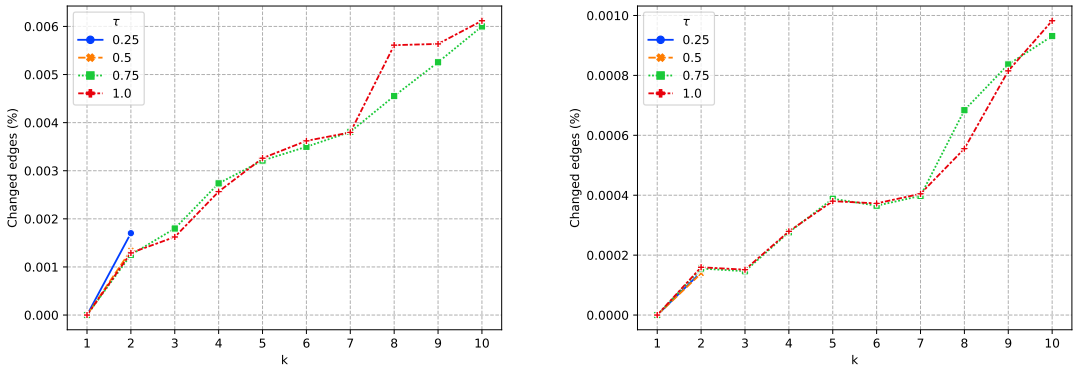
This experiment evaluates the impact of our parameters k, ω, and τ on the quality of the anonymized graphs generated by our algorithm. We denote as *changed edges* $= \frac{|\overline{E}\backslash E|+|E\backslash\overline{E}|}{|E|\times|E|}$ the number of edges that are added or removed to generate the anonymized graph. We normalize it by $|E| \times |E|$, as it is the maximum number of added and removed edges. The more changes we make, the lower the quality the anonymized graph is.



(a) Email-temp (b) Bitcoin Alpha

Figure 3.2: Changed edges by varying k and ω.

Effects of ω. CDGA merges an invalid cluster, which has less than k users with the nearest one. We use ω to control how many invalid clusters CDGA checks in each iteration. Figure. 3.2 illustrates the ratio of changes made to the original graph to generate its anonymized version on *Email-temp* (Figure. 3.2a) and *Bitcoin Alpha* (Figure. 3.2b). We assign τ to 1.0. Here, in all datasets, we achieve good quality anonymized graphs even with $\omega = 0.01$. Furthermore, the differences in the ratio of changes in the anonymized graphs by varying ω is also small. The maximum differences is about 0.001 in *Email-temp* and 0.0002 in *Bitcoin Alpha*. Therefore, we do not need to specify a high value of ω to obtain good quality anonymized graphs.



(a) Email-temp (b) Bitcoin Alpha

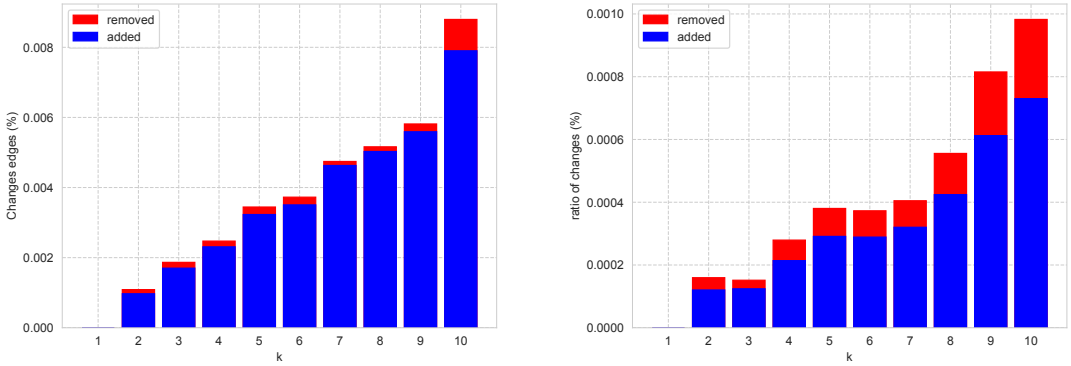
Figure 3.3: Changed edges by varying k and τ.

Effects of τ. We use τ to specify the maximum distance between the chosen clusters. Figure. 3.3 shows the ratio of changes made to the original graph to generate its anonymized version on *Email-temp* (Figure. 3.3a) and *Bitcoin Alpha* (Figure. 3.3b). If the distances of all clusters are higher than τ, the algorithm will not merge clusters any more. τ allows us to control the trade-off between the quality and the anonymity of the anonymized graphs. In both datasets, with $\tau = 0.25$ and $\tau = 0.5$, we can only anonymize graphs satisfying the Paired 2-degree. With $\tau = 0.75$ and $\tau = 1.0$, we can obtain the anonymized graphs with k from 2 to 10. If we want CDGA to always generate the anonymized graphs satisfying the Paired k-degree [10] and K-In&Out-Degree Anonymity [60], we should assign τ with 1.0. Otherwise, we can keep decreasing τ until the resulting graphs reach the expected quality while ensuring a certain degree of anonymity.

Effects of k. As showed in the previous experiments (Figure. 3.2-3.3), the quality of the anonymized graph decreases by increasing k.

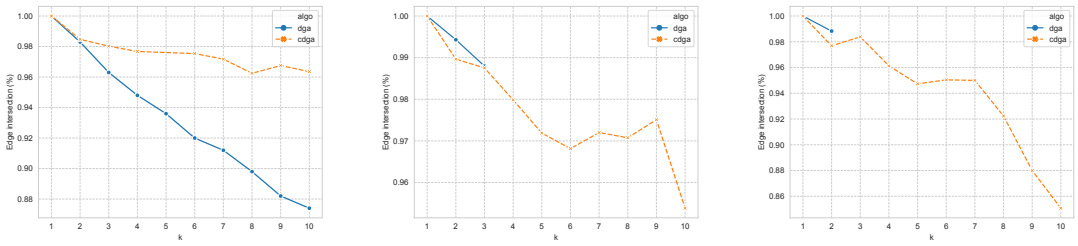
3.4.3 Evaluating the Degree Decrement

In this experiment, we further evaluate the effectiveness of the Degree Decrement technique. Figure. 3.4 illustrates the ratio of edges addition and removal to anonymize graphs on *Email-temp* (Figure. 3.4a) and *Bitcoin Alpha* (Figure. 3.4b), respectively. We only need to remove a small number of edges to generate anonymized graphs satisfying the Paired k-degree. More precisely, we only need to modify a small number of edges (0.006 of edges in *Email-temp* and 0.001 of edges in *Bitcoin Alpha*) to generate the anonymized graphs. Moreover, the number of removed edges is smaller than the number of added edges in all datasets.



(a) Email-temp (b) Bitcoin Alpha

Figure 3.4: Details on the changes of graphs on varying k.



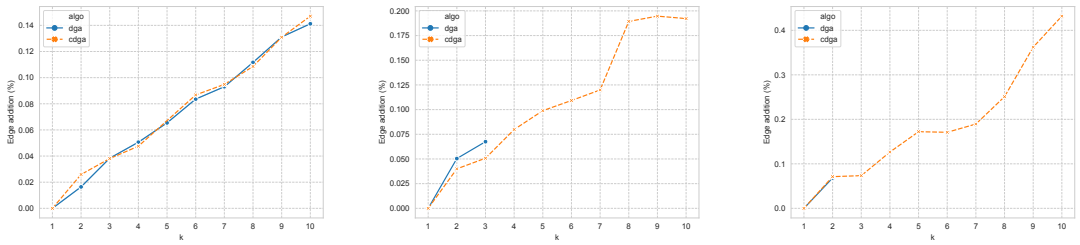
(a) DBLP (b) Email-temp (c) Bitcoin Alpha

Figure 3.5: Intersection Edges of DGA and CDGA.

3.4.4 Comparative Analysis

In this experiment, we compare the quality of the anonymized graphs generated by our algorithm (CDGA) and those generated by DGA [10] and DSNDG-KIODA [60].

To compare with DGA [10], we use metrics that Casa et al. [10] used to evaluate DGA: the added edges and the intersection edges. The added edges are the number of edges added to the original graphs to generate their anonymized versions. The intersection edges are edges that exist in both the original and anonymized graphs. Casa et al. [10] normalize both of them by the number of edges in the original graphs. Figure. 3.5 and 3.6 illustrate the intersection and added edges on the three datasets, namely *DBLP* (Figure. 3.6a), *Email-temp* (Figure. 3.6b), and *Bitcoin Alpha* (Figure. 3.6c), respectively. Only the results of DGA on *DBLP* are obtained from their paper [10]. We used their software to obtain the results on the remaining datasets: *Email-temp* and *Bitcoin Alpha*. Here, the number of edges added by CDGA is similar to that of DGA on all datasets. However, in *DBLP*, DGA needs to remove more edges. When $k = 10$, CDGA preserves more than 0.08 of edges



(a) DBLP (b) Email-temp (c) Bitcoin Alpha

Figure 3.6: Edges Addition of DGA and CDGA.

comparing to DGA. These results show that DGA does not always generate the anonymized graphs satisfying the Paired k-degree. The maximum k that DGA can generate is 3 on *Email-temp* and 2 on *Bitcoin Alpha*.

To compare the results of CDGA with DSNDG-KIODA [60], we use the average clustering coefficient (ACC) differences between the anonymized and the original graphs as it is used to evaluate the quality of the resulting graphs generated by DSNDG-KIODA in [60]. Here, we denote as $ACC(G)$ the average clustering coefficient of a directed graph G. Then, the average clustering coefficient differences between the anonymized graph $\overline{G}$ and its original version G is $ACC = \frac{|ACC(\overline{G}) - ACC(G)|}{ACC(G)}$. We obtain the results of DSNDG-KIODA from their paper [60].

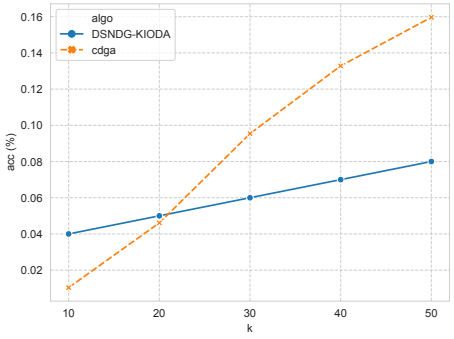


Figure 3.7: ACC of CDGA and DSNDG-KIODA.

Figure. 3.7 illustrates the quality of the anonymized graphs generated by CDGA and DSNDG-KIODA on *Email-temp*, respectively. Here, we only evaluate *Email-temp* as we do not have access to their software. When $k = 10$ and 20, our ACC is less than theirs 0.03 at $k = 10$ and 0.005 at $k = 20$. Although our results are worse than theirs 0.035 at $k = 30$, 0.06 at $k = 40$, and 0.08 at $k = 50$, DSNDG-KIODA requires adding both fake edges and nodes to generate the anonymized graphs. This requirement makes their anonymized

graphs useless to the modern machine learning applications (e.g., spam detection) as these applications need to be trained with real-life nodes. Here, CDGA only needs to modify edges of the graphs to anonymize them. Therefore, data providers should use CDGA to anonymize their graphs if they want to preserve all of the original nodes.

Chapter 4

Anonymization of Knowledge Graphs

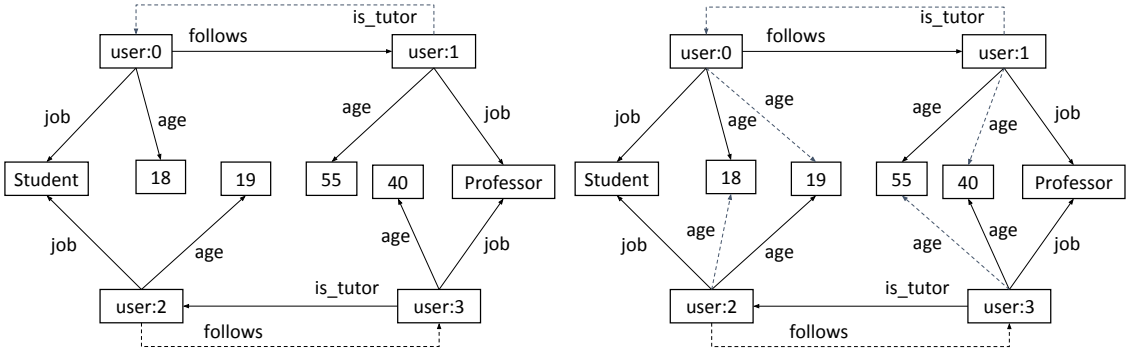
4.1 Introduction

The Paired k-degree [10] and K-In&Out-Degree Anonymity [60] described in Section 2.3 protect users' identities in anonymized directed graphs. Unfortunately, these models are insufficient to protect users in KGs as adversaries can exploit both users' attributes and different types of relationships [45, 46] as the following example shows.

Example 7. Let $\overline{G}_1$ in Figure. 4.1a be a KG anonymized according to the Paired k-degree. For all types of relationships (e.g., $follows$, is_tutor), *user:0* and *user:1*'s out- and in-degrees are equal to those of *user:2* and *user:3*, respectively. Therefore, $\overline{G}_1$ satisfies the Paired 2-degree. However, as the values of nodes (e.g., *18*, *19*, *Student*) representing attributes' values are not replaced by random numbers, adversaries can re-identify users by their attributes.

In this chapter, we present k-Attribute Degree (k-ad). Different from previous models, k-ad requires that, for each user, his/her attributes and the out-/in-degree of any type of his/her relationships are indistinguishable from those of $k-1$ other users. Therefore, an adversary cannot re-identify any user in k-ad KGs with a confidence higher than $\frac{1}{k}$. Figure. 4.1b illustrates the 2-ad version of $\overline{G}_1$. As we add fake edges to make both the attributes (e.g., *age*, *job*) and the relationships out-/in-degree (e.g., $follows$, is_tutor) of *user:0* and *user:1* equal to those of *user:2* and *user:3*, respectively, all of them cannot be re-identified with a confidence higher than $\frac{1}{2}$.

The modifications performed to generate k-ad KGs decrease the amount of information the anonymized KGs have. To cope with this, we present the Attribute and Degree Information Loss (ADM) to measure the information loss of anonymized KGs. Moreover, as k-ad requires modifying values of users' attributes, we have to consider whether the new values are truthful or not. As an example, we cannot associate to *age* attribute the value *18* if user's *job* is $Professor$, because an adversary can easily detect these untruthful associations and try to remove them to infer real values. For instance, the adversary can figure



(a) Paired 2-degree Knowledge Graph $\overline{G}_1$ (b) 2-ad Knowledge Graph $\overline{G}_2$

Figure 4.1: Knowledge graphs satisfying Paired k-degree and k-ad.

out that either the user' *age* or *job* is fake. As such, we have to measure the truthfulness of attributes' values in anonymized KGs. To estimate this, we propose to consider how truthful are associations between attributes' values. Indeed, according to the Closed-World Assumption [29], an association between two attributes' values is untruthful if the original KG does not contain any user that have these values at the same time.[1]

Then, by using this information, we can measure the truthfulness of the values of users' attributes, through the Attribute Truthfulness Information Loss (ATM). Moreover, since verifying the existence of an association in KGs is time consuming, we use bilinear functions to learn an indicator deciding whether an association is truthful. We exploit this indicator to measure how many untruthful associations a user has after anonymizing his/her attributes. The more untruthful associations a user has, the higher his/her ATM is. Then, by minimizing the number of these untruthful associations, we can maximize the truthfulness of users' attributes.

We present a Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA) allowing data providers to use any type of clustering algorithm (e.g., k-means [14]) to anonymize KGs. First, we turn users into data points in Euclidean space such that their information loss is almost equal to the Euclidean distance of their corresponding points. By minimizing the distances between points in the same cluster, we minimize the information loss of users in these clusters. Also, we present strategies to make the sizes of the generated clusters in between k and $2 \times k - 1$. Finally, we extend the Directed Graph Generalization Algorithm (DGG) (Algorithm 2) in Chapter 3 to make the attributes' values and out-/in-degrees of users in the same cluster identical. We conduct experiments on five real-life datasets to evaluate the quality of anonymized KGs generated by our algorithm and compare it with our anonymization algorithm for directed graphs (Chapter 3) and the previous work [10].

The remaining sections of this chapter are organized as follows. Section 4.2 illustrates

[1]In case the truthful associations are absent in the original KG, our work can be extended easily by checking whether these associations exist in completed KGs containing more truthful data.

the adversaries' attack model. We introduce our information loss metrics in Section 4.3 and our algorithms in Section 4.4. Finally, in Section 4.5, we evaluate the efficiency of our approach.

4.2 Anonymizing Knowledge Graphs

We model a knowledge graph (KG) as a graph $G(V, E, R)$, where V is the set of nodes, E is the set of edges connecting these nodes, and R is the set of relationship types. Figure. 4.1 shows an example of using KGs to model users' data. Since a node can be used to represent a user or the value of an attribute, there are two subsets of nodes: the set $V^U \subseteq V$, modelling users (e.g., *user:0*, *user:1*), and the set $V^A \subseteq V$, representing attributes' values (e.g., *18*, *Student*). Thus, $V = V^U \cup V^A$. Different from previous work [10, 60], users can have more than one value for each attribute.

Relationship types in R are categorized into two subsets: user-to-user relationship types R^{UU}, representing users' relationships (e.g., $follows$), and user-to-attribute relationship types R^{UA}, modelling users' attributes (e.g., *age*, *gender*, *job*). Then, $R = R^{UU} \cup R^{UA}$. Each edge $e \in E$ is defined as a triple (u, r, v), where $u, v \in V$ and $r \in R$. We denote with $E^{UA} \subseteq E$ those $e = (u, r_a, v_a)$, such that $r_a \in R^{UA}$. Similarly, we denote with E^{UU} those $e = (u, r_u, v_u)$, such that $r_u \in R^{UU}$. Appendix A illustrates notations used in this chapter.

4.2.1 Adversary Background Knowledge

Let $G(V, E, R)$ be a KG and $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be an anonymized version of G created by modifying V, E, and R. Let u be a user in $\overline{V}^U$. As KGs contain values of many attributes, adversaries can re-identify u by using all of his/her attributes' values in $\overline{G}$, denoted as $I_a(\overline{G}, u) = \{(r_a, v_a) | (u, r_a, v_a) \in \overline{E}^{UA}\}$. Additionally, adversaries can also re-identify u if they know his/her out- and in-degree. However, as $\overline{G}$ represents many types of relationships, adversaries can exploit the out- and in-degree from any type of these relationships to re-identify u. More precisely, given a relationship type $r_u \in \overline{R}^{UU}$, we denote with $d_o(\overline{G}, r_u, u) = |\{(u, r_u, v_u) \in \overline{E}^{UU}\}|$ and $d_i(\overline{G}, r_u, u) = |\{(v_u, r_u, u) \in \overline{E}^{UU}\}|$ the u's out- and in-degree of the relationship type r_u, respectively. Then, the out- and in-degree from all relationship types of u in $\overline{G}$ are $I_o(\overline{G}, u) = \{(r_u, d_o(\overline{G}, r_u, u)) | r_u \in \overline{R}^{UU}\}$ and $I_i(\overline{G}, u) = \{(r_u, d_i(\overline{G}, r_u, u)) | r_u \in \overline{R}^{UU}\}$, respectively.

An adversary who has access to $\overline{G}$ can re-identify a user $u \in \overline{V}^U$ by combining his background knowledge that he/she knows about u and the extracted attributes' values and out-/in-degrees of u in $\overline{G}$. More formally, we define the background knowledge of the adversary as follows:

Definition 6 (Adversary Knowledge on a KG)**.** *Let $\overline{G}$ be an anonymized KG. The knowledge that an adversary can use to re-identify a user $u \in \overline{G}$ contains:*

- *The background knowledge that the adversary knows about u: $\mathcal{BK}(u)$.*

- *Attributes' values and out-/in-degrees of relationships of user u in $\overline{G}$:* $I(\overline{G}, u) = (I_a(\overline{G}, u), I_o(\overline{G}, u), I_i(\overline{G}, u))$.

Example 8. In Figure. 4.1b, the attributes, out-, and in-degrees about user *user:0* that can be extracted from the anonymized KG $\overline{G}_2$ is $I_a(\overline{G}_2$, *user:0*$) = \{(age, 18), (age, 19), (job, Student)\}$; $I_o(\overline{G}_2$, *user:0*$) = \{(follows, 1), (is\_tutor, 0)\}$; $I_i(\overline{G}_2$, *user:0*$) = \{(follows, 0), (is\_tutor, 1)\}$. Thus, the knowledge about *user:0* that an adversary can extract from $\overline{G}_2$ is $I(\overline{G}_2$, *user:0*$) = (\{(age, 18), (age, 19), (job, Student)\}, \{(follows, 1), (is\_tutor, 0)\}, \{(follows, 0), (is\_tutor, 1)\})$.

An adversary can use the above knowledge to re-identify his/her victim u by using an attacking mechanism $\mathcal{T}_{\overline{G}}$ to identify if a user $v \in \overline{G}$ is the representation of u. We formally define the attacking mechanism as follows:

Definition 7 (Attacking Mechanism to a KG)**.** *Let $\overline{G}$ be an anonymized KG, u be the user that an adversary wants to re-identify, and v be an arbitrary user in $\overline{G}$. The attacking mechanism $\mathcal{T}_{\overline{G}}$ is represented as*

$$\mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, v)) = \begin{cases} 1, \textit{if } u, v \textit{ is the same user.} \\ 0, \textit{otherwise.} \end{cases}$$

We define the risk of re-identifying any user in $\overline{G}$ under the attacking mechanism $\mathcal{T}_{\overline{G}}$ as follows:

Definition 8 (Privacy Disclosure Risk in a KG)**.** *Let $\overline{G}$ be an anonymized KG that a data provider has published, and u be a user that an adversary want to re-identify. The Privacy Disclosure Risk of u is the confidence that the adversary can re-identify u by using his/her background knowledge about u and the attacking mechanism $\mathcal{T}_{\overline{G}}$:*

$$risk(\mathcal{T}_{\overline{G}}, \overline{G}, u) = \frac{1}{\sum\limits_{v \in \overline{V}^U} \mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, v))}$$

4.2.2 Anonymity of Knowledge Graphs

We present k-Attribute Degree (k-ad), a privacy protection model to protect users in anonymized KGs. In particular, let $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be an anonymized version of G. For each user u in $\overline{V}^U$, we have to ensure that his attributes' values, out-, and in-degrees extracted from $\overline{G}$ are indistinguishable from those of at least $k-1$ other ones in $\overline{V}^U$. More formally, k-Attribute Degree (k-ad) is defined as follows:

Definition 9 (k-Attribute Degree)**.** *Let $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be an anonymized KG. $\overline{G}$ satisfies k-Attribute Degree (k-ad), if and only if, for every user u in $\overline{V}^U$, there exists a set of users, denoted $\mathcal{C}(\overline{G}, u)$, such that $\mathcal{C}(\overline{G}, u) = \{v \in \overline{V}^U | I_a(\overline{G}, u) = I_a(\overline{G}, v) \wedge I_o(\overline{G}, u) = I_o(\overline{G}, v) \wedge I_i(\overline{G}, u) = I_i(\overline{G}, v)\}$ and $|\mathcal{C}(\overline{G}, u)| \geq k$.*

If an anonymized KG $\overline{G}$ satisfies k-ad, the Privacy Disclosure Risk of all users in $\overline{G}$ is less than or equal to $\frac{1}{k}$.

Theorem 4. Let $\overline{G}$ be an anonymized KG that an adversary has access to and $\mathcal{T}_{\overline{G}}$ be an attacking mechanism that he/she uses to re-identify users in $\overline{G}$. If $\overline{G}$ satisfies k-ad, for every user $u \in \overline{G}$, $risk(\mathcal{T}_{\overline{G}}, \overline{G}, u) \leq \frac{1}{k}$.

Proof. Suppose $\overline{G}$ satisfies k-ad. Let u be an arbitrary user in $\overline{G}$. According to Definition 9, there is a set $\mathcal{C}(\overline{G}, u) = \{v \in \overline{V}^U | I(\overline{G}, u) = I(\overline{G}, v)\}$ and $|\mathcal{C}(\overline{G}, u)| \geq k$. Then, for every user $v \in \mathcal{C}(\overline{G}, u)$, $\mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, v)) = 1$. Thus, $risk(\mathcal{T}_{\overline{G}}, \overline{G}, u) = \frac{1}{\sum_{v \in \overline{V}^U} \mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, u))} \leq \frac{1}{k}$. Since u is arbitrary, we can conclude that if $\overline{G}$ satisfies k-ad, for every user $u \in \overline{V}^U$, $risk(\mathcal{T}_{\overline{G}}, \overline{G}, u) \leq \frac{1}{k}$. □

As the values of all attributes and the out- and-in degree of all relationship types of every user in the k-ad anonymized KGs are indistinguishable from those of $k-1$ other users, it can be easily proved that these KGs also satisfy the previous protection models: k-anonymity [26], the Paired k-degree [10], and K-In&Out-Degree Anonymity [60].

Theorem 5. Let $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be an anonymized KG. If $\overline{G}$ satisfies k-ad, $\overline{G}$ satisfies k-anonymity, the Paired k-degree, and K-In&Out-Degree Anonymity.

Proof. Suppose $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ satisfies k-ad. Then, according to Definition 6, for every user $u \in \overline{V}^U$, $|\mathcal{C}(\overline{G}, u)| \geq k$. Suppose that $\overline{G}$ does not satisfy k-anonymity, the Paired k-degree, or K-In&Out-Degree Anonymity. Then, there is at least one user $u \in \overline{V}^U$, such that there are less than $k-1$ other users whose values of all attributes (i.e., I_a), out-, and in-degree of all types of relationships (i.e., I_o and I_i, respectively) are identical to those of u. Thus, $|\mathcal{C}(\overline{G}, u)| < k$. But, this contradicts the fact that $|\mathcal{C}(\overline{G}, u)| \geq k$ for every user $u \in \overline{V}^U$. Thus, we can conclude that if $\overline{G}$ satisfies k-ad, it also satisfies k-anonymity, the Paired k-degree, and K-In&Out-Degree Anonymity. □

4.3 Information Loss Metrics

In what follows, we describe the information loss metrics we use to evaluate the quality of anonymized KGs.

4.3.1 Attribute and Degree Information Loss

As KGs contain both users' attributes and relationships, our information loss metrics consider the loss on all of these types of information. We present the Attribute Information Loss Metric (AM') to evaluate the loss of attributes' information on a user in anonymized KGs. In what follows, we denote with $G(V, E, R)$ and with $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ the original KG and one of its anonymized version. Let u be a user in G and r_a be an attribute in R^{UA}. We denote with $I_a(G, r_a, u) = \{v_a | (u, r_a, v_a) \in E^{UA}\}$ and $I_a(\overline{G}, r_a, u) = \{v_a | (u, r_a, v_a) \in \overline{E}^{UA}\}$ the values of attribute r_a of user u in G and $\overline{G}$, respectively. If r_a is a categorical attribute, AM' measures the differences of values of u's r_a attribute in $\overline{G}$ and G:

$I_a(\overline{G}, r_a, u) \setminus I_a(G, r_a, u)$. In contrast, if r_a is a numerical attribute, the value of u's attribute r_a in $\overline{G}$ and G can be represented as a range $[\min I_a(\overline{G}, r_a, u), \max I_a(\overline{G}, r_a, u)]$ and $[\min I_a(G, r_a, u), \max I_a(G, r_a, u)]$, respectively. Then, AM' measures the changes of these ranges: $|\min I_a(\overline{G}, r_a, u) - \min I_a(G, r_a, u)| + |\max I_a(\overline{G}, r_a, u) - \max I_a(G, r_a, u)|$. Therefore, we define AM' as follows:

Definition 10 (AM')**.** *Let u be a user in G. The Attribute Information Loss Metric (AM') of anonymizing user u in $\overline{G}$ is:*

$$AM_c'^{\overline{G}}(u, r_a) = \frac{|I_a(\overline{G}, r_a, u) \setminus I_a(G, r_a, u)|}{|dom_a(G, r_a) \setminus I_a(G, r_a, u)| + 1}$$

$$AM_n'^{\overline{G}}(u, r_a) = \frac{|\min I_a(\overline{G}, r_a, u) - \min I_a(G, r_a, u)| + |\max I_a(\overline{G}, r_a, u) - \max I_a(G, r_a, u)|}{|\min dom_a(G, r_a) - \min I_a(G, r_a, u)| + |\max dom_a(G, r_a) - \max I_a(G, r_a, u)| + 1}$$

$$AM'^{\overline{G}}(u) = \frac{1}{|R^{UA}|} \times \sum_{r_a}^{R^{UA}} \begin{cases} AM_c'^{\overline{G}}(u, r_a), \textit{if } r_a \textit{ is a categorical attribute} \\ AM_n'^{\overline{G}}(u, r_a), \textit{if } r_a \textit{ is a numerical attribute} \end{cases}$$

where $dom_a(G, r_a) = \{v_a | (u, r_a, v_a) \in E^{UA}\}$.

Then, we exploit this metric to evaluate the information loss of making identical the attributes' values of two users by introducing this further definition:

Definition 11 (AM)**.** *Let u, v be two users in G. The Attribute Information Loss (AM) of making u and v having the same values for all of their attributes in $\overline{G}$ is:*

$$AM^{\overline{G}}(u, v) = \frac{AM'^{\overline{G}}(u) + AM'^{\overline{G}}(v)}{2}$$

Example 9. Let $\overline{G}_2$ be the KG showed in Fig.4.1b and G be its original version. $R^{UA} = \{age, job\}$, $dom_a(G, age) = \{18, 19, 40, 50\}$ and $dom_a(G, job) = \{Student, Professor\}$. Let assume we make the values of all attributes of *user:0* and *user:2* identical, $I_a(\overline{G}_2, age, user{:}0) = I_a(\overline{G}_2, age, user{:}2) = \{18, 19\}$, and $I_a(\overline{G}_2, job, user{:}0) = I_a(\overline{G}_2, job, user{:}2) = \{Student\}$. The information loss of anonymizing *user:0*'s *job* and *age* are $AM_c'^{\overline{G}_2}(user{:}0, job) = \frac{|\{Student\} \setminus \{Student\}|}{|\{Student, Professor\} \setminus \{Student\}| + 1} = \frac{0}{2} = 0$, $AM_n'^{\overline{G}_2}(user{:}0, age) = \frac{|\min\{18,19\} - \min\{18\}| + |\max\{18,19\} - \max\{18\}|}{|\min\{18,19,40,50\} - \min\{18\}| + |\max\{18,19,40,50\} - \max\{18\}|} + 1 = \frac{0+1}{0+32+1} = 0.03$. The information loss of anonymizing *user:2*'s *job* is $AM_c'^{\overline{G}_2}(user{:}2, job) = \frac{0}{2} = 0$, $AM_n'^{\overline{G}_2}(user{:}2, age) = \frac{1+0}{1+31+1} = 0.03$. The information loss of anonymizing all attributes of *user:0* and *user:2* are: $AM'^{\overline{G}_2}(user{:}0) = AM'^{\overline{G}_2}(user{:}2) = \frac{0+0.03}{2} = 0.015$. The information loss of making *user:0* and *user:2* having the same values for their attributes is: $AM^{\overline{G}_2}(user{:}0, user{:}2) = \frac{0.015+0.015}{2} = 0.015$.

To measure the loss of out- and in-degree information on a user u, we calculate the difference between the out- and in-degree of u in the original and anonymized KG. At this purpose, we extend metrics described in Chapter 3 to consider also multiple relationship types.

Definition 12 (DM'_o)**.** *Let u be a user in G. The out-degree loss metric (DM'_o) of anonymizing user u in $\overline{G}$ is:*

$$DM'^{\overline{G}}_{o}(u) = \frac{1}{|R^{UU}|} \times \sum_{r}^{R^{UU}} \frac{d_o(\overline{G}, r, u) - d_o(G, r, u)}{|V^U|}$$

where $d_o(G, r, u)$ and $d_o(\overline{G}, r, u)$ are the out-degree of the relationship type $r \in R^{UU}$ of user u in G and $\overline{G}$, respectively.

Similarly, we define the in-degree information loss (DM'_i) of anonymizing a user u in KG as follows:

Definition 13 (DM'_i)**.** *Let u be a user in G. The in-degree loss metric (DM'_i) of anonymizing user u in $\overline{G}$ is:*

$$DM'^{\overline{G}}_{i}(u) = \frac{1}{|R^{UU}|} \times \sum_{r}^{R^{UU}} \frac{d_i(\overline{G}, r, u) - d_i(G, r, u)}{|V^U|}$$

where $d_i(G, r, u)$ and $d_i(\overline{G}, r, u)$ are the in-degree of the relationship type $r \in R^{UU}$ of user u in G and $\overline{G}$, resp.

By exploiting these metrics, we define the Out- and In-Degree Information Loss (DM) of making identical the out- and in-degree on all types of relationships of two users.

Definition 14. *Let u, v be two users in G. The Out- and In-Degree Information Loss Metric (DM) of making u and v having the same out- and in-degree on all types of relationships in $\overline{G}$ is defined as follows:*

$$DM^{\overline{G}}(u, v) = \frac{DM'^{\overline{G}}_{o}(u) + DM'^{\overline{G}}_{o}(v) + DM'^{\overline{G}}_{i}(u) + DM'^{\overline{G}}_{i}(v)}{4}$$

Example 10. Let $\overline{G}_2$ be the KG showed in Fig.4.1b and G be its original version. $R^{UU} = \{follows, is\_tutor\}$ and $|V^U| = 4$. If we make *user:0* and *user:2* have the same out- and in-degree on all types of relationships: $d_o(\overline{G}_2, follows, user{:}0) = d_o(\overline{G}_2, follows, user{:}2) = 1$, $d_o(\overline{G}_2, is\_tutor, user{:}0) = d_o(\overline{G}_2, is\_tutor, user{:}2) = 0$; $d_i(\overline{G}_2, follows, user{:}0) = d_i(\overline{G}_2, follows, user{:}2) = 0$, $d_i(\overline{G}_2, is\_tutor, user{:}0) = d_i(\overline{G}_2, is\_tutor, user{:}2) = 1$. Then, $DM'^{\overline{G}_2}_{o}(user{:}0) = \frac{1}{2} \times (\frac{1-1}{4} + \frac{0-0}{4}) = 0$ and $DM'^{\overline{G}_2}_{i}(user{:}0) = \frac{1}{2} \times (\frac{0-0}{4} + \frac{1-0}{4}) = 0.125$. Similarly, $DM'^{\overline{G}_2}_{o}(user{:}2) = \frac{1}{2} \times (\frac{1-0}{4} + \frac{0-0}{4}) = 0.125$ and $DM'^{\overline{G}_2}_{i}(user{:}2) = \frac{1}{2} \times (\frac{0-0}{4} + \frac{1-1}{4}) = 0$. Out- and In-Degree Information Loss of anonymizing *user:0* and *user:2* is $DM^{\overline{G}_2}(user{:}0, user{:}2) = \frac{0+0.125+0.125+0}{4} = 0.0615$.

Finally, we combine AM and DM in the Attribute and Degree Information Loss Metric (ADM).

Definition 15 (ADM). *Let u, v be two users in G. The Attribute and Degree Information Loss Metric (ADM) of making u and v having the same values on all attributes, and the same out- and in-degree on all types of relationships in $\overline{G}$ is as follows:*

$$ADM^{\overline{G}}(u,v) = \alpha^{AM} \times AM^{\overline{G}}(u,v) + (1-\alpha^{AM}) \times DM^{\overline{G}}(u,v)$$

where α^{AM} is a number between 0 and 1.

As anonymizing KGs modifies both users' attributes and relationships, we use α^{AM} to control the information loss on these two categories of information. If data providers want to preserve users' attributes more than their relationships, they can assign α^{AM} a value greater than 0.5. They can also assign α^{AM} a values less than 0.5 if they want to preserve more users' relationships. Otherwise, they can assign α^{AM} to 0.5.

Example 11. With $\alpha^{AM} = 0.5$, the information loss of making values of all attributes and the out- and in-degrees of all types of relationships of *user:0* and *user:2* in $\overline{G}_2$ (Fig.4.1b) identical is: $ADM^{\overline{G}_2}(\textit{user:0}, \textit{user:2}) = \alpha^{AM} \times AM^{\overline{G}_2}(\textit{user:0}, \textit{user:2}) + (1 - \alpha^{AM}) \times DM^{\overline{G}_2}(\textit{user:0}, \textit{user:2}) = 0.5 \times 0.015 + (1-0.5) \times 0.0615 = 0.03825$.

4.3.2 The Attribute Truthfulness Information Loss

AM measures the information loss of users' attribute independently, whereas it does not consider the associations between values of different attributes. In particular, let u be a user. We denote with $h \leftrightarrow t$, where $h, t \in I_a(\overline{G}, u)$, the association between his/her values in $\overline{G}$. These associations can be extracted by finding all combinations of size 2 in $I_a(\overline{G}, u)$. By using the original KG G as the knowledge base, we assume an association is truthful if it can be extracted from the attributes of any user in G. In addition, if two associations $h_1 \leftrightarrow t_1$ and $h_2 \leftrightarrow t_2$ are truthful, $h_1 \leftrightarrow t_2$ and $h_2 \leftrightarrow t_1$ are truthful as well. Conversely, any association that cannot be extracted from G is untruthful. Then, the truthfulness of u's attributes in $\overline{G}$ can be measured as the number of truthful associations that can be extracted from $I_a(\overline{G}, u)$.

Example 12. Let $\overline{G}_2$ be the KG showed in Fig.4.1b and G be its original version. $I_a(G, \textit{user:0}) = \{(\textit{job}, \textit{Student}), (\textit{age}, 18)\}$, $I_a(G, \textit{user:2}) = \{(\textit{job}, \textit{Student}), (\textit{age}, 19)\}$. As we make values of *user:0* and *user:2*'s attributes in $\overline{G}_2$ identical, $I_a(\overline{G}_2, \textit{user:0}) = I_a(\overline{G}_2, \textit{user:2}) = \{(\textit{job}, \textit{Student}), (\textit{age}, 18), (\textit{age}, 19)\}$. We can extract the following associations $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 18)$, $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 19)$, $(\textit{age}, 19) \leftrightarrow (\textit{age}, 18)$. $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 18)$ and $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 19)$ are truthful as they can be extracted from $I_a(G, \textit{user:0})$ and $I_a(G, \textit{user:2})$ while $(\textit{age}, 18) \leftrightarrow (\textit{age}, 19)$ is truthful as $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 18)$ and $(\textit{job}, \textit{Student}) \leftrightarrow (\textit{age}, 19)$ are truthful.

Let D^+ and D^- be the set of truthful and untruthful associations, respectively. The naive solution to check whether an association $(r_a, v_a) \leftrightarrow (r'_a, v'_a)$ is truthful or not is to check if it exists in D^+ or D^-. However, this solution is time consuming and impractical due to the high number of associations in D^+ and D^-. Therefore, we implement a bilinear function $f((r_a, v_a), (r'_a, v'_a))$ to measure the probability that the association $(r_a, v_a) \leftrightarrow (r'_a, v'_a)$ is truthful. We define f as follows:

$$g(r_a, v_a) = \tanh(e_{r_a} * W_g * e_{v_a} + b_g)$$
$$f((r_a, v_a), (r'_a, v'_a)) = \text{sigmoid}(g(r_a, v_a) * W_f * g(r'_a, v'_a) + b_f)$$

where $e_{r_a}, e_{v_a} \in \mathbb{R}^{d_1}$ are d_1-dimensional vectors illustrating r_a, v_a. $W_g \in \mathbb{R}^{d_1 \times d_1 \times d_1}$, $W_f \in \mathbb{R}^{d_1 \times d_1}$, $b_g, b_f \in \mathbb{R}^{d_1}$ are the parameters of f and g. Here, we use tanh and sigmoid to normalize the outputs of g and f to $[-1, 1]$ and $[0, 1]$, respectively, as they are showed to achieve good results in previous work [17].

We train f in the set of all associations $D = D^+ \cup D^-$. Each association $h \leftrightarrow t \in D$ is assigned a positive label $y = 1$, if $h \leftrightarrow t \in D^+$, and $y = 0$, otherwise. Let θ be the set of all parameters of f and g. We learn f by minimizing the Cross Entropy Loss function:

$$\mathcal{J}_1(\theta) = -\sum_{x}^{D} y \log(f(h,t)) + (1-y)\log(1 - f(h,t)) \tag{4.1}$$

By using the learned function f, we implement an indicator $\mathcal{R}$ deciding whether the association $h \leftrightarrow t$ is truthful or not as follows:

$$\mathcal{R}(h,t) = \begin{cases} 1, \text{if } f(h,t) \geq 0.5 \\ 0, \text{otherwise} \end{cases}$$

We therefore introduce the Attribute Truthfulness Information Loss Metric (ATM) to minimize the number of untruthful associations in the anonymized KGs.

Definition 16 (ATM). *Let u, v be two users in G, and $\mathcal{R}$ be the indicator deciding whether an association is truthful. ATM is defined as follows:*

$$ATM'^{\overline{G}}(u) = \frac{1}{|I_a(\overline{G}, u)|^2} \sum_{(r,v)}^{I_a(\overline{G},u)} \sum_{(r',v')}^{I_a(\overline{G},u)} 1 - \mathcal{R}((r_a, v_a), (r'_a, v'_a))$$

$$ATM^{\overline{G}}(u, v) = \frac{ATM'^{\overline{G}}(u) + ATM'^{\overline{G}}(v)}{2}$$

Furthermore, we define the Attribute Truthfulness and Degree Information Loss Metric ($ATDM$) by combining ATM and DM to minimize the untruthfulness of users' attributes and the degree information loss:

Definition 17 (ATDM). *Let u, v be two users in G. The Attribute Truthfulness and Degree Information Loss ($ATDM$) of making u and v having the same values on all attributes, and the same out- and in-degree on all types of relationships in $\overline{G}$ identical is:*

$$ATDM^{\overline{G}}(u, v) = \alpha^{ATM} \times ATM^{\overline{G}}(u, v) + (1 - \alpha^{ATM}) \times DM^{\overline{G}}(u, v)$$

where α^{ATM} is a number between 0 and 1.

Similar to ADM, $ATDM$ uses α^{ATM} to control the contribution of ATM and DM to $ATDM$.

4.4 Cluster-Based Knowledge Graph Anonymization

Our Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA) is designed to modify the structure of the original KG such that users' identities in the anonymized KG are protected according to k-ad while maximizing its quality. Given a KG $G(V, E, R)$ and a positive number k, CKGA generates G's k-ad anonymized version $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ according to three main steps:

1. **Users' points generation.** This step generates a point $e_u \in \mathbb{R}^{d_2}$ for each user $u \in V^U$ such that the Euclidean distance between two points e_u, e_v is nearly equal to the information loss, measured by using ADM and $ATDM$, of their corresponding users u, v.[2]

2. **Clusters generation.** The goal of this step is to construct a set of user clusters $C^{\overline{G}} = \{c \subseteq \overline{V}^U || c| >= k\}$ that minimizes the Euclidean distances between the points of users who are in the same cluster.

3. **Knowledge graph generalization.** In this step, we add and remove edges such that all users in the same cluster have the same values for all of their attributes and the same out- and in-degrees for all relationships.

4.4.1 Users' Points Generation

Given a KG G and a positive number d_2. We denote with $InfoLoss(u, v)$ the information loss of two users u, v in V^U, measured by using either ADM or $ATDM$. To generate users' points, we first generate a random point $e_u \in \mathbb{R}^{d_2}$ for every user $u \in V^U$. Then, given two users $u, v \in V^U$, we calculate the Squared Euclidean distance between their points e_u, e_v: $d^2(e_u, e_v) = \sum_{i=1}^{d_2}(e_u[i] - e_v[i])^2$.

Then, for every pair of users u, v in V^U, we minimize the differences between the squared information loss of u, v: $InfoLoss(u, v)^2$ and the Squared Euclidean distance between their corresponding points e_u, e_v: $d^2(u, v)$ by using the Mean Squared Error loss function:

$$\mathcal{J}_2(\theta) = \sum_{u,v \in V^U \times V^U} (d^2(e_u, e_v) - InfoLoss(u, v)^2)^2 \tag{4.2}$$

After minimizing $\mathcal{J}_2$, we obtain points such that for every pair of users $u, v \in V^U$, their squared $InfoLoss(u, v)^2$ is almost equal to the Squared Euclidean distance between their points e_u, e_v:$d^2(e_u, e_v)$. In other words, $InfoLoss(u, v)$ is almost equal to $d(e_u, e_v)$.

4.4.2 Clusters Generation

Given a KG G, a positive number k, the generated points $\mathcal{U}$ representing users in G, and a clustering algorithm $\mathcal{A}$, we aim at generating a set of clusters that have at least k users such

[2]Although we use ADM and $ATDM$, our algorithm can be easily extended by using other information loss metrics as well.

Algorithm 3 k-Means Partition(C,$\mathcal{U}, k, \tau$)

Input: C: the set of clusters generated by the clustering algorithm $\mathcal{A}$; $\mathcal{U}$: points of users; k: a positive number; and τ: a number in $[0, 1]$.

Output: The set of clusters $C^{\overline{G}}$.

1: Let Δ be the set of users whose clusters have less than k users.
2: $C^{\overline{G}} \leftarrow \{c | c \in C \wedge |c| \geq k\}$
3: $assign\_new\_clusters(\Delta, C^{\overline{G}}, \mathcal{U}, k, \tau)$
4: **while** $C^{\overline{G}} \neq \varnothing$ **do**
5: Let c be the first cluster in C
6: **if** $|c| \geq 2 * k$ **then**
7: $C^{\overline{G}} \leftarrow C^{\overline{G}} \setminus \{c\}$
8: Let $\mathcal{U}_c$ be the points in $\mathcal{U}$ representing users in c.
9: $C_{new} = run\_balanced\_kmeans(\mathcal{U}_c, |c|/k)$
10: $C^{\overline{G}} \leftarrow C^{\overline{G}} \cup C_{new}$
11: **end if**
12: **end while**
13: **return** $C^{\overline{G}}$

that the average Euclidean distances between the points of the users in the same cluster are minimized. As we represent users by using points in Euclidean space, most of the state-of-the-art clustering algorithms (e.g., k-means, DBSCAN [14], and HDBSCAN [9]) can be used.

As the clustering algorithm $\mathcal{A}$ can generate clusters that have less than k users, we use two strategies to make all clusters having at least k users. The first one is the Invalid Removal strategy (IR), which removes all clusters that have less than k users. However, this approach can remove too many users since some clustering algorithms (e.g., k-means) do not consider how many users each cluster has. Furthermore, as showed in Chapter 3, the more users each cluster has, the more information we lose. To cope with these issues, we present the k-Means Partition strategy (KP). Instead of removing users whose clusters have less than k users, KP adds these users to their nearest clusters that have at least k users. To prevent the resulting clusters from having too many users, KP splits clusters that have at least $2 \times k$ users such that the number of users in all of these clusters is between k and $2 \times k - 1$.

However, adding all users whose clusters have less than k users to new clusters can make the resulting clusters containing outliers whose distances to remaining users in the merged clusters are too big. Therefore, our algorithm uses a parameter $\tau \in [0, 1]$ to prevent merging these outliers. We denote with d_{min}, d_{max} the minimum and maximum Euclidean distance between all users' points in $\mathcal{U}$. Then, $\tau_d = \tau \times (d_{max} - d_{min}) + d_{min}$. We only merge a user to a cluster if the maximum distance between this user and all users in the cluster is less than or equal to τ_d.

Algorithm 3 takes as input the clusters of users generated by the selected clustering

Procedure 1 *assign_new_clusters* $(\Delta, C^{\overline{G}}, \mathcal{U}, k, \tau)$

1: Let d_{max}, d_{min} be the maximum and minimum distance between all users' points in $\mathcal{U}$.
2: $\tau_d = \tau \times (d_{max} - d_{min}) + d_{min}$.
3: **for** $u \in \Delta$ **do**
4: Let $e_u \in \mathcal{U}$ be the point of user u.
5: $d^c_{min} \leftarrow +\infty$
6: **for** $c \in C^{\overline{G}}$ **do**
7: $d_c \leftarrow 0$
8: **for** $v \in c$ **do**
9: Let $e_v \in \mathcal{U}$ be the point of user v.
10: $d_c \leftarrow \max\{d_c, d(e_u, e_v)\}$
11: **end for**
12: **if** $d_c < d^c_{min}$ **and** $d_c \leq \tau_d$ **then**
13: $d^c_{min} \leftarrow d_c$
14: $c_{selected} \leftarrow c$
15: **end if**
16: **end for**
17: **if** $d^c_{min} \neq +\infty$ **then**
18: $c_{selected} \leftarrow c_{selected} \cup \{u\}$
19: **end if**
20: **end for**

algorithm $\mathcal{A}$, C, the set of points representing users $\mathcal{U}$, a positive number k, and the threshold τ. At the beginning, it finds the set of users Δ whose clusters have less than k users (line 1) and the set $C^{\overline{G}}$ containing clusters that have at least k users (line 2). Then, it calls function *assign_new_clusters*() to assign a cluster in $C^{\overline{G}}$ to each user in Δ (line 3). Then, for each cluster $c \in C^{\overline{G}}$ that has more than $2 \times k$ users, it removes this cluster from $C^{\overline{G}}$ (line 7), finds points $\mathcal{U}_c$ corresponding to users in c (line 8), and uses the Balanced k-Means algorithm [38] to split c to $|c|/k$ smaller clusters C_{new} which have at least k users (line 9). This algorithm finds $|c|/k$ centers, each of which represents a cluster in C_{new}. Then, it assigns users in c to these clusters such that all clusters in C_{new} have at least k users and the average squared Euclidean distances of user's points the same cluster is minimized. C_{new} is then added to $C^{\overline{G}}$ (line 10). Finally, it returns $C^{\overline{G}}$ (line 13).

Procedure *assign_new_clusters()*. Given a set of users Δ, a set of clusters $C^{\overline{G}}$, a set of users' points $\mathcal{U}$, a positive number k, and the parameter τ, it assigns a cluster in $C^{\overline{G}}$ for each user in Δ. At the beginning it calculates τ_d (line 2). Then, for each user u in Δ, it finds u's point: e_u and calculates the maximum Euclidean distance between e_u and points of users in c: d_c (lines 7-11). If d_c is less than d^c_{min} and less than or equal to τ_d, it updates d^c_{min} and $c_{selected}$ with d_c and c, respectively (lines 13-14). Next, if d^c_{min} is not equal to $+\infty$, it adds u to $c_{selected}$ (line 18).

It is straightforward to show that, given a set of clusters C and denoting with $C^{\overline{G}}$

Algorithm 4 Attributes Generalization $(G, C^{\overline{G}})$

Input: Graph $G(V, E, R)$; clusters of users $C^{\overline{G}}$.
Output: The set of edges representing users' attributes $\overline{E}^{UA}$.

1: $\overline{E}^{UA} \leftarrow \varnothing$
2: **for** $c \in C^{\overline{G}}$ **do**
3: $\quad \Im_c^G \leftarrow \bigcup_{u \in c} I_a(G, u)$
4: $\quad$ **for** $u \in c$ **do**
5: $\quad\quad$ **for** $(r_a, v_a) \in \Im_c^G$ **do**
6: $\quad\quad\quad \overline{E}^{UA} \leftarrow \overline{E}^{UA} \cup \{(u, r_a, v_a)\}$
7: $\quad\quad$ **end for**
8: $\quad$ **end for**
9: **end for**
10: **return** $\overline{E}^{UA}$

the set of clusters returned by Algorithm 3, executed with a positive number k and the parameter τ, all clusters in $C^{\overline{G}}$ have at least k users. Indeed, let c be an arbitrary cluster in $C^{\overline{G}}$. Suppose that c has less than k users. As Algorithm 3 only adds clusters that have at least k users to $C^{\overline{G}}$ (lines 2 and 10), $C^{\overline{G}}$ cannot contain c.

4.4.3 Knowledge Graph Generalization

Given a KG $G(V, E, R)$ and the set of clusters $C^{\overline{G}}$ returned by Algorithm 3, we aim at generating the anonymized KG $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ such that all users in the same cluster have the same information (i.e., $I^{\overline{G}}$). At this purpose, we present the Knowledge Graph Generalization algorithm (KGG) to add and remove edges such that the values of all attributes and the out- and in-degree of all types of relationships of users in the same cluster are identical. KGG contains two steps: 1) **Attributes generalization**. The goal of this step is to generate values of users' attributes such that all users in the same cluster have the same values for all of their attributes. Formally, $\forall c \in C^{\overline{G}} \wedge \forall u, v \in c \wedge I_a(\overline{G}, u) = I_a(\overline{G}, v)$; 2) **Out- and in-degree generalization**. In this step, we aim at adding and removing users' relationships such that every user in the same cluster has the same out- and in-degree for all types of his/her relationships. More precisely, $\forall c \in C^{\overline{G}} \wedge \forall u, v \in c \wedge I_o(\overline{G}, u) = I_o(\overline{G}, v) \wedge I_i(\overline{G}, u) = I_i(\overline{G}, v)$.

For step 1, we have defined Algorithm 4. It takes as input the original KG G and the set of clusters $C^{\overline{G}}$ generated by Algorithm 3. First, the algorithm initializes empty the set of edges describing users' attributes $\overline{E}^{UA}$ (line 1). Then, for every cluster c in $C^{\overline{G}}$, it calculates the union of attributes' values in G (i.e., I_a) of all users $u \in c$: $\Im_c^G$ (line 3). For each user u in c, it adds edge (u, r_a, v_a) to represent that u has value v_a for attribute r_a for all pairs (r_a, v_a) in $\Im_c^G$ (line 6). Finally, it returns $\overline{E}^{UA}$ (line 10). Therefore, attributes' values of users in the same clusters are identical.

Theorem 6. Given a KG G and the set of clusters $C^{\overline{G}}$ generated by Algorithm 3. Let

$\overline{E}^{UA}$ be set of edges representing users' attributes in $\overline{G}$ created by Algorithm 4. For every cluster c in $C^{\overline{G}}$, for every user u and v in c, $I_a(\overline{G}, u) = I_a(\overline{G}, v)$.

Proof. Suppose that there is a cluster $c \in C^{\overline{G}}$ containing two users u, v such that $I_a(\overline{G}, u) \neq I_a(\overline{G}, v)$. However, if u, v are in the same cluster c, $I_a(\overline{G}, u) = I_a(\overline{G}, v)$ as Algorithm 4 initializes $\overline{E}^{UA}$ to the empty set (line 1) and adds the same attributes' values to u and v (lines 4-8). Therefore, we can conclude that, for every cluster c in $C^{\overline{G}}$, for every user u and v in c, $I_a(\overline{G}, u) = I_a(\overline{G}, v)$. □

The Out- and In-Degree Generalization algorithm takes as input the original KG G and the set of clusters $C^{\overline{G}}$. For every relationship type $r_u \in R^{UU}$, it uses the Directed Graph Generalization algorithm (DGG) in Chapter 3 to make the out- and in-degree for r_u of all users in the same cluster identical. Therefore, the out- and in-degrees for every relationship types of users in the same cluster are identical.

As attributes' values and out- and in-degrees in the anonymized KG $\overline{G}$ of users in the same cluster are identical, $\overline{G}$ satisfy k-ad.

Theorem 7. Given a KG $G(V, E, R)$ and the set of clusters $C^{\overline{G}}$ created by Algorithm 3. Let $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be the anonymized version of G created by the Knowledge Graph Generalization algorithm. $\overline{G}$ satisfies k-ad.

Proof. As $C^{\overline{G}}$ is generated by Algorithm 3, all clusters in $C^{\overline{G}}$ have at least k users. Then, for every user, there is a cluster $\mathcal{C}(\overline{G}, u) \in C^{\overline{G}}$ such that $|\mathcal{C}(\overline{G}, u)| \geq k$. Additionally, for every user $u, v \in \overline{V}^U$, if u and v are in the same cluster, $I_a(\overline{G}, u) = I_a(\overline{G}, v)$ (according to Theorem 6), $I_o(\overline{G}, u) = I_o(\overline{G}, v)$, and $I_i(\overline{G}, u) = I_i(\overline{G}, v)$ (according to Theorem 2). Then, for every user $u \in \overline{V}^U$, there is a set of users $\mathcal{C}(\overline{G}, u) = \{v | v \in \overline{V}^U \wedge I_a(\overline{G}, u) = I_a(\overline{G}, v) \wedge I_o(\overline{G}, u) = I_o(\overline{G}, v) \wedge I_i(\overline{G}, u) = I_i(\overline{G}, v)\}$ and $|\mathcal{C}(\overline{G}, u)| \geq k$. Therefore, we can conclude that $\overline{G}$ satisfies k-ad. □

4.5 Experiments

In this section, we evaluate the quality of anonymized KGs generated by the proposed anonymization algorithm.

4.5.1 Datasets

Due to the flexibility of KGs in illustrating users' data, we use different kinds of real-world datasets to evaluate the capabilities of the proposed technique, namely *Email-Eu-core*, *Google+*, and *Freebase* datasets. Also, we use *Email-temp* and *DBLP* to compare our work with previous algorithms: DGA [10] and CDGA in Chapter 3. Appendix B describes these datasets and shows their properties in details.

4.5.2 Evaluating Users' Points

In this experiment, we evaluate the impact of d_2 to the difference of the Euclidean distance between data points in Euclidean space representing users and the information loss

of making their information identical. We generate these points by minimizing the loss function $\mathcal{J}_2$ (equation 4.2) until the mean of these differences has stopped decreasing for 50 epochs. The initial learning rate is 0.1 and we decrease it by multiplying it with 0.5 if the mean of these differences does not decrease for 10 epochs. We implement this step by using PyTorch.

Table. 4.1 illustrates mean and standard deviation of the differences between ADM of users and the Euclidean distance of their points in all datasets. Here, we use ADM with $\alpha^{ADM} = 0.5$ and $\alpha^{DM} = 0.5$. In all datasets, the higher d_2 is, the lower the differences are. In *Email-Eu-core*, by increasing d_2 from 2 to 50, the mean and standard deviation of these differences decreases from 0.0046 to 0.0005 and from 0.0038 to 0.0009, respectively. At $d_2 = 50$, these differences are very small in all datasets.

Therefore, our approach is efficient enough to find points representing users such that the differences between the Euclidean distances of these points are almost similar to the information loss of their corresponding users.

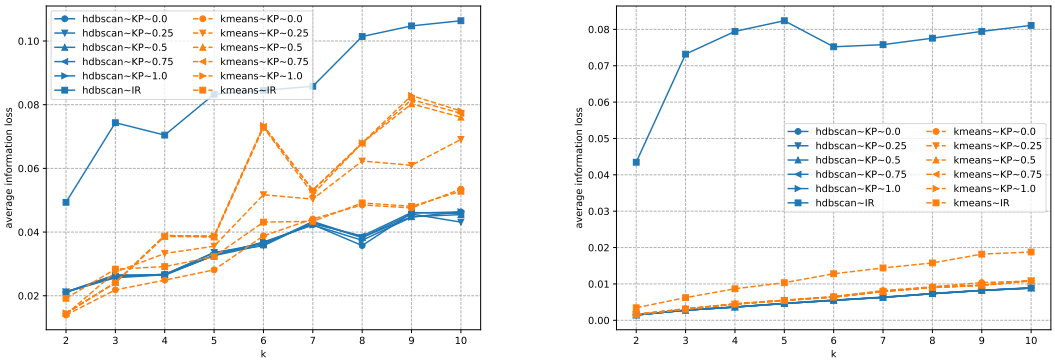
4.5.3 Tuning CKGA

In this experiment, we aim at evaluating the effects of the adopted clustering algorithm $\mathcal{A}$; of the strategies IR, KP; and k to the average information loss. The average information loss is calculated by taking the average of the loss of users' attributes (i.e., $AM'^{\overline{G}}$), the out- and in-degrees (i.e., $DM'^{\overline{G}}_o$, and $DM'^{\overline{G}}_i$).

Effects of $\mathcal{A}$. Figure. 4.2 shows the average information loss in two datasets: *Email-Eu-core* and *Freebase*, where k-means [14] and HDBSCAN [9] are chosen. We choose these algorithms as they are the state-of-the-art algorithms of two most popular clustering approaches: centroid-based and density-based clustering. With k-means, we assign the number of resulting clusters to $\frac{|V^U|}{k}$. Also, we assign the minimum size of all resulting clusters to k, when running HDBSCAN. We keep default values for all of the remaining parameters of these algorithms. Also, we use IR to show the effects of $\mathcal{A}$ as it only removes clusters that have less than k users. k-means returns anonymized KGs with the information loss lower than those returned from HDBSCAN in both datasets (about 0.06 at $k = 10$). The reason is that HDBSCAN's clusters have more users than those of k-means. At $k = 10$, the average number of users in clusters returned from HDBSCAN and k-means are 36.89 and 22.33 in *Email-Eu-core*, and 76.40 and 21.17 in *Freebase*, respectively.

Table 4.1: The mean (± standard deviation) of the differences between the Euclidean distance of the learned points and the ADM of the corresponding users.

Dataset	$\mathbf{d_2 = 2}$	$\mathbf{d_2 = 10}$	$\mathbf{d_2 = 50}$
Email-Eu-core	0.0046 (±0.0038)	0.0012 (±0.0015)	**0.0005** (±**0.0009**)
Google+	0.0099 (±0.0083)	0.0054 (±0.0040)	**0.0008** (±**0.0012**)
Freebase	0.0072 (±0.0073)	0.0036 (±0.0032)	**0.0003** (±**0.0010**)
Email-temp	0.0030 (±0.0030)	0.0019 (±0.0012)	**0.0001** (±**0.0002**)
DBLP	0.0073 (±0.0021)	0.0031 (±0.0011)	**0.0002** (±**0.0001**)



(a) Email-Eu-core (b) Freebase

Figure 4.2: Average information loss of users with varying clustering algorithms and IR/KP strategies.

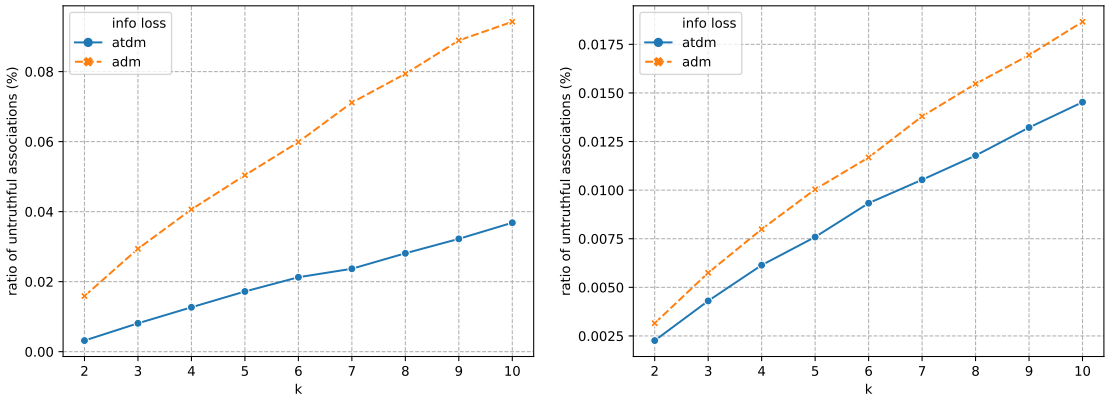
However, IR removes about 30% and 25% of users returned from HDBSCAN and k-means, respectively, in the *Email-Eu-core* dataset, whereas about 20% and 27% of users returned from HDBSCAN and k-means, respectively, are removed by IR in the *Freebase* dataset. IR only removes users returned from k-means as all clusters returned from HDBSCAN have at least k users. Therefore, k-means is better than HDBSCAN as it returns clusters that have less users than those returned from HDBSCAN.

Effects of IR and KP. KP ensures that the resulting clusters have a number of users from k to $2 \times k - 1$. Thus, we can decrease the average information loss of anonymized KGs returned from HDBSCAN by 0.06 and 0.07 in the *Email-Eu-core* and *Freebase* dataset, respectively. Although we increase the average information loss of the anonymized KGs returned from k-means by at least 0.02 in Email-Eu-core, we preserve at least 22% more users than those generated by IR. KP decreases the average information loss of the anonymized KGs returned from HDBSCAN more than those returned from k-means because HDBSCAN generates clusters that have more users than k-means' clusters. At $k = 10$, by increasing τ from 0 to 1.0, we decrease the average information loss of the anonymized KGs returned from k-means from 0.08 to 0.05 and increase the ratio of remaining users from about 74% to 100% in both datasets. Here, τ does not affect the ratio of remaining users of the anonymized KGs returned from HDBSCAN as HDBSCAN does not return any cluster that have less than k users. The anonymized KGs executed with HDBSCAN and KP have lower average information loss than those executed with k-means in both datasets as they contain less users than the other ones. Therefore, KP is more effective than IR in improving the quality of anonymized KGs.

Effects of k. The results of these experiments show that the quality of the anonymized KGs decreases by increasing k in all datasets.

Table 4.2: Accuracy of the indicator $\mathcal{R}$ (%).

Dataset	$\mathbf{d_1 = 2}$	$\mathbf{d_1 = 10}$	$\mathbf{d_1 = 50}$
Google+	97.91	99.43	**99.91**
Freebase	95.7	95.7	**96.9**



(a) Google+ (b) Freebase

Figure 4.3: Ratio of untruthful associations by using ADM and $ATDM$.

4.5.4 Evaluating the Truthfulness of Anonymized KGs

In this experiment, we investigate the impact of using ADM and $ATDM$ on minimizing the number of untruthful associations.

We first analyze the impact of d_1 on the accuracy of the indicator $\mathcal{R}$. The accuracy is evaluated in truthful/untruthful associations extracted from the *Google+* and *Freebase* datasets. We optimize $\mathcal{J}_1$ (equation 4.1) as in the previous experiment in Section 4.5.2. Table 4.2 shows the accuracy of the trained models on varying values of d_1. The accuracy is higher than 95%. In both datasets, the accuracy is increased by increasing d_1 from 5 to 50 (2% in *Google+* and 1.2% in *Freebase*). We achieve the highest accuracy with $d = 50$ on both the *Google+* and *Freebase* datasets.

We use the trained models with $d_1 = 50$ to evaluate the impacts of $ATDM$ and ADM on the ratio of the untruthful associations in anonymized KGs. We normalize the number of untruthful associations by the number of maximum untruthful associations in the original KG: $\frac{D^+(\overline{G}) \setminus D^+(G)}{D^-(G)}$. Figure. 4.3 illustrates the ratio of untruthful associations of anonymized KGs by considering with ADM and $ATDM$ in: *Google+* (Figure. 4.3a) and *Freebase* (Figure. 4.3b). We use k-means and KP strategy with $\tau = 1.0$ to generate the anonymized KGs in this experiment. The anonymized KGs generated by using $ATDM$ contain less untruthful associations than those generated by using ADM in both datasets. At $k = 10$, $ATDM$ decreases the ratio of untruthful associations by 0.04% compared to those generated by ADM in the *Google+* dataset. We achieve similar results for the *Freebase* dataset. The

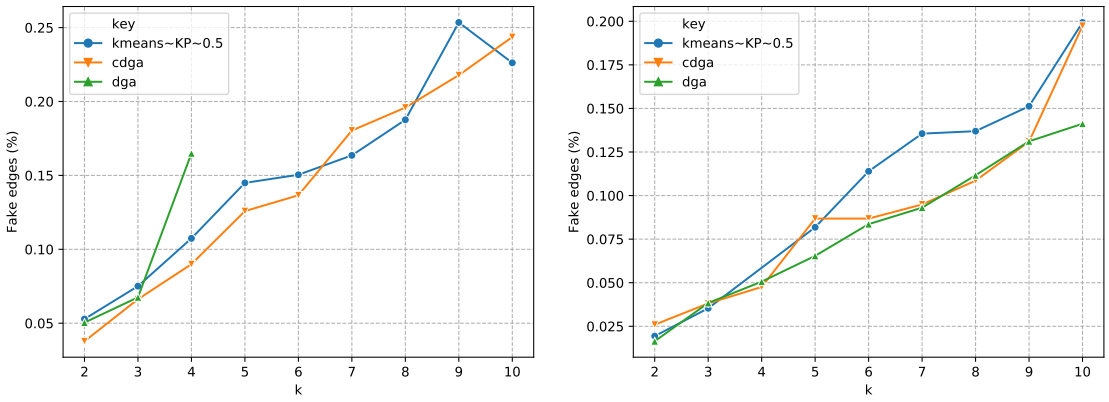
Table 4.3: Performance of our algorithm (CKGA) and CDGA on varying values of k (seconds).

	Email-temp		**DBLP**	
k	**CDGA**	**CKGA**	**CDGA**	**CKGA**
2	680.8	6.5	91,855.4	607.9
3	825.7	4.4	111,181.6	420.4
4	893.3	3.2	118,269.7	629.9
5	929.7	2.7	122,246.1	581.5
6	950.4	2.3	124,145.7	540.3
7	977.4	1.9	126,542.7	483.3
8	987.9	1.8	127,343.4	474.9
9	1,000.7	1.5	128,254.3	457.6
10	1,007.1	1.5	128,727.9	398.9

anonymized KGs of *Freebase* contain less untruthful associations than those of *Google+* as *Freebase* contains less untruthful associations than *Google+* does. Therefore, $ATDM$ is better than ADM in making users' attributes truthful.

4.5.5 Comparative Analysis

In this experiment, we compare the quality of anonymized KGs generated by our algorithm and those generated by DGA [10] and CDGA (in Chapter 3) on two datasets: *DBLP* and *Email-temp*. Furthermore, we compare the efficiency of our algorithm and CDGA as we have their implementation. We generate users' points with $d_2 = 50$ and use $\mathcal{A} = k\text{-}means$ and KP strategy with $\tau = 0.5$ to anonymize both datasets.



(a) Email-temp (b) DBLP

Figure 4.4: Ratio of fake edges of anonymized graphs returned by DGA, CDGA, and our algorithm on varying values of k.

As our algorithm removes some users to reduce the number of fake edges of the anonymized KGs, we use the number of fake edges of the remaining users in the anonymized KGs to compare our work with DGA and CDGA. Figure. 4.4 illustrates the results. We take results of DGA and CDGA from the respective papers [10, 21]. For both datasets, the anonymized KGs returned by our algorithm contain a number of fake edges similar to that of those returned from CDGA and DGA while preserving more than 99% of users in the original KG. Furthermore, the execution time of our algorithm is extremely lower than that of CDGA. Although our algorithm needs 408 and 40,415 seconds to train users' points for the *Email-temp* and *DBLP* datasets, the trained points can be reused to anonymize users at any values of k as well as tune the quality of anonymized KGs under different values of our parameters. After obtaining the trained points, it only needs 1.5 seconds and 398.9 seconds to anonymize KGs that have at least $k = 10$ users for the *Email-temp* and *DBLP* datasets, respectively, while CDGA needs 1007.1 and 128,727.9 seconds. Therefore, data providers can use our algorithm to anonymize directed graphs satisfying the Paired k-degree [10] and K-In&Out-Degree Anonymity [60] and achieve good quality anonymized graphs.

Chapter 5

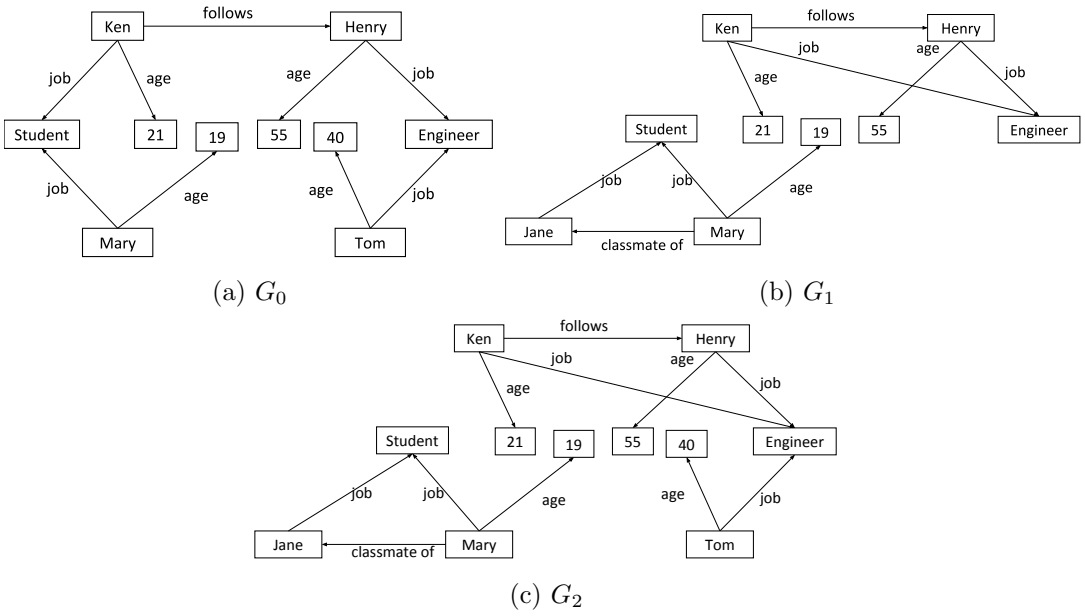
Sequential Anonymization of Knowledge Graphs

5.1 Introduction

In Chapter 4, we described k-Attribute Degree (k-ad), the first model that protects users in a KG from being re-identified. Due to the flexibility of k-ad, it protects users' identities not only in anonymized KGs but also in anonymized directed graphs and relational data. However, k-ad cannot protect users if data providers periodically publish new versions of the anonymized KG. The following example shows how adversaries can re-identify users in anonymized KGs satisfying k-ad.

Example 13. Figure. 5.1 and Figure. 5.2 illustrate the original KGs and their 2-ad anonymized versions at time: $0, 1, 2$. As $\overline{G}'_0$, $\overline{G}'_1$, and $\overline{G}'_2$ satisfy 2-ad, adversaries cannot re-identify any user in these KGs if they gain access to only one of them. However, by associating $\overline{G}'_0$, $\overline{G}'_1$, and $\overline{G}'_2$, they can re-identify *user:0*, *user:1*, *user:2*, *user:3*, and *user:4*. First, if they know that $Jane$ exists in $\overline{G}'_1$ and does not exist in $\overline{G}'_0$, they can identify that *user:4* is $Jane$, as she is the only new user. Moreover, they can also identify that *user:0* in $\overline{G}'_1$ is Ken, if they know that Ken has just changed his job from $Student$ to $Engineer$ at time 1. In addition, they can detect that *user:3* in $\overline{G}'_1$ is Tom, as he is the only one who has been removed in $\overline{G}'_1$. Finally, they can figure out that *user:3* in $\overline{G}'_2$ is Tom, as there is only one user who has been re-inserted in $\overline{G}'_2$.

In Chapter 2, we showed that the previous anonymization solutions for relational data [3,8,55,63], undirected graphs [28,37,39,50] and directed graphs [60] cannot protect users' sensitive information when adversaries have access to different versions of anonymized KGs. To cope with this issue, in this chapter, we present the k^w-Time-Varying Attribute Degree (k^w-tad), an extension of k-ad, to protect users from being re-identified even if adversaries gain access to w continuous anonymized KGs. We extend k-ad because it is the only protection model that can protect users' identities in anonymized KGs. In particular, we assume that adversaries can know if a victim is in an anonymized KG and also his/her attributes' values and relationships' out/in-degrees in these KGs. k^w-tad protects any user



(a) G_0 (b) G_1

(c) G_2

Figure 5.1: Different snapshots of a KG at time $t = 0, 1, 2$.

appearing at least once in w continuous anonymized KGs by ensuring that the series of his/her attributes' values and relationships' out-/in-degree in these KGs is indistinguishable from that of $k-1$ other users who also appear at least once in these KGs. Then, the adversaries cannot infer the users' identities with a confidence higher than $\frac{1}{k}$ even if they gain access to w continuous anonymized KGs.

Example 14. Figure. 5.3 illustrates the 2^3-tad anonymized versions of G_0, G_1, and G_2 presented in Figure. 5.1. Here $\overline{G}_0$ is identical to $\overline{G}'_0$ (Figure. 5.2a), as G_0 is the first version of the original KG. By adding the fake user *fake:0* to $\overline{G}_1$ and make its *job* and out-degree of *classmate_of* relationship identical to those of *user:4*, adversaries cannot identify which node is *Jane*. Moreover, as *job* of both *user:0* and *user:1* in $\overline{G}_1$ is *Student* or *Engineer*, the adversaries cannot detect which node is *Ken*. Because *age*, *job*, and in-degree of *follows* relationship in $\overline{G}_0$ of *user:2* and *user:3* are identical, and both of them are removed in $\overline{G}_1$, the adversaries also cannot identify which node is *Tom* in $\overline{G}_1$. Furthermore, as both of them are re-inserted in $\overline{G}_2$, the adversaries cannot identify which node is *Tom* in $\overline{G}_2$. For each user appearing at least once in $\overline{G}_0$, $\overline{G}_1$, and $\overline{G}_2$, the series of his/her attributes' values and out-/in-degrees of relationships in these KGs is indistinguishable from that of another user who also appear in these KGs. Then, the adversaries cannot re-identify any user with a confidence higher than $\frac{1}{2}$.

Moreover, we present the Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm (CTKGA) to anonymize the current version of the original KG such that the series of the current and $w-1$ previous anonymized KGs satisfies k^w-tad. First, the proposed algorithm splits users in the original KG to clusters such that the series of attributes' values and relationships' out-/in-degrees in $w-1$ previous anonymized KGs of

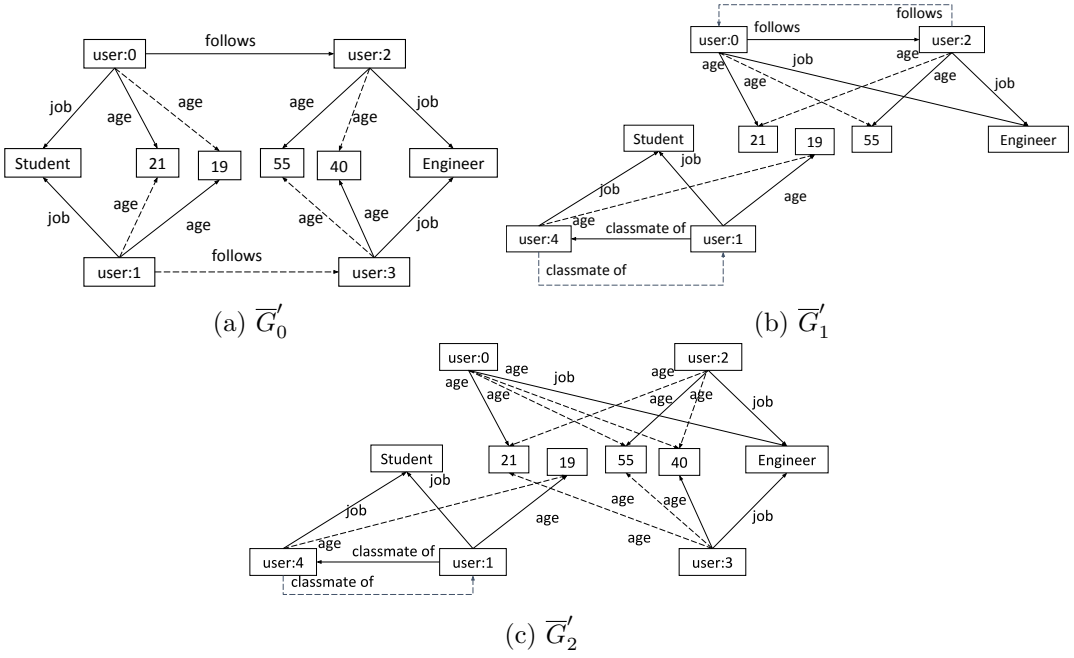


Figure 5.2: 2-ad anonymized versions of G_0, G_1, and G_2.

users who are in the same clusters is identical and all of these clusters have at least k users. Also, it adds fake users to ensure that all inserted users, who exist in the current original KG but do not exist in previous anonymized KGs, are in clusters that have at least k users. Furthermore, it deletes some users to ensure that for each deleted user, who exists in previous anonymized KGs and does not exist in the current original KG, the series of his/her attributes' values and relationships' out-/in-degrees in $w-1$ previous anonymized KGs is indistinguishable from that of $k-1$ other deleted users. Also, our algorithm ensures that all of the resulting clusters have from k to $2\times k-1$ users by splitting those that have more than $2\times k-1$ users. Finally, we use the Knowledge Graph Generalization Algorithm (KGG) (Chapter 4) to make the attributes' values and relationships' out-/in-degrees of users who are in the same cluster identical. Additionally, we prove that even if data providers update their original KG by inserting, re-inserting, and deleting users or updating edges of their

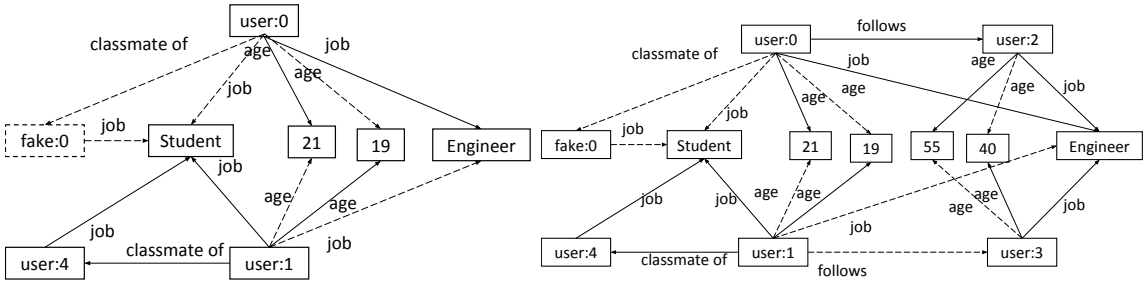


Figure 5.3: 2^3-tad versions of G_1 and G_2 ($\overline{G}_0 = \overline{G}'_0$).

users, our algorithm can anonymize the original KG such that the series of w continuous anonymized KGs always satisfies k^w-tad.

As our algorithm requires to know attributes' values and relationships' out-/in-degrees of users appearing at least once in $w-1$ previous anonymized KGs, the running time of our algorithm will increase as w increases. Therefore, we propose the Attribute Degree Sequence Table, an extension of the Cluster Sequence Table [50], to store not only degrees of a single relationship type, but also attributes' values and out-/in-degrees of many relationship types in $w-1$ continuous anonymized KGs. We conduct experiments on six real-life datasets to evaluate the quality of anonymized KGs generated by our algorithm and compare our work with previous algorithms in Chapter 3, Chapter 4, and [10].

The remaining sections of this chapter are organized as follows. Section 5.2 illustrates our privacy protection model. Then, Section 5.3 presents our anonymization algorithm. In Section 5.4, we evaluate the efficiency of our approach.

5.2 Identity Protection in Sequential Publishing of Knowledge Graphs

As users' data change over time, a data provider may hold many versions of their data. We model each version as a distinct knowledge graph (KG). Since Chapter 4 does not consider timestamp of each KG, we reformalize notations for a KG at a specific timestamp. Specifically, we model a KG generated at time t as $G_t(V_t, E_t, R_t)$, where V_t is the set of nodes, E_t is the set of edges connecting these nodes, and R_t is the set of relationship types at time t. As a node in a KG illustrates either a user or an attribute' value, the set V_t contains two subsets: the set $V_t^U \subseteq V_t$, modelling users (e.g., *user:0*, *user:1*), and the set $V_t^A \subseteq V_t$, representing attributes' values (e.g., *18*, *Student*). Similarly, the set R_t contains two subsets: the set of user-to-user relationship types R_t^{UU}, representing users' relationships (e.g., $follows$), and the set of user-to-attribute relationship types R_t^{UA}, modelling users' attributes (e.g., age, job). Each edge $e \in E_t$ is defined as a triple (u, r, v), where $u, v \in V_t$ and $r \in R_t$. We denote with $E_t^{UA} \subseteq E_t$ those $e = (u, r_a, v_a)$, such that $r_a \in R_t^{UA}$. Similarly, we denote with E_t^{UU} those $e = (u, r_u, v_u)$, such that $r_u \in R_t^{UU}$.

Let u be a user in G_t. We denote with $I_a(G_t, u) = \{(r_a, v_a) | (u, r_a, v_a) \in E_t^{UA}\}$ the values of u's attributes in G_t. Let $d_o(G_t, r_u, u) = |\{(u, r_u, v) \in E_t^{UU}\}|$ and $d_i(G_t, r_u, u) = |\{(v, r_u, u) \in E_t^{UU}\}|$ be the out- and in-degree of relationship type $r_u \in R_t^{UU}$ of u. We denote with $I_o(G_t, u) = \{(r_u, d_o(G_t, r_u, u)) | r_u \in R_t^{UU}\}$, and $I_i(G_t, u) = \{(r_u, d_i(G_t, r_u, u)) | r_u \in R_t^{UU}\}$ the out- and in-degrees of u in G_t, respectively. Then, we denote with $I(G_t, u) = (I_a(G_t, u), I_o(G_t, u), I_i(G_t, u))$ the information of u in G_t.

Appendix A summarizes the main notations used in the paper.

5.2.1 Adversary Background Knowledge

At each publication time t, a data provider releases a KG G_t that evolves from G_{t-1} by adding/removing some of its edges/nodes. The data provider anonymizes G_t and publishes

its anonymized version $\overline{G}_t(\overline{V}_t, \overline{E}_t, \overline{R}_t)$.

In Chapter 4, we use k-ad to protect the identity of any user u in $\overline{G}_t$. In particular, we denote with $\mathcal{BK}_t(u)$ the background knowledge at time t about u of an adversary. $\mathcal{BK}_t(u)$ contains all of the attributes' values and relationships at time t that the adversary knows about user u in real-life. We proved that the adversary is not able to re-identify user u by using the background knowledge about u at time t (e.g., $\mathcal{BK}_t(u)$) and the attributes' values and out-/in-degrees extracted from $\overline{G}_t$ (e.g., $I(\overline{G}_t, v)$) with a confidence higher than $\frac{1}{k}$, in Chapter 4. We illustrate the attributes' values and relationships' out-/in-degrees that the adversary can extract from an anonymized KG in the following example.

Example 15. By using $\overline{G}'_0$ (Figure. 5.2a), adversaries can extract the following information about *user:2* and *user:3*: their attributes' values $I_a(\overline{G}'_0, \textit{user:2}) = I_a(\overline{G}'_0, \textit{user:3})$ = {(*age*, 40), (*age*, 55), (*job*, *Engineer*)}, out-degree information $I_o(\overline{G}'_0, \textit{user:2}) = I_o(\overline{G}'_0, \textit{user:3}) = \{(follows, 0)\}$, and in-degree information $I_i(\overline{G}'_0, \textit{user:2}) = I_i(\overline{G}'_0, \textit{user:3})$ = $\{(follows, 1)\}$. Then, $I(\overline{G}'_0, \textit{user:2}) = I(\overline{G}'_0, \textit{user:3})$ = ({(*age*, 40), (*age*, 55), (*job*, *Engineer*)}, {(*follows*, 0)}, {(*follows*, 1)}).

Unfortunately, if an adversary gains access not only to $\overline{G}_t$ but also to the $w-1$ previous anonymized KGs $\overline{G}_{t-w+1}, \ldots, \overline{G}_{t-1}$, he/she can potentially re-identify any user u appearing at least once in these KGs. We formally define these users as follows:

Definition 18 (Historical Users Set)**.** *Let $\overline{g}_t^w$ = ($\overline{G}_{t-w+1}$, $\overline{G}_{t-w+2}$, ..., $\overline{G}_t$) be a series of w continuous k-ad anonymized KGs that a data provider has published. The Historical Users Set of $\overline{g}_t^w$, denoted as $\mathcal{U}(\overline{g}_t^w)$, is the set of users appearing at least once in these KGs. Formally:*

$$\mathcal{U}(\overline{g}_t^w) = \bigcup_{t-w+1 \leq i \leq t} \overline{V}_i^U$$

Example 16. Let $\overline{g}_2^3$ = ($\overline{G}_0$, $\overline{G}_1$, $\overline{G}_2$) be the series of anonymized KGs, shown in Figure. 5.3. The Historical Users Set of the series is: $\mathcal{U}(\overline{g}_2^3)$ = {*user:0*, *user:1*, *user:2*, *user:3*, *user:4*, *fake:0*}.

An adversary can re-identify a user u in $\mathcal{U}(\overline{g}_t^w)$ by combining the background knowledge that he/she knows about u from times $t-w+1$,$t-w+2$,...,t and the series of u's attributes' values and relationships' out-/in-degrees that he/she can extract from anonymized KGs in $\overline{g}_t^w$. We formally define the adversary knowledge as follows:

Definition 19 (Adversary Knowledge on a series of KGs)**.** *Let $\overline{g}_t^w$ = ($\overline{G}_{t-w+1}$, $\overline{G}_{t-w+2}$, ..., $\overline{G}_t$) be a series of w continuous k-ad anonymized KGs that a data provider has published. The knowledge that an adversary can use to re-identify a user $u \in \mathcal{U}(\overline{g}_t^w)$ contains:*

- *The series of background knowledge that the adversary knows about user u: $\mathcal{BK}_t^w(u)$ = ($\mathcal{BK}_{t-w+1}(u)$, $\mathcal{BK}_{t-w+2}(u)$,..., $\mathcal{BK}_t(u)$).*
- *The series of attributes' values and relationships' out-/in-degrees that the adversary can extract from $\overline{g}_t^w$: $\mathcal{I}(\overline{g}_t^w, u)$ = ($I(\overline{G}_{t-w+1}, u)$, $I(\overline{G}_{t-w+2}, u)$, ..., $I(\overline{G}_t, u)$).*

Example 17. The adversary can extract the series of attributes' values and out-/in-degrees in $\overline{g'}_2^3$ (Figure. 5.2), $\overline{g}_2^3$ (Figure. 5.3) of *user:3* as: $\mathcal{I}(\overline{g'}_2^3, \textit{user:3}) = (\{(age, 40), (age, 55), (job, Engineer)\}, \{(follows, 0)\}, \{(follows, 1)\}), (\varnothing, \varnothing, \varnothing), (\{(age, 21), (age, 40), (age, 55), (job, Engineer)\}, \{(follows, 0)\}, \{(follows, 0)\}))$, $\mathcal{I}(\overline{g}_2^3, \textit{user:3}) = ((\{(age, 40), (age, 55), (job, Engineer)\}, \{(follows, 0)\}, \{(follows, 1)\}), (\varnothing, \varnothing, \varnothing), (\{(age, 40), (age, 55), (job, Engineer)\}, \{(follows, 0)\}, \{(follows, 1)\}))$.

Based on the above knowledge, an adversary can re-identify his/her victim u by using an attacking mechanism $\mathcal{T}_{\overline{g}}$ to identify if a user $v \in \mathcal{U}(\overline{g}_t^w)$ is the representation of u. We formally define the attacking mechanism as follows:

Definition 20 (Attacking Mechanism to a series of KGs)**.** *Let $\overline{g}_t^w = (\overline{G}_{t-w+1}, \overline{G}_{t-w+2}, \ldots, \overline{G}_t)$ be a series of w continuous k-ad anonymized KGs that a data provider has published, $u \in \mathcal{U}(\overline{g}_t^w)$ be the user that an adversary wants to re-identify, and v be an arbitrary user in $\mathcal{U}(\overline{g}_t^w)$. The attacking mechanism $\mathcal{T}_{\overline{g}}$ is represented as*

$$\mathcal{T}_{\overline{g}}(\mathcal{BK}_t^w(u), \mathcal{I}(\overline{g}_t^w, v)) = \begin{cases} 1, \textit{if } u, v \textit{ is the same user.} \\ 0, \textit{otherwise.} \end{cases}$$

We define the Privacy Disclosure Risk based on the attacking mechanism $\mathcal{T}_{\overline{g}}$ as follows:

Definition 21 (Privacy Disclosure Risk in a series of KGs)**.** *Let $\overline{g}_t^w = (\overline{G}_{t-w+1}, \overline{G}_{t-w+2}, \ldots, \overline{G}_t)$ be a series of w continuous k-ad anonymized KGs that a data provider has published, and $u \in \mathcal{U}(\overline{g}_t^w)$ be the user that an adversary want to re-identify. The Privacy Disclosure Risk of u is the confidence that an adversary can re-identify u by using his/her background knowledge and attacking mechanism $\mathcal{T}_{\overline{g}}$:*

$$risk(\mathcal{T}_{\overline{g}}, \overline{g}_t^w, u) = \frac{1}{\sum\limits_{v \in \mathcal{U}(\overline{g}_t^w)} \mathcal{T}_{\overline{g}}(\mathcal{BK}_t^w(u), \mathcal{I}(\overline{g}_t^w, v))}$$

Example 18. Let assume that an adversary has access to $\overline{g'}_2^3$ (Figure. 5.2). As the series of *user:0*'s attributes' values and out-/in-degrees in $\overline{g'}_2^3$: $\mathcal{I}(\overline{g'}_2^3, \textit{user:0})$ is unique, the Privacy Disclosure Risk of *Ken* is $risk(\mathcal{T}, \overline{g'}_2^3, Ken) = \frac{1}{1} = 1$. Similarly, the Privacy Disclosure Risk of all remaining users in $\mathcal{U}(\overline{g'}_2^3)$ is 1.

5.2.2 Protection Model

We present k^w-Time-Varying Attribute Degree (k^w-tad) to protect users from being re-identified with a confidence higher than $\frac{1}{k}$ even if an adversary combines attributes' values and out-/in-degrees of these users in w continuous anonymized KGs. Specifically, let $\overline{g}_t^w$ be a series of w continuous k-ad anonymized KGs. We ensure that, for every user $u \in \mathcal{U}(\overline{g}_t^w)$, there are at least $k-1$ other users $v \in \mathcal{U}(\overline{g}_t^w)$ whose series of attributes' values and out-/in-degrees in $\overline{g}_t^w$ (e.g., $\mathcal{I}(\overline{g}_t^w, v)$) is equal to that of u (e.g., $\mathcal{I}(\overline{g}_t^w, u)$). Then, for all users $u \in \mathcal{U}(\overline{g}_t^w)$, the Privacy Disclosure Risk of u is at most $\frac{1}{k}$. We formally define k^w-tad as follows:

Definition 22 (k^w-Time-Varying Attribute Degree)**.** *Let $\overline{g}_t^w$ be a series of w continuous k-ad anonymized KGs that a data provider has published. $\overline{g}_t^w$ satisfies k^w-Time-Varying Attribute Degree (k^w-ad), if and only if, for every user $u \in \mathcal{U}(\overline{g}_t^w)$, there exists a subset of users in $\mathcal{U}(\overline{g}_t^w)$, denoted as $\mathcal{C}(\overline{g}_t^w, u)$, such that $\mathcal{C}(\overline{g}_t^w, u) = \{v \in \mathcal{U}(\overline{g}_t^w) | \mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)\}$ and $|\mathcal{C}(\overline{g}_t^w, u)| \geq k$.*

Example 19. We have $\mathcal{C}(\overline{g}_2^3, \textit{user:0}) = \mathcal{C}(\overline{g}_2^3, \textit{user:1}) = \{\textit{user:0}, \textit{user:1}\}$, $\mathcal{C}(\overline{g}_2^3, \textit{user:2}) = \mathcal{C}(\overline{g}_2^3, \textit{user:3}) = \{\textit{user:2}, \textit{user:3}\}$, and $\mathcal{C}(\overline{g}_2^3, \textit{user:4}) = \mathcal{C}(\overline{g}_2^3, \textit{fake:0}) = \{\textit{user:4}, \textit{fake:0}\}$. As for every user $u \in \mathcal{U}(\overline{g}_2^3)$, $|\mathcal{C}(\overline{g}_2^3, u)| \geq 2$, $\overline{g}_2^3$ satisfies 2^3-tad.

If a series of KGs $\overline{g}_t^w$ satisfies k^w-tad, the Privacy Disclosure Risk of all users in $\mathcal{U}(\overline{g}_t^w)$ is at most $\frac{1}{k}$.

Theorem 8. Let $\overline{g}_t^w$ be a series of w continuous k-ad anonymized KGs that an adversary has access to and $\mathcal{T}_{\overline{g}}$ be an attacking mechanism that he/she uses to re-identify users in $\mathcal{U}(\overline{g}_t^w)$. If $\overline{g}_t^w$ satisfies k^w-tad, for every user $u \in \mathcal{U}(\overline{g}_t^w)$, $risk(\mathcal{T}_{\overline{g}}, \overline{g}_t^w, u) \leq \frac{1}{k}$.

Proof. Suppose $\overline{g}_t^w$ satisfies k^w-tad. Let u be an arbitrary user in $\mathcal{U}(\overline{g}_t^w)$. According to Definition 22, there is a set $\mathcal{C}(\overline{g}_t^w, u) = \{v \in \mathcal{U}(\overline{g}_t^w) | \mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)\}$ and $|\mathcal{C}(\overline{g}_t^w, u)| \geq k$. Then, for every user $v \in \mathcal{C}(\overline{g}_t^w, u)$, $\mathcal{T}_{\overline{g}}(\mathcal{BK}_t^w(u), \mathcal{I}(\overline{g}_t^w, v)) = 1$. Thus, $risk(\mathcal{T}_{\overline{g}}, \overline{g}_t^w, u) = \frac{1}{\sum_{v \in \mathcal{U}(\overline{g}_t^w)} \mathcal{T}_{\overline{g}}(\mathcal{BK}_t^w(u), \mathcal{I}(\overline{g}_t^w, u))} \leq \frac{1}{k}$. Since u is arbitrary, we can conclude that if $\overline{g}_t^w$ satisfies k^w-tad, for every user $u \in \mathcal{U}(\overline{g}_t^w)$, $risk(\mathcal{T}_{\overline{g}}, \overline{g}_t^w, u) \leq \frac{1}{k}$. □

For instance, with reference to Example 19, the Privacy Disclosure Risk of all users in $\mathcal{U}(\overline{g}_2^3)$ is $risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{user:0}) = risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{user:1}) = risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{user:2}) = risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{user:3}) = risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{user:4}) = risk(\mathcal{T}_{\overline{g}}, \overline{g}_2^3, \textit{fake:0}) = \frac{1}{2}$.

5.3 Algorithm

5.3.1 Overview

Our Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm (CTKGA) aims at modifying the structure of the original KG such that users' identities in its anonymized version are protected according to k^w-tad. The algorithm relies on two global parameters, that is, k, w. More precisely, let $\overline{g}_{t-1}^{w-1} = (\overline{G}_{t-w+1}, \ldots, \overline{G}_{t-1})$ be the series of $w-1$ previous anonymized KGs, generated by CTKGA. Given a KG G_t at time t, CTKGA generates the anonymized version of G_t: $\overline{G}_t$ according to three main steps:

1. **Clusters generation.** This step aims at generating a set of clusters $\mathcal{C}_t$ containing users in G_t such that all of these clusters have at least k users and the series of attributes' values and relationships' out-/in-degrees in all KGs in $\overline{g}_{t-1}^{w-1}$ of users who are in the same cluster is identical. Moreover, it deletes some users to ensure that for each deleted user who is not in $\mathcal{C}_t$ but was in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$, there are at least $k-1$ other deleted users, that is, users who also are not in $\mathcal{C}_t$ but were in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$ such that their series of attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$ is identical. In case $\mathcal{C}_t$ contains new users, this step also adds fake users to ensure that there are

id	information
user:0	$I(\overline{G}_0, user{:}0)$
user:1	$I(\overline{G}_0, user{:}1)$
user:2	$I(\overline{G}_0, user{:}2)$
user:3	$I(\overline{G}_0, user{:}3)$

(a) $\mathcal{H}_0^2$

id	information	
user:0	$I(\overline{G}_0, user{:}0)$	$I(\overline{G}_1, user{:}0)$
user:1	$I(\overline{G}_0, user{:}1)$	$I(\overline{G}_1, user{:}1)$
user:2	$I(\overline{G}_0, user{:}2)$	$\varnothing$
user:3	$I(\overline{G}_0, user{:}3)$	$\varnothing$
user:4	$\varnothing$	$I(\overline{G}_1, user{:}4)$
fake:0	$\varnothing$	$I(\overline{G}_1, fake{:}0)$

(b) $\mathcal{H}_1^2$

id	information	
user:0	$I(\overline{G}_1, user{:}0)$	$I(\overline{G}_2, user{:}0)$
user:1	$I(\overline{G}_1, user{:}1)$	$I(\overline{G}_2, user{:}1)$
user:2	$\varnothing$	$I(\overline{G}_2, user{:}2)$
user:3	$\varnothing$	$I(\overline{G}_2, user{:}3)$
user:4	$I(\overline{G}_1, user{:}4)$	$I(\overline{G}_2, user{:}4)$
fake:0	$I(\overline{G}_1, fake{:}0)$	$I(\overline{G}_2, fake{:}0)$

(c) $\mathcal{H}_2^2$

Figure 5.4: The ADS-Table corresponding to $\overline{G}_0$, $\overline{G}_1$, and $\overline{G}_2$ ($w = 2$).

at least k new/fake users who are in $\mathcal{C}_t$ but were not in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$. Therefore, these new/fake users are in clusters that have at least k new/fake users. To improve the performance of this step, we leverage on a data structure, called Attribute Degree Sequence Table (ADS-Table) (see Section 5.3.2 for more details), to efficiently retrieve attributes' values and relationships' out-/in-degrees of users in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$.

2. **Knowledge graph generalization.** In this step, we use KGG, the Knowledge Graph Generalization Algorithm described in Chapter 4, to generate $\overline{G}_t$, the anonymized version of G_t, by adding and removing edges in G_t such that attributes' values and relationships' out-/in-degrees of users in $\overline{G}_t$ who are in the same cluster are identical.

3. **ADS-Table update.** Our algorithm extracts attributes' values and relationships' out-/in-degrees of users in the resulting anonymized KG $\overline{G}_t$ to update the current content of the ADS-Table.

We present the Attribute Degree Sequence Table in Section 5.3.2 and clusters generation in Section 5.3.3. We refer the interested reader to Chapter 4 for more details on knowledge graph generalization.

5.3.2 Attribute Degree Sequence Table

The Attribute Degree Sequence Table (ADS-Table), denoted $\mathcal{H}_t^w$, is a table containing two columns: *user id*, containing ids of users in $\mathcal{U}(\overline{g}_t^w)$, and *information*, consisting of the array of attributes' values and degrees of users in these KGs, that is, $\mathcal{I}(\overline{g}_t^w, u)$, $\forall u \in \mathcal{U}(\overline{g}_t^w)$.

ADS-Table $\mathcal{H}_t^w$ increases the performance of our algorithm by allowing it to efficiently extract information on attributes' values and out-/in-degrees of users in w previous

anonymized KGs, using their ids. Moreover, we can generate an ADS-Table $\mathcal{H}_t^w$ from $\overline{G}_t$ and $\mathcal{H}_{t-1}^w$ instead of extracting users' attributes' values and relationships' out-/in-degrees in $w-1$ previous anonymized KGs. Figure. 5.4 illustrates the ADS-Tables $\mathcal{H}_0^2$, $\mathcal{H}_1^2$, and $\mathcal{H}_2^2$ corresponding to anonymized KGs $\overline{G}_0, \overline{G}_1, \overline{G}_2$ in Figure. 5.3.

5.3.3 Clusters Generation

All clustering algorithms relies on a notion of distance between users, to group users into clusters such that closer users are grouped into the same cluster. As our algorithm makes attributes' values and relationships' out-/in-degrees of users in the same clusters identical in the next step, we must minimize the information loss of making the attributes' values and relationships' out-/in-degrees of users in the same cluster. Therefore, as distance between users, we make use of an information loss metric which measures information loss of making the attributes' values and relationships' out-/in-degrees of two users identical. To be as general as possible, our algorithm allows data providers to use any information loss metric, such as the Attribute and Degree Information Loss Metric (ADM) (Definition 15), or the Attribute Truthfulness and Degree Information Loss Metric (Definition 17) to calculate the distance between pairs of users in their KGs.

We use a distance matrix $\mathcal{D}_t$ to store the distance between pairs of users in V_t^U, which is given as a parameter to our cluster generation algorithm.

Moreover, to increase the generality of our approach, our algorithm allows data providers to use any clustering algorithm $\mathcal{A}$ that supports distance matrix to generate clusters. However, clusters returned from $\mathcal{A}$ can have less than k users, which violates k^w-tad, or too many users, which decreases the quality of the anonymized KG.

To cope with this issue, we present the k-Medoids Partition strategy (KP) (see Algorithm 6). The strategy exploits k-Medoids [14] to split clusters that have more than or equal to $2 \times k$ users such that the number of users in all of the resulting clusters is between k and $2 \times k - 1$. Moreover, it assigns users in clusters that have less than k users to their nearest clusters that have at least k users.

Given a KG G_t at time t, the distance matrix $\mathcal{D}_t$ containing the distance between pairs of users in G_t, the ADS-Table $\mathcal{H}_{t-1}^{w-1}$; a clustering algorithm $\mathcal{A}$, and two positive numbers: k, w, we aim at generating a set of clusters that have at least k users and such that users in the same cluster have the same attributes' values and out-/in-degrees for all relationships in $w-1$ previous anonymized KGs.

Our cluster generation approach is described by Algorithm 5. The algorithm takes as input the original KG G_t at time t, two positive integers: k, w, the clustering algorithm $\mathcal{A}$, the distance matrix $\mathcal{D}_t$, and the ADS Table $\mathcal{H}_{t-1}^{w-1}$. At the beginning, it initializes $\mathcal{U}^{w-1}$ with the set of users who existed in any of the $w-1$ previous anonymized KGs $\overline{G}_{t-w+2}, \ldots, \overline{G}_{t-1}$(line 1). Then, it generates the set of users who existed in both V_t^U and any of the $w-1$ previous anonymized KGs, denoted as $\mathcal{U}^w$ (line 2). Next, it calls function *generate_clusters*() to find clusters of users in $\mathcal{U}^w$, denoted as $\mathcal{C}^w$, such that the series of attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$ of those in the same clusters is identical (line 3). This function puts new users who are in V_t^U but were not in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$

Algorithm 5 Clusters Generation($G_t, \mathcal{D}_t, \mathcal{H}_{t-1}^{w-1}, \mathcal{A}, k, w$)

Input: G_t: the original KG at time t; $\mathcal{D}_t$: the distance matrix of users in G_t; $\mathcal{H}_{t-1}^{w-1}$: the ADS-Table at time $t-1$; $\mathcal{A}$: the clustering algorithm; k, w: two positive numbers.
Output: The set of clusters $\mathcal{C}_t$.

1: $\mathcal{U}^{w-1} \leftarrow \mathcal{U}(\overline{g}_{t-1}^{w-1})$
2: $\mathcal{U}^{w} \leftarrow V_t^U \cup \mathcal{U}^{w-1}$
3: $\mathcal{C}^w \leftarrow generate\_clusters(\mathcal{H}_{t-1}^{w-1}, \mathcal{U}^w)$
4: $\mathcal{C}_t \leftarrow \varnothing$
5: **for** $c \in \mathcal{C}^w$ **do**
6: $\quad \mathcal{U}_c \leftarrow c \cap V_t^U$
7: $\quad \mathcal{U}_d \leftarrow c \setminus V_t^U$
8: $\quad \mathcal{C}_c \leftarrow \varnothing$
9: $\quad$ **if** $0 < |\mathcal{U}_d| < k$ **then**
10: $\quad\quad remove\_users(\mathcal{U}_c, k - |\mathcal{U}_d|)$
11: $\quad$ **end if**
12: $\quad$ **if** $|\mathcal{U}_c| \geq k$ **then**
13: $\quad\quad \mathcal{C}_c \leftarrow$ run clustering algorithm $\mathcal{A}$ with $\mathcal{D}_t$ and $\mathcal{U}_c$
14: $\quad\quad \mathcal{C}_c \leftarrow$ run $KP\_strategy$ with $\mathcal{D}_t$ and $\mathcal{C}_c$
15: $\quad$ **else**
16: $\quad\quad$ **if** c is the cluster of new users **then**
17: $\quad\quad\quad \mathcal{C}_c \leftarrow add\_fake\_users(\mathcal{U}_c, k - |\mathcal{U}_c|)$
18: $\quad\quad$ **end if**
19: $\quad$ **end if**
20: $\quad \mathcal{C}_t \leftarrow \mathcal{C}_t \cup \mathcal{C}_c$
21: **end for**
22: **return** $\mathcal{C}_t$

in the same cluster. It then initializes the set of clusters $\mathcal{C}_t$ with the empty set (line 4). Next, for each cluster c in $\mathcal{C}^w$, it finds the set of users who exist in both c and V_t^U, denoted as $\mathcal{U}_c$, (line 6), and the set of users in c who have been removed from G_t, denoted as $\mathcal{U}_d$ (line 7). It then initializes the set of clusters $\mathcal{C}_c$ with the empty set (line 8). If $\mathcal{U}_d$ has from 1 to $k-1$ users, it calls function *remove_users()* to randomly remove $k - |\mathcal{U}_d|$ users in $\mathcal{U}_c$ (line 10). In this way, we can ensure that for each deleted user who is not in V_t^U but was in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$, his/her series of attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$ is indistinguishable from that of $k-1$ other deleted users. If $\mathcal{U}_c$ has at least k users, it uses the clustering algorithm $\mathcal{A}$ to split users in $\mathcal{U}_c$ into clusters and assign these clusters to $\mathcal{C}_c$(line 13). Our algorithm calls Algorithm 6 to split clusters in $\mathcal{C}_c$ into clusters that have from k to $2 \times k - 1$ users (line 14). If $\mathcal{U}_c$ has less than k users and c is the group of new users, it calls function $add\_fake\_users()$ to generate $\mathcal{C}_c$ by adding $k - |\mathcal{U}_c|$ fake users to $\mathcal{U}_c$ (line 17). The function generates fake users by assigning a unique id to each fake user. Then, it generates a cluster containing these fake users and users in $\mathcal{U}_c$ and adds the

Function 3 *generate_clusters* ($\mathcal{H}_{t-1}^{w-1}$,$\mathcal{U}^w$)

1: Let H be an empty hash table.
2: **for** $u \in \mathcal{U}^w$ **do**
3: $\quad\mathcal{I}(\overline{g}_{t-1}^{w-1}, u) \leftarrow get\_user\_info(\mathcal{H}_{t-1}^{w-1}, u)$
4: $\quad\mathcal{K} \leftarrow hash(\mathcal{I}(\overline{g}_{t-1}^{w-1}, u))$
5: $\quad$**if** $\mathcal{K} \notin H$ **then**
6: $\quad\quad c(\mathcal{K}) \leftarrow \varnothing$
7: $\quad\quad append\_cluster(H, \mathcal{K}, c(\mathcal{K}))$
8: $\quad$**else**
9: $\quad\quad c(\mathcal{K}) \leftarrow get\_by\_key(H, \mathcal{K})$
10: $\quad$**end if**
11: $\quad c(\mathcal{K}) \leftarrow c(\mathcal{K}) \cup \{u\}$
12: **end for**
13: **return** *get_values*(H)

cluster to $\mathcal{C}_c$. We only add fake users when c is the cluster of new users because both fake and new users do not have any information in $\overline{g}_{t-1}^{w-1}$. By doing this, we can ensure that for each new/fake user, there are at least $k-1$ other new/fake users that were not present in $\overline{g}_{t-1}^{w-1}$ and are now in the same cluster. The generated clusters $\mathcal{C}_c$ are then added to $\mathcal{C}_t$ (line 20). Finally, our algorithm returns the set of clusters $\mathcal{C}_t$ (line 22).

Function *generate_clusters()*. Given an ADS-Table $\mathcal{H}_{t-1}^{w-1}$ and the set of users $\mathcal{U}^w$, this function assigns to the same cluster users whose series of attributes' values and relationships' out-/in-degrees in $\mathcal{H}_{t-1}^{w-1}$, is identical. It exploits a hash table which maps the series of attributes' values and relationships' out-/in-degrees to the cluster of users who have that series. Furthermore, to improve the performance of finding the cluster by the series, we use MD5 hash function to calculate a unique hash value for each series. Then, instead of using the series, we use its hash to associate the series with the clusters of users who have it in $\mathcal{H}_{t-1}^{w-1}$. In particular, this function first initializes an empty hash table H (line 1). Then, for each user u in $\mathcal{U}^w$, it retrieves u's attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$, denoted as $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u)$, by looking for u's id in $\mathcal{H}_{t-1}^{w-1}$ (line 3). It then uses MD5 to generate the hash of $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u)$, denoted as $\mathcal{K}$ (line 4). If $\mathcal{K}$ is not in H, it generates an empty cluster, denoted as $c(\mathcal{K})$, (line 6) and adds $c(\mathcal{K})$ to H (line 7). Otherwise, it gets the cluster $c(\mathcal{K})$ in H by using the key $\mathcal{K}$ (line 9). Then, it adds u to $c(\mathcal{K})$ (line11). Finally, it returns the clusters in H (line 13).

Algorithm 6 implements our strategy to split clusters. At the beginning, it finds the set of users whose clusters have less than k users, denoted as U, (line 1) and clusters that have at least k users, denoted as C_{valid} (line 2). Then, it calls function *assign_new_clusters*() to assign each user in U to his/her nearest cluster in C_{valid} (line 3). Next, it initializes the set of final clusters C_{final} to the empty set (line 4). For each cluster c in C_{valid}, if c has at least $2 \times k$ users, it uses the Same Size k-Medoids algorithm to split users in c into a set of $|c|/k$ clusters that have from k to $2 \times k - 1$ users, denoted as C_c (line 7). We create

Algorithm 6 k-Medoids Partition strategy($\mathcal{D}_t, \mathcal{C}_c, k$)

Input: $\mathcal{D}_t$: the distance matrix of users in G_t; $\mathcal{C}_c$: a set of clusters; k: a positive number.
Output: The set of clusters $\mathcal{C}_t$.

1: Let U be the set of users in C_c whose cluster has less than k users.
2: $C_{valid} \leftarrow \{c \in \mathcal{C}_c || c| \geq k\}$
3: $assign\_new\_clusters(\mathcal{D}, C_{valid}, U)$
4: $C_{final} \leftarrow \varnothing$
5: **for** $c \in C_{valid}$ **do**
6: **if** $|c| \geq 2 \times k$ **then**
7: $C_c \leftarrow run\_same\_size\_kmedoids(\mathcal{D}_t, c, |c|/k)$
8: $C_{final} \leftarrow C_{final} \cup C_c$
9: **else**
10: $C_{final} \leftarrow C_{final} \cup \{c\}$
11: **end if**
12: **end for**
13: **return** C_{final}

the Same Size k-Medoids algorithm by modifying the Balanced k-Means algorithm [38], in such a way that it uses k-Medoids instead of k-Means. Then, C_c is added to C_{final} (line 8). If c has from k to $2 \times k - 1$, Algorithm 6 adds c to C_{final} (line 10). Finally, it returns C_{final} (line 13).

Procedure *assign_new_clusters()*. Given a distance matrix $\mathcal{D}_t$, a set of clusters C_{valid}, each of which contains at least k users, and a set of users U, this procedure assigns each user u in U to his/her closest cluster in C_{valid}. The procedure initializes the smallest distance, denoted as d_{min}, with $+\infty$ (line 2) and the selected cluster, denoted as $c_{selected}$, with $\varnothing$ (line 3). Then, for each cluster c in C_{valid}, it calculates the distance between user u and cluster c, denoted as d_c, (line 5). If d_c is less than d_{min}, it assigns d_c to d_{min} (line 7) and c to $c_{selected}$ (line 8). Then, it adds u to the found closest cluster $c_{selected}$ (line 11).

5.3.4 Privacy Analysis

Given a series of anonymized KGs $\overline{g}_{t-1}^{w-1}$ satisfying k^{w-1}-tad, a KG G_t generated by adding/removing edges and nodes of G_{t-1}, and the set of clusters $\mathcal{C}_t$ generated by our algorithm, we want to prove that: (A) inserted users in G_t are in clusters that have at least k users (see Theorem 9); (B) updated users also are in clusters that have at least k users and all users in the same cluster have the same series of attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$ (see Theorem 10); (C) deleted users are not in clusters in $\mathcal{C}_t$ and for every deleted user, his/her series of attributes' values and relationships' out-/in-degrees is indistinguishable from that of $k-1$ other deleted users (see Theorem 11). In addition, if the data providers re-insert in G_t some users who have been previously deleted in $\overline{g}_{t-1}^{w-1}$, these users are in clusters that have at least k users since they are in both $\mathcal{U}(\overline{g}_{t-1}^{w-1})$ and G_t (according to Theorem 10). Additionally, if these users have been deleted in anonymized

Procedure 2 *assign_new_clusters* $(\mathcal{D}_t, C_{valid}, U)$

```
1: for u ∈ U do
2:     d_min ← +∞
3:     c_selected ← ∅
4:     for c ∈ C_valid do
5:         d_c ← max{D_t(u, v)|v ∈ c}
6:         if d_c < d_min then
7:             d_min ← d_c
8:             c_selected ← c
9:         end if
10:    end for
11:    c_selected ← c_selected ∪ {u}
12: end for
```

KGs that have been published earlier than those in $\overline{g}_{t-1}^{w-1}$, they also are in clusters that have at least k users because they are in G_t but not in $\mathcal{U}(\overline{g}_{t-1}^{w-1})$ (according to Theorem 12.

In Theorem 9, we prove the assertion (A).

Theorem 9. Given a KG G_t and an ADS-Table $\mathcal{H}_{t-1}^{w-1}$ storing users' information in $\overline{g}_{t-1}^{w-1}$. Let $\mathcal{C}_t$ be the set of clusters returned from Algorithm 5, executed with two positive numbers: k, w. Let $\overline{V}_t^U = \cup_{c \in \mathcal{C}_t} c$ be the set of users in $\mathcal{C}_t$ and $\mathcal{U}_t^i = \overline{V}_t^U \setminus \mathcal{U}(\overline{g}_{t-1}^{w-1})$ be the set of new and fake users of $\mathcal{C}_t$. For every user $u \in \mathcal{U}_t^i$, u is in a cluster in $\mathcal{C}_t$, denoted c_u, such that c_u has at least k users.

Proof. Let u be an arbitrary user in $\mathcal{U}_t^i$ and c_u be u's cluster in $\mathcal{C}_t$. Suppose c_u has less than k users. If $\mathcal{U}_t^i$ has at least k users, c_u also has at least k users that are real users as it is returned from Algorithm 6 which only returns clusters that have at least k users (lines 3, 10, Algorithm6). Otherwise, c_u also has at least k users that are both fake/real users (line 17, Algorithm 5). Therefore, c_u always has at least k users. But this contradicts to the fact that c_u has less than k users. Therefore, we can conclude that for every user $u \in \mathcal{U}_t^i$, u is in a cluster in $\mathcal{C}_t$, denoted c_u, such that c_u has at least k users. □

Theorem 10 proves assertion (B).

Theorem 10. Given a KG G_t and an ADS-Table $\mathcal{H}_{t-1}^{w-1}$ storing users' information in $\overline{g}_{t-1}^{w-1}$. Let $\mathcal{C}_t$ be the set of clusters returned from Algorithm 5, executed with two positive numbers: k, w. Let $\overline{V}_t^U = \cup_{c \in \mathcal{C}_t} c$ be the set of users in $\mathcal{C}_t$ and $\mathcal{U}_t^u = \overline{V}_t^U \cap \mathcal{U}(\overline{g}_{t-1}^{w-1})$ be the set of users existing in both $\overline{V}_t^U$ and $\mathcal{U}(\overline{g}_{t-1}^{w-1})$. If $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, for every user u in $\mathcal{U}_t^u$, u is in a cluster in $\mathcal{C}_t$, denoted c_u, such that c_u has at least k users and $\forall u', v' \in c_u$, $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$.

Proof. Suppose $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad. Let u be an arbitrary user in $\mathcal{U}_t^u$ and c_u be its cluster in $\mathcal{C}_t$. Suppose c_u has less than k users or there are two user $u', v' \in c_u$ such that $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') \neq \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$. Let c be one of the clusters returned by Function 3 and c contains u. As $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, c also has at least k users. c consists of users

whose information in $\overline{g}_{t-1}^{w-1}$ is identical (line 11, Function 3). If c has less than k real users, Algorithm 5 does not add it to $\mathcal{C}_t$ (line 12, Algorithm 5). In this case, real users in c are protected with deleted users (according to Theorem 11). If c has at least k real users, Algorithm 5 uses Algorithm 6, executed real users in c, to generate c_u. c_u has at least k users as it is returned by Algorithm 6 which only returns clusters that have from k to $2 \times k - 1$ users (lines 3,10, Algorithm 6). Moreover, $\forall u', v' \in c_u$, $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$ as c_u is a subset of c. But this contradicts the fact that c_u has less than k users or there are two user $u', v' \in c_u$ such that $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') \neq \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$. Therefore, we can conclude that if $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, for every user u in $\mathcal{U}_t^u$, u is in a cluster in $\mathcal{C}_t$, denoted c_u, such that c_u has at least k users and $\forall u', v' \in c_u$, $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$. □

Theorem 11 proves the assertion (C).

Theorem 11. Given a KG G_t and an ADS-Table $\mathcal{H}_{t-1}^{w-1}$ storing users' information in $\overline{g}_{t-1}^{w-1}$. Let $\mathcal{C}_t$ be the set of clusters returned from Algorithm 5, executed with two positive numbers: k, w. Let $\overline{V}_t^U = \cup_{c \in \mathcal{C}_t} c$ be the set of users in $\mathcal{C}_t$ and $\mathcal{U}_t^d = \mathcal{U}(\overline{g}_{t-1}^{w-1}) \setminus \overline{V}_t^U$ be the set of deleted users. If $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, for every user $u \in \mathcal{U}_t^d$, there is a set $\mathcal{C}_t^d(u) = \{v \in \mathcal{U}_t^d | \mathcal{I}(\overline{g}_{t-1}^{w-1}, u) = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v)\}$ such that $|\mathcal{C}_t^d(u)| \geq k$.

Proof. Suppose $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad and there is a user $u \in \mathcal{U}_t^d$ such that $|\mathcal{C}_t^d(u)| < k$. Let c be one of the clusters returned by Function 3 and c contains u. c consists of users whose information in $\overline{g}_{t-1}^{w-1}$ is identical (line 11, Function 3). As $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, c also has at least k users. Let $\mathcal{U}^d = \mathcal{U}_t^d \cap c$ be the set users in c who are in $\overline{V}_t^U$ and c. Here, $\mathcal{U}^d$ contains u as he/she is in both c and $\mathcal{U}_t^d$. $\mathcal{U}^d$ has at least k users as Algorithm 5 will remove real users in c if c has from 1 to $k-1$ deleted users (line 10, Algorithm 5). Moreover, $\forall u', v' \in \mathcal{U}^d$, $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u') = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v')$. Therefore, $|\mathcal{C}_t^d(u)| \geq k$. But this contradicts the fact that $|\mathcal{C}_t^d(u)| < k$. Therefore, if $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, for every user u in $\mathcal{U}_t^d$, $|\mathcal{C}_t^d(u)| \geq k$. □

In Theorem 12, we prove that if $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, $\overline{g}_t^w$ satisfies k^w-tad.

Theorem 12. Given a series of w continuous KGs $\overline{g}_t^w$, if all $\overline{G}_t \in \overline{g}_t^w$ are generated by our algorithm, then $\overline{g}_t^w$ satisfies k^w-tad.

Proof. We prove this theorem by induction on t. At time $t = 0$, the set of clusters $\mathcal{C}_t$ returned from Algorithm 5 consists of new users. Then, all clusters in $\mathcal{C}_t$ have at least k users (according to Theorem 9). By using the Knowledge Graph Generalization Algorithm (KGG) (see Chapter 4) to generate the first anonymized KG $\overline{G}_0$, we make the attributes' values and relationships' out-/in-degrees of users in the same clusters identical. Then, for every user u in $\mathcal{U}(\overline{g}_0^w)$, $|\mathcal{C}(\overline{g_0^w}, u)| \geq k$. Therefore, at time $t = 0$, $\overline{g}_0^w$ satisfies k^w-tad.

In the following paragraphs, we prove the induction step. Suppose $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad and $\overline{g}_t^w$ does not satisfy k^w-tad.

The set of users in $\overline{G}_t$, $\overline{V}_t^U$, consists of all of inserted/re-inserted/updated users. Moreover, these users are in clusters in $\mathcal{C}_t$ that have at least k users and users in the same clusters have the same series of attributes' values and relationships' out-/in-degrees in $\overline{g}_{t-1}^{w-1}$ (according to Theorem 9 and 10). Then, let c be an arbitrary cluster in $\mathcal{C}_t$ and u, v be two arbitrary users in c. c has at least k users and $\mathcal{I}(\overline{g}_{t-1}^{w-1}, u) = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v)$. As we

use KGG to make attributes' values and relationships' out-/in-degrees of users in the same clusters identical in $\overline{G}_t$, $I(\overline{G}_t, u) = I(\overline{G}_t, v)$. Thus, $\mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)$. As u, v are arbitrary, for every users $u, v \in c$, $\mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)$. Since c is arbitrary, for every cluster $c \in \mathcal{C}_t$, c has at least k users and all users in c has the same series of attributes' values and relationships' out-/in-degrees in $\overline{g}_t^w$. Then, for every user $u \in \overline{V}_t^U$, there is a subset $\mathcal{C}(\overline{g}_t^w, u) = \{v \in \overline{V}_t^U | \mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)\}$ and $|\mathcal{C}(\overline{g}_t^w, u)| \geq k$.

Let $\mathcal{U}_t^d = \overline{g}_{t-1}^{w-1} \setminus \overline{V}_t^U$ be the set of deleted users in $\overline{G}_t$. Let u be an arbitrary user in $\mathcal{U}_t^d$. According to Theorem 11, there is a set $\mathcal{C}(\overline{g}_{t-1}^{w-1}, u) = \{v \in \mathcal{U}_t^d | \mathcal{I}(\overline{g}_{t-1}^{w-1}, u) = \mathcal{I}(\overline{g}_{t-1}^{w-1}, v)\}$ and $|\mathcal{C}(\overline{g}_{t-1}^{w-1}, u)| \geq k$. Because users in $\mathcal{C}(\overline{g}_{t-1}^{w-1}, u)$ are not in $\overline{G}_t$, for every user $v \in \mathcal{C}(\overline{g}_{t-1}^{w-1}, u)$, $I(\overline{G}_t, u) = I(\overline{G}_t, v)$. Then, the set $\mathcal{C}(\overline{g}_t^w, u) = \{v \in \mathcal{U}_t^d | \mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)\}$ also has at least k users. As u is arbitrary, for every user u in $\mathcal{U}_t^d$, $\mathcal{C}(\overline{g}_t^w, u) \geq k$.

Because $\mathcal{U}(\overline{g}_t^w) = \mathcal{U}_t^d \cup \overline{V}_t^U$, for every user u in $\mathcal{U}(\overline{g}_t^w)$, there is a set $\mathcal{C}(\overline{g}_t^w, u) = \{v \in \mathcal{U}(\overline{g}_t^w) | \mathcal{I}(\overline{g}_t^w, u) = \mathcal{I}(\overline{g}_t^w, v)\}$ such that $|\mathcal{C}(\overline{g}_t^w, u)| \geq k$. Therefore, $\overline{g}_t^w$ satisfies k^w-tad. But this contradicts the fact that $\overline{g}_t^w$ does not satisfy k^w-tad. Then, we can conclude that if $\overline{g}_{t-1}^{w-1}$ satisfies k^{w-1}-tad, $\overline{g}_t^w$ also satisfies k^w-tad.

Therefore, we can conclude that if all $\overline{G}_t \in \overline{g}_t^w$ generated by our algorithm, then $\overline{g}_t^w$ satisfies k^w-tad. □

5.4 Experiments

In this section, we evaluate the quality of anonymized KGs generated by our anonymization algorithm.

5.4.1 Datasets and Settings

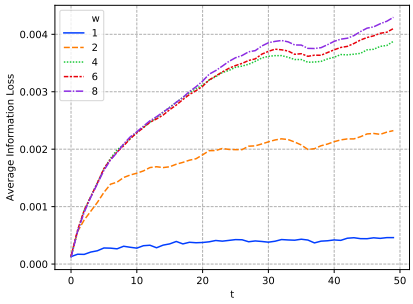
As KGs can illustrate many types of graphs, we use six real-life datasets to evaluate the effectiveness of our algorithm, namely: *Email-Eu-core* [57], *ICEWS* [16], *Yago* [16], *Email-temp* [43], *Freebase* [5], and *DBLP* [32]. Appendix B illustrates properties of these datasets.

In Chapter 4, we use many clustering algorithms to evaluate the quality of anonymized KGs generated by their Cluster-Based Knowledge Graph Anonymization (CKGA) and shows that k-means [14] gives best results. However, k-means does not support distance matrix. Therefore, we use a version of k-means, k-Medoids [14] which supports distance matrix, to run all of our experiments.

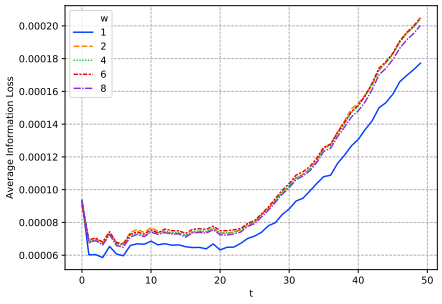
5.4.2 Tuning CTKGA

In this experiment, we aim at evaluating the effects of parameters: k, w to the average information loss of users in the anonymized KGs. The average information loss is calculated by taking the average of the information loss of users by using the Attribute and Degree Information Loss Metric (ADM) (Definition 15). ADM measures the amount of information a user loses on both his/her attributes and relationships by comparing his/her attributes'

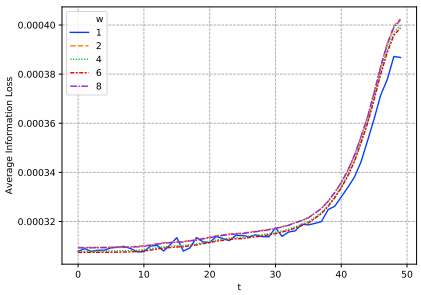
values and relationships' out-/in-degrees in the anonymized KG with those in the original one.



(a) Email-temp



(b) ICEWS

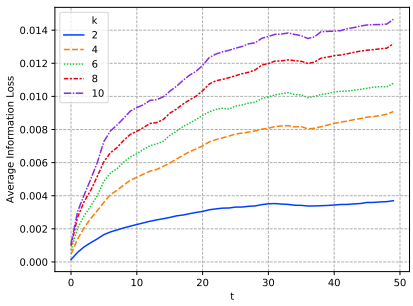


(c) Yago

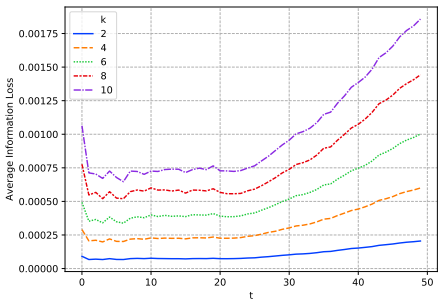
Figure 5.5: Information loss by varying w.

Effects of w. We first fix $k = 2$ to study the impact of w. Figure. 5.5 shows the average information loss of anonymized KGs generated by CTKGA by varying values of w in three datasets: *Email-temp* (Figure. 5.5a), *ICEWS* (Figure. 5.5b), and *Yago* (Figure. 5.5c). We observe that higher values of w generally lead to high information loss while the running time is not increased. In particular, when we increase w from 1 to 2, the average information loss of the last anonymized KG increases 0.0017 in *Email-temp*, 2.77×10^{-5} in *ICEWS*, and 1.58×10^{-5} in *Yago*. However, if w is high enough, the increment of the average information loss is very small. When we increase w from 4 to 6, the average information loss of the last anonymized KG increases 0.0003 in *Email-temp*, 8.89×10^{-7} in *ICEWS*, and 1.13×10^{-6} *Yago*. Furthermore, the higher MCE is, the more w affects the average information loss of anonymized KGs. Indeed, *Email-temp*'s anonymized KGs lose more information than those of *ICEWS* and *Yago* because the amount of modified information of users in *Email-temp* ($MCE = 0.9$) is higher than that of *ICEWS* ($MCE = 0.2$) and *Yago* ($MCE = 0.04$).

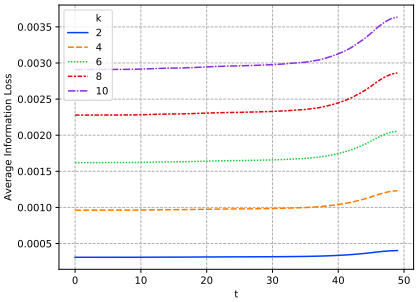
Effects of k. We select $w = 4$ for *Email-temp*, and $w = 2$ for *ICEWS* and *Yago* to observe the impact of k on the average information loss of anonymized KGs. Figure. 5.6 illustrates our results by varying k on three datasets: *Email-temp* (Figure. 5.6a), *ICEWS* (Figure. 5.6b), and *Yago* (Figure. 5.6c). Different from w, increasing k always makes the



(a) Email-temp ($w = 4$)



(b) ICEWS ($w = 2$)



(c) Yago ($w = 2$)

Figure 5.6: Information loss by varying k.

average information loss increase. By increasing k from 2 to 4, the average information loss increases 0.0053 in *Email-temp*, 0.0004 in *ICEWS*, and 0.0008 in *Yago*. Similarly, the average information loss increases 0.0015 in *Email-temp*, 0.0004 in *ICEWS*, and 0.0008 in *Yago*, when we increase k from 8 to 10. The Coefficient of Variation of these increments, calculated by dividing their standard deviation by their mean, is 0.26 when we increase k from 2 to 10 in *Email-temp*, 0.04 in *ICEWS* and 0.03 in *Yago*. Then, the more users a KG has, the less k affects the quality of its anonymized version (*Email-temp*, *ICEWS*, and *Yago* have 986, 2,617, 8,918, respectively).

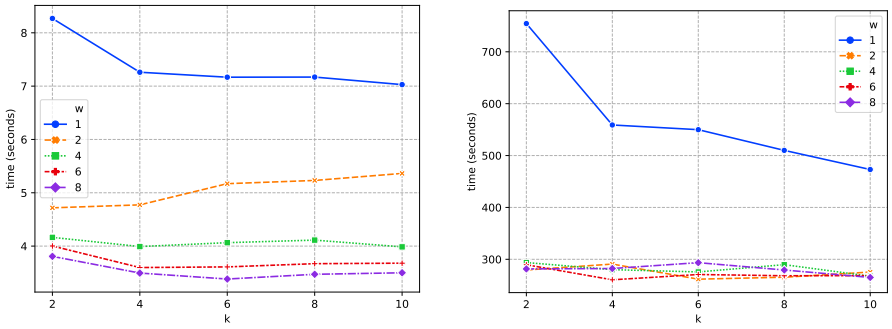
These experiments allow us to learn relevant insights that might be useful for properly setting our anonymized algorithm. Indeed, the experiment on w suggests that data providers can set w based on MCE of their KGs and the number of continuous KGs they want to consider. For instance, if we select $w \geq 4$ for *Email-temp*, $w \geq 2$ for *ICEWS*, and *Yago* the average information loss increment is very small. Additionally, the experiment on k suggests that data providers should choose k based on the number of users their KGs have and the Privacy Disclosure Risk they want.

5.4.3 Performance Evaluation

In this experiment, we evaluate the impact of w, k to performance of our Clusters Generation Algorithm (Algorithm 5). Figure. 5.7 illustrates the average running time of algorithm

on all snapshots of two datasets: *ICEWS* (Figure. 5.7a) and *Yago* (Figure. 5.7b).

The running time that our algorithm needs to generate clusters for *Yago* is higher than the one for *ICEWS* as *Yago* has more users than *ICEWS* (*ICEWS* and *Yago* have 2,617, 8,918, respectively). Moreover, the higher values of w are, the algorithm needs less time in all datasets because of two reasons. First, the series of users' attributes and relationships' out-/in-degrees in previous anonymized KGs is extracted from ADS-Table. Secondly, while our algorithm needs almost the same time to run on the first snapshot, it can save time by relying on clusters generated in the previous snapshot, which have at least k users. Thus, our algorithm does not need too much time to split these clusters. At $w = 1$, our algorithm always needs to split a big cluster containing all users in the current snapshot; therefore, its running time is higher than that of those running with $w > 1$. Furthermore, with $w = 1$, the running time of our algorithm decreases when increasing k because the resulting clusters have more users and it does not need to split these clusters too much. Additionally, w affects to the performance of our algorithm on *ICEWS* more than on *Yago* as MCE of *Yago* ($MCE = 0.04$) is less than that of *ICEWS* ($MCE = 0.2$). Therefore, ADS-Table is effective to improve the performance of our algorithm and data providers can specify w, k based on the MCE of their KGs. We can also conclude that time needed to generate the clusters is negligible, considering that this task is performed only once per data publishing.



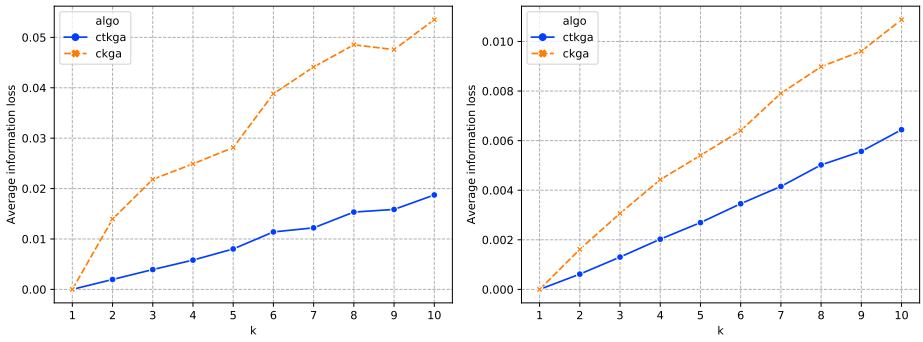
(a) ICEWS (b) Yago

Figure 5.7: Performance of our algorithm on varying values of k and w.

5.4.4 Comparative Evaluation

In this experiment, we compare the quality of anonymized KGs returned from our algorithm (*CTKGA*) with those returned from *CKGA* (Chapter 4), *CDGA* (Chapter 3), and *DGA* [10] on four datasets: *Email-Eu-core*, *Freebase*, *Email-temp*, and *DBLP*.

Figure. 5.8 illustrates the average information loss of users in anonymized KGs returned from our algorithm and *CKGA* on two datasets: *Email-Eu-core*(Figure. 5.8a) and *Freebase*(Figure. 5.8b). Our algorithm returns anonymized KGs that lose less information than those returned from *CKGA* for both datasets.

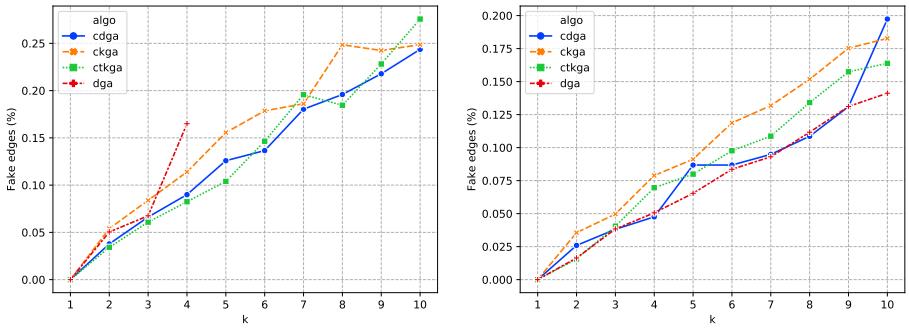


(a) Email-Eu-core (b) Freebase

Figure 5.8: Average information loss of anonymized KGs returned from our algorithm (CTKGA) and CKGA.

Similar to Chapter 4, to compare CTKGA with CDGA and DGA, we have considered the ratio of fake edges instead of information loss. Figure. 5.9 shows the ratio of fake edges of anonymized graphs returned from our algorithm and *CKGA*, *CDGA*, and *DGA* for two datasets: *Email-temp*(Figure. 5.9a) and *DBLP*(Figure. 5.9b). In both datasets, in most values of k, our algorithm adds less fake edges to generate anonymized graphs than *CKGA*. At $k = 7$ in *Email-temp* dataset, we need to add more fake edges than *CKGA* but the difference is very small (0.01%). The number of fake edges in anonymized graphs returned from our algorithm is similar to that of *CDGA* and *DGA*.

Therefore, our algorithm is effective enough to generate not only anonymized KGs but also anonymized directed graphs.



(a) Email-temp (b) DBLP

Figure 5.9: Ratio of fake edges of anonymized KGs returned from our algorithm (CTKGA) and CKGA, CDGA, and DGA.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In the era of big data, data sharing is crucial and protecting users' sensitive information in these data is as important as analyzing them [26]. Many companies start using knowledge graphs (KGs) to share their data due to their flexibility in modelling both attributes' values and relationships, Even though many work [26] have been presented to generate anonymized data (cfr. Chapter 2), they cannot be applied to anonymize KGs. Only either users' attributes or their relationships are anonymized. Therefore, in this thesis, we focus on presenting anonymization solutions for KGs containing both users' attributes and relationships.

Since KGs are directed graphs, in Chapter 2, we reviewed anonymization solutions [10, 60] for directed graphs. Although these solutions can protect users from being re-identified, their anonymization algorithms do not always generate anonymized graphs satisfying the requirements of their protection models. To cope with this issue, in Chapter 3, we presented our anonymization algorithm for directed graphs, i.e. the Cluster-Based Directed Graph Anonymization Algorithm (CDGA), and proved that CDGA always generates anonymized directed graphs satisfying constraints of protection models in [10, 60]. In addition, we proposed the Degree Information Loss (DM) to minimize the information loss made by anonymizing these graphs.

Then, in Chapter 4, we addressed the problem of anonymizing KGs containing both attributes' values and relationships. We firstly analyzed attacking scenario that an adversary can use to re-identify users in anonymized KGs and proposed k-Attribute Degree (k-ad) to protect users in this scenario. To minimize the loss of users' attributes in KGs, we firstly presented the Attribute Information Loss Metric (AM). Since AM does not consider the truthfulness of attributes' values of users, we further presented the Attribute Truthfulness Information Loss Metric (ATM) to minimize the untruthfulness of these values. By combining AM and ATM with DM in Chapter 3, we can minimize the loss of both users' attributes and their relationships. Furthermore, we presented the Cluster-Based Knowledge Graph Anonymization Algorithm (CKGA) to anonymize KGs. CKGA extends CDGA

to anonymize not only users' relationships but also their attributes. Different from CDGA and other previous cluster-based anonymization algorithms [2, 6, 11, 42, 48], CKGA allows data providers to specify any clustering algorithm, that supports Euclidean distance, to anonymize KGs.

In Chapter 5, we further extended the work in Chapter 4 to allow data providers to publish newer versions of their anonymized KGs after inserting/updating/re-inserting/deleting nodes and edges in their original KG. We investigated the attacking scenario when an adversary exploits many versions of anonymized KGs to re-identify users. Then, we presented the k^w-Time-Varying Attribute Degree (k^w-tad) to ensure that users are not re-identified with a confidence higher than $\frac{1}{k}$ even if adversaries have access to w continuous anonymized KGs. We proposed the Cluster-Based Time-Varying Knowledge Graph Anonymization Algorithm (CTKGA) to generate anonymized KGs satisfying requirements of k^w-tad.

The conducted experiments on real-life datasets showed that our anonymized solutions for KGs are good enough to generate anonymized KGs. Furthermore, we compared our solutions with those in [10, 60] to show that ours can also generate high quality directed graphs.

6.2 Future work

We plan to extend the solutions presented in this thesis in many directions. In Section 6.2.1, we present the Personalized k-Attribute Degree, the personalized version of k-Attribute Degree. We plan to develop a personalized anonymization algorithm for the Personalized k-Attribute Degree and conduct experiments to evaluate the effectiveness of the developed algorithm. Then, we discuss other research directions of our work in Section 6.2.2.

6.2.1 Personalized Anonymization of Knowledge Graphs

In Chapter 4, we introduced k-Attribute Degree (k-ad) to protect users when the adversaries exploit both users' attributes and degrees of their relationships. k-ad prevents the adversaries from re-identifying any user in the anonymized KG with a confidence higher than $\frac{1}{k}$ when they know the attributes' values and out-/in-degree of all relationship types of the user. However, k-ad, as many other protection models (e.g., k-anonymity [49], the Paired k-degree [10], the K-in&out-Degree Anonymity [10]) is not able to protect different users with different protection levels [54].

Unfortunately, as showed in Chapter 2, the personalized anonymization solutions for directed graphs and KGs are still missing. In this chapter, we start to fill in this void by presenting the Personalized k-Attribute Degree (p-k-ad), an extension of k-ad, to allow users specifying their own values of k. The p-k-ad ensures that for every user u, attributes' values and out-/in-degrees of him/her are indistinguishable from those of other $k_u - 1$ users, where k_u is the specified k value of u.

Furthermore, we introduce the idea of how to generate anonymized KGs satisfying requirements of p-k-ad. To allow data providers to use their own clustering algorithm, we use the same approach of Chapter 4 to generate data points for users in the original KG

such that the Euclidean distance between two points is almost equal to the information loss of making attributes' values and out-/in-degree of these users identical. The information loss of users is measured by using the Attribute and Degree Information Loss (ADM) and the Attribute Truthfulness and Degree Information Loss (ATDM) described in Chapter 4. However, since the provided clustering algorithm can be not aware of the minimum number of users in the generated clusters, we need to modify the clusters. Such modifications must ensure that each user is in a cluster whose size is greater than or equal to his/her specified k value. Finally, we use the Knowledge Graph Generalization Algorithm (KGG) introduced in Chapter 4 to ensure that users in the same clusters having the same attributes' values and out-/in-degrees for all types of their relationships. The anonymization solution in this chapter is in progress. We are implementing the anonymization algorithm based on the above intuitive idea. Then, we will conduct experiments to evaluate the effectiveness of this anonymization solution.

In what follows, we will describe p-k-ad and the intuitive idea of our anonymization algorithm for p-k-ad.

Personalized Anonymization of Knowledge Graphs

Let G be a KG and $\overline{G}$ be its anonymized version. By using the extracted information, according to Definition 7, an adversary can use the attacking mechanism $\mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, u))$ to re-identify his/her victim u. The risk that u can be re-identified is $risk(\mathcal{T}_{\overline{G}}, \overline{G}, u) = \frac{1}{\sum_{v \in \overline{V}^U} \mathcal{T}_{\overline{G}}(\mathcal{BK}(u), I(\overline{G}, v))}$ (according to Definition 8).

In this section, we present the Personalized k-Attribute Degree (p-k-ad) to allow each user to specify their own value of k. More formally, p-k-ad is defined as follows:

Definition 23 (Personalized k-Attribute Degree)**.** *Let $\overline{G}(\overline{V}, \overline{E}, \overline{R})$ be an anonymized KG. $\overline{G}$ satisfies the Personalized k-Attribute Degree (p-k-ad) if and only if for every user in $\overline{V}^U$, there is a set $\mathcal{C}(\overline{G}, u)$ such that $\mathcal{C}(\overline{G}, u) = \{v \in \overline{V^U} | I(\overline{G}, u) = I(\overline{G}, v)\}$ and $|\mathcal{C}(\overline{G}, u)| \geq k_u$, where k_u is a positive integer number specified by u.*

If an anonymized KG $\overline{G}$ satisfies p-k-ad, data providers allows all users in $\overline{G}$ to control their risk of being re-identified by specifying their own values of k. The risk that any user u in $\overline{G}$ can be re-identified is at most $\frac{1}{k_u}$, where k_u is specified by u.

Overview of the Personalized Anonymization Approach

To generate anonymized KGs satisfying requirements of p-k-ad, we are working on a cluster-based anonymization algorithm, which will leverage on algorithms we have previously defined in Chapter 4. The algorithm will take as input a KG $G(V, E, R)$ and a set of positive numbers $\mathcal{K}$, where each positive number $k_u \in \mathcal{K}$ models the privacy protection level of a user $u \in V^U$. It then will generate the anonymized version of G, i.e., $\overline{G}$, according to three main steps:

1. **Users' points generation.** This step aims at generating a point $e_u \in \mathbb{R}^{d_2}$ for each user $u \in V^U$ such that the Euclidean distance between two points e_u, e_v is almost similar to the information loss of making attributes' values and out-/in-degrees of two users $u, v \in V^U$ identical. We use the same approach in Chapter 4 to generate these points.

2. **Clusters generation.** The goal of this step is to generate a set of clusters $C^{\overline{G}} = \{c \subseteq V^U || c| \geq \max_{u \in c} k_u\}$ such that the Euclidean distance between points of users in the same cluster are minimized.

3. **Knowledge graph generalization.** This step takes as input the set of clusters generated from the previous step and modifies G to generate $\overline{G}$ such that attributes' values and out-/in-degrees of users in the same cluster identical. At this purpose, we plan to use the Knowledge Graph Generalization Algorithm in Chapter 4 to perform this step.

6.2.2 Other Research Directions

We are going to extend our work to protect not only users' identities but also their sensitive values. This extension requires us to investigate new attacking methods, protection models, and anonymization algorithms for the scenarios presented in Chapters 4 and 5. Then, we plan to extend the techniques presented in Chapter 4 to support distributed scenarios where multiple parties collaborate to generate a single anonymized KG without disclosing users' sensitive information to other parties. Finally, we plan to study how to apply Differential Privacy [13] to anonymize KGs.

Bibliography

[1] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)*, 2005.

[2] Gagan Aggarwal, Rina Panigrahy, Tomás Feder, Dilys Thomas, Krishnaram Kenthapadi, Samir Khuller, and An Zhu. Achieving anonymity via clustering. *ACM Trans. Algorithms*, 2010.

[3] Adeel Anjum, Guillaume Raschia, Marc Gelgon, Abid Khan, Naveed Ahmad, Mansoor Ahmed, Sabah Suhail, M Masoom Alam, et al. τ-safety: A privacy model for sequential publication with arbitrary updates. *Computers & Security*, pages 20–39, 2017.

[4] R. J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *21st International Conference on Data Engineering (ICDE'05)*, pages 217–228, 2005.

[5] Kurt Bollacker, Colin Evans, et al. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. Association for Computing Machinery, 2008.

[6] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-anonymization using clustering techniques. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*, page 188–200. Springer-Verlag, 2007.

[7] Ji-Won Byun, Yonglak Sohn, Elisa Bertino, and Ninghui Li. Secure anonymization for incremental datasets. In *Workshop on secure data management*, pages 48–63. Springer, 2006.

[8] Ji Won Byun, Yonglak Sohn, Elisa Bertino, and Ninghui Li. Secure anonymization for incremental datasets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4165 LNCS, pages 48–63, 2006.

[9] Ricardo JGB Campello, Davoud Moulavi, et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, pages 1–51, 2015.

[10] Jordi Casas-Roma, Julián Salas, Fragkiskos D Malliaros, and Michalis Vazirgiannis. k-degree anonymity on directed networks. *Knowledge and Information Systems*, pages 1743–1768, 2019.

[11] Zhi-Guo Chen, Ho-Seok Kang, et al. An efficient privacy protection in mobility social network services with novel clustering-based anonymization. *EURASIP Journal on Wireless Communications and Networking*, page 275, 2016.

[12] 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*, pages 459–470, 2010.

[13] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming*, pages 1–12. Springer Berlin Heidelberg, 2006.

[14] A. Fahad, N. Alshatri, et al. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, pages 267–279, 2014.

[15] Benjamin CM Fung, Ke Wang, and Philip S Yu. Top-down specialization for information and privacy preservation. In *21st international conference on data engineering (ICDE'05)*, pages 205–216. IEEE, 2005.

[16] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, 2018.

[17] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, pages 78–94, 2018.

[18] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114, August 2008.

[19] Tiantian He and Keith CC Chan. Discovering fuzzy structural patterns for graph analytics. *IEEE Transactions on Fuzzy Systems*, 26(5):2785–2796, 2018.

[20] A. Hoang, M. Tran, A. Duong, and I. Echizen. An indexed bottom-up approach for publishing anonymized data. In *2012 Eighth International Conference on Computational Intelligence and Security*, pages 641–645, 2012.

[21] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. Cluster-based anonymization of directed graphs. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 91–100, 2019.

[22] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. Cluster-based anonymization of knowledge graphs. In *18th International Conference on Applied Cryptography and Network Security*, 2020.

[23] Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. Privacy-preserving sequentially publishing of knowledge graphs. In *37th IEEE International Conference on Data Engineering*, 2021.

[24] Xia Hu, Jiliang Tang, Yanchao Zhang, and Huan Liu. Social spammer detection in microblogging. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[25] Vijay S Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 279–288, 2002.

[26] S. Ji, P. Mittal, et al. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys Tutorials*, pages 1305–1326, 2017.

[27] Jia Jiao, Peng Liu, and Xianxian Li. A personalized privacy preserving method for publishing social network data. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8402 LNCS, pages 141–157. Springer Verlag, 2014.

[28] Vadisala Jyothi and V. Valli Kumari. Privacy preserving in dynamic social networks. In *Proceedings of the International Conference on Informatics and Analytics*, pages 1–8, 2016.

[29] C. Maria Keet. Closed world assumption. In *Encyclopedia of Systems Biology*, pages 415–415. Springer New York, 2013.

[30] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341. ACM, 2018.

[31] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 49–60, 2005.

[32] Michael Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*, pages 1–10. Springer, 2002.

[33] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.

[34] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 93–106, 2008.

[35] Xiangwen Liu, Qingqing Xie, and Liangmin Wang. Personalized extended (α, k)-anonymity model for privacy-preserving data publishing. *Concurrency Computation*, 29:1–18, 2017.

[36] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, 2006.

[37] Kamalkumar Macwan and Sankita Patel. Privacy preserving approach in dynamic social network data publishing. In *International Conference on Information Security Practice and Experience*, pages 381–398. Springer, 2019.

[38] Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 32–41, 2014.

[39] Nigel Medforth and Ke Wang. Privacy risk in graph stream publishing for social network data. In *IEEE 11th International Conference on Data Mining*, pages 437–446. IEEE, 2011.

[40] Prateek Mittal, Charalampos Papamanthou, and Dawn Xiaodong Song. Preserving link privacy in social network based systems. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[41] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[42] M Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. *Data & Knowledge Engineering*, pages 622–645, 2007.

[43] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 2017.

[44] Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 67–78, 2007.

[45] Jianwei Qian, Xiang-Yang Li, et al. Social network de-anonymization and privacy inference with knowledge graph model. *IEEE Transactions on Dependable and Secure Computing*, pages 679–692, 2017.

[46] Jianwei Qian, Xiang-Yang Li, et al. Social network de-anonymization: More adversarial knowledge, more users re-identified? *ACM Transactions on Internet Technology (TOIT)*, pages 1–22, 2019.

[47] Jianwei Qian, Xiang Yang Li, Chunhong Zhang, and Linlin Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *Proceedings - IEEE INFOCOM*, pages 1–9. IEEE, 2016.

[48] Klara Stokes. Graph k-anonymity through k-means and as modular decomposition. In *Nordic Conference on Secure IT Systems*, pages 263–278, 2013.

[49] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[50] Chih-Hua Tai, Peng-Jui Tseng, S Yu Philip, and Ming-Syan Chen. Identity protection in sequential releases of dynamic networks. *IEEE transactions on Knowledge and Data Engineering*, 26(3):635–651, 2013.

[51] Chenxu Wang, Zhiyuan Zhao, et al. Deepmatching: A structural seed identification framework for social network alignment. In *IEEE 38th International Conference on Distributed Computing Systems*, pages 600–610, 2018.

[52] Ke Wang, Philip S Yu, and Sourav Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 249–256. IEEE, 2004.

[53] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 754–759. Association for Computing Machinery, 2006.

[54] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240, 2006.

[55] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 689–700, 2007.

[56] Xiaojun Ye, Yawei Zhang, and Ming Liu. A personalized (α, k)-anonymity model. In *Proceedings - The 9th International Conference on Web-Age Information Management, WAIM 2008*, pages 341–348, 2008.

[57] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564, 2017.

[58] Mingxuan Yuan, Lei Chen, S Yu Philip, and Ting Yu. Protecting sensitive labels in social network data anonymization. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):633–647, 2011.

[59] Mingxuan Yuan, Lei Chen, and Philip S. Yu. Personalized privacy protection in social networks. *Proceedings of the VLDB Endowment*, 4(2):141–150, 2010.

[60] Xiaolin Zhang, Jiao Liu, Jian Li, and Lixin Liu. Large-scale dynamic social network directed graph k-in&out-degree anonymity algorithm for protecting community structure. *IEEE Access*, pages 108371–108383, 2019.

[61] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *2008 IEEE 24th International Conference on Data Engineering*, pages 506–515. IEEE, 2008.

[62] Fan Zhou, Lei Liu, et al. Deeplink: A deep learning approach for user identity linkage. In *IEEE Conference on Computer Communications*, pages 1313–1321, 2018.

[63] Hui Zhu, Hong-Bin Liang, Lian Zhao, Dai-Yuan Peng, and Ling Xiong. τ-safe (l, k)-diversity privacy model for sequential publication with high utility. *IEEE Access*, 7:687–701, 2018.

[64] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. pages 946–957. VLDB Endowment, 2009.

Appendices

Appendix A

Notations and abbreviations

A.1 Abbreviations

Abbreviation	Meaning
AM	The Attribute Information Loss Metric
ATM	The Attribute Truthfulness Information Metric
ADM	The Attribute And Degree Information Loss Metric
ATDM	The Attribute Truthfulness and Degree Information Loss Metric
CDGA	The Cluster-Based Directed Graph Anonymization Algorithm
CKGA	The Cluster-Based Knowledge Graph Anonymization Algorithm
CTKGA	The Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm
DM	The Out-and In-degree Information Loss Metric
k-ad	The k-Attribute Degree
k^w-tad	The k^w-Time-Varying Attribute Degree
KG	Knowledge Graph
OSN	Online Social Network
p-k-ad	The Personalized k-Attribute Degree

A.2 Notations

Symbol	Meaning
$\mathcal{G}$, $\overline{\mathcal{G}}$	the original and anonymized directed graph
G, $\overline{G}$	the original and anonymized KG
G_t, $\overline{G}_t$	the original and anonymized KG at time t
g_t^w, $\overline{g}_t^w$	the series of w original and anonymized KGs from time $t - w + 1$ to t
V, $\overline{V}$	the set of nodes in the original and anonymized KG
E, $\overline{E}$	the set of edges in the original and anonymized KG
R, $\overline{R}$	the set of relations in the original and anonymized KG
V^U, $\overline{V}^U$	the set of users in the original and anonymized KG
V^A, $\overline{V}^A$	the set of attributes' values in the original and anonymized KG
E^{UA}, $\overline{E}^{UA}$	the set of edges modelling attributes' values in the original and anonymized KG
E^{UU}, $\overline{E}^{UU}$	the set of edges modelling users' relationships in the original and anonymized KG
R^{UA}, $\overline{R}^{UA}$	the set of attributes in the original and anonymized KG
R^{UU}, $\overline{R}^{UU}$	the set of relationship types in the original and anonymized KG
V_t, $\overline{V}_t$	the set of nodes in G_t and $\overline{G}_t$
V_t^U, $\overline{V}_t^U$	the set of users in G_t and $\overline{G}_t$
E_t, $\overline{E}_t$	the set of edges in G_t and $\overline{G}_t$
R_t, $\overline{R}_t$	the set of relations in G_t and $\overline{G}_t$
$d_o(\mathcal{G}, u)$	out-degrees of user u in $\mathcal{G}$
$d_i(\mathcal{G}, u)$	in-degrees of user u in $\mathcal{G}$
$I_d(\mathcal{G}, u)$	out-/in-degrees of user u in $\mathcal{G}$
$d_o(G, r_u, u)$,$d_i(G, r_u, u)$	out- and in-degree of relationship type r_u of user u in G
$I_a(G, r_a, u)$	values of attribute r_a of user u in G
$I_a(G, u)$,$I_o(G, u)$,$I_i(G, u)$	attribute, out- and in-relationship of user u in G
$I(G, u)$	attributes' values and out-/in-degrees of user u in G
$\mathcal{I}(g_t^w, u)$	the series of attributes' values and out-/in-degrees in anonymized KGs of g_t^w
$\mathcal{BK}(u)$	the background knowledge that the adversary knows about user u
$\mathcal{BK}_t(u)$	the background knowledge about user u at time t
$\mathcal{BK}_t^w(u)$	the series of background knowledge about user u from time $t - w + 1$ to t
$\mathcal{U}(\overline{g}_t^w)$	the set of users appearing at least once in $\overline{g}_t^w$
$\mathcal{T}_{\overline{G}}$	the attacking mechanism to re-identify a user in a KG
$\mathcal{T}_{\overline{g}}$	the attacking mechanism to re-identify a user in a series of KGs

Appendix B

Datasets

In this thesis, we use 8 real-life datasets to perfom experiments: *Email-Eu-core* [57], *Google+* [19], *Freebase* [5], *Email-temp* [43], *Bitcoin Alpha* [30], *DBLP* [32], *Yago* [16], and *ICEWS* [16]. *Email-temp* [43] is an dynamic directed graph which contains emails sent between members of a research institution and their timestamps. Each member is a node and each edge models an email sent from a member to another. *Bitcoin Alpha* [30] contains data of users who trade using Bitcoin on Bitcoin Alpha[1]. Each node of this dataset represents a user. These users rate other members in a scale of -10 (total distrust) to +10 (total trust) and these ratings are modelled using edges between nodes. *DBLP* [32] illustrates citation networks of publications in DBLP. Each node is a publication and each edge describes a citation of a publication by another one. Similar to *Email-temp*, *Email-Eu-core* [57] also contains email data from a large European research institution. However, each user in this dataset belongs to a department and its edges do not consist of timestamps. *Google+* [19] contains attributes' values and a relationship type (i.e., $follow$) of Google+ users. Its nodes model either users or attributes' values and its edges represent attributes' values or the relationships of these users. *Freebase* [5] is a KG which contains attributes' values (e.g., *nationality*, *location*) and relationships (e.g., *spouse*, *parent*) of famous people (e.g., the film director Anthony Asquith). Each node models either a user or a attribute's value while each edge represents either the attribute's value of a user or a relationship between two users. *ICEWS* [16] is a dynamic KG consisting of political events with a specific timestamp. *Yago* [16] is a dynamic KG containing information and its timestamp derived from Wikipedia, WordNet, and other data sources.

Table. B.1 illustrates properties of these datasets. $|V^U|$ and $|V^A|$ denote the number of users and attributes' values of each dataset, respectively. The number of user-to-attribute and user-to-user relationship types are represented in columns $|R^{UA}|$ and $|R^{UU}$, respectively. $|E^{UA}|$ and $|E^{UU}|$ show the number of edges modelling user-to-attribute and user-to-user relationships, respectively. Since, in Chapter 5, we need to evaluate the effectiveness of our sequential anonymization solution for KGs on anonymized snapshots of dynamic KGs, we use two columns, i.e., $|T|$ and MCE, to show how these graphs change.

[1]http://www.btcalpha.com/

While $|T|$ demonstrates the number of timestamps these datasets have, MCE is the Mean Changed Edges of graph snapshots. MCE is measured as follows:

$$MCE(g_t^w) = \frac{1}{|T| \times |\mathcal{U}(g_t^w)|} \sum_{u \in \mathcal{U}(g_t^w)} \sum_{t \in T} n_i(G_t, u) + n_d(G_t, u)$$

where $n_i(G_t, u)$ and $n_d(G_t, u)$ are the number of inserted and deleted edges of a user u in G_t, respectively. If a user is deleted, all of his/her edges are calculated as deleted edges. Similarly, if he/she is inserted, his/her edges are considered as inserted edges. Therefore, MCE can measure how nodes and edges are inserted/deleted. The higher the values of MCE is, the more the user data changes.

Table B.1: Properties of datasets used for experiments.

Dataset	$\|\mathbf{V^U}\|$	$\|\mathbf{V^A}\|$	$\|\mathbf{R^{UA}}\|$	$\|\mathbf{R^{UU}}\|$	$\|\mathbf{E^{UA}}\|$	$\|\mathbf{E^{UU}}\|$	$\|\mathbf{T}\|$	**MCE**
Email-temp [43]	986	4	1	1	791,758	332,334	207,880	0.9
ICEWS [16]	2,617	1,870	187	144	19,814	12,807	365	0.2
Yago [16]	8,918	4,084	18	5	10,382,538	242,095	151	0.04
Email-Eu-core [57]	1,005	42	1	1	1,005	25,571	1	0
Freebase [5]	5,000	4,016	10	3	41,067	2,713	1	0
Google+ [19]	7,805	1,962	6	1	20,780	321,268	1	0
Bitcoin Alpha [30]	3,783	0	0	1	0	24,186	1	0
DBLP [32]	12,591	0	0	1	0	49,743	1	0

Three of the datasets illustrated in Table B.1, namely *Email-temp*, *ICEWS*, and *Yago*), have been used to conduct experiments which evaluate the effectiveness of our sequential anonymization for KGs (cfr. Chapter 5). These datasets are organized in snapshots with a different number of edges and nodes. Each snapshot is generated by adding and removing nodes/edges of its previous snapshot. However, some snapshots of these datasets have very few edges and it is impractical that a data provider will publish a new version of their dataset which contains very few edges. Therefore, we assume that the data provider will publish the new version of their dataset every time the number of edges their dataset has reached a specific threshold and the threshold will be the same for generating all of the snapshots. Then, we generate 50 snapshots for each dataset by merging continuous raw snapshots such that the standard deviation of the number of edges of the generated snapshots is minimized. So, all snapshots have a similar number of edges.

Since KGs can also illustrate directed graphs, we use three directed graphs (i.e., *Email-temp*, *Bitcoin Alpha*, and *DBLP*) to evaluate the effectiveness of our anonymization solutions (Chapters 3, 4, and 5) comparing to those for directed graphs in [10, 60]. *Email-Eu-core*, *Google+*, and *Freebase* have been used to evaluate our anonymization solutions for KGs (Chapters 4 and 5). Finally, we used *Email-temp*, *ICEWS*, and *Yago* to perform experiments on our sequential anonymization solution (Chapter 5).

Appendix C

Publications

1. Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Clusters-Based Anonymization of Directed Graphs". Proceedings of the IEEE International Conference on Collaboration and Internet Computing, 2019.

   Abstract: Social network providers anonymize graphs storing users' relationships to protect users from being re-identified. Despite the fact that most of the relationships are directed (e.g., follows), few work (e.g., the Paired k-degree [10] and K-In&Out-Degree Anonymity [60]) have been designed to work with directed graphs. In this paper, we show that given a graph, DGA [10] and DSNDG-KIODA [60] are not always able to generate its anonymized version. We overcome this limitation by presenting the Cluster-based Directed Graph Anonymization Algorithm (CDGA) and prove that, by choosing the appropriate parameters, CDGA can generate an anonymized graph satisfying both the Paired k-degree [10] and K-In&Out-Degree Anonymity [60]. Also, we present the Out- and In-Degree Information Loss Metric to minimize the number of changes made to anonymize the graph. We conduct extensive experiments on three real-life datasets to evaluate the effectiveness of CDGA and compare the quality of the graphs anonymized by CDGA, DGA, and DSNDG-KIODA. The experimental results show that we can generate anonymized graphs, by modifying less than 0.007% of edges in the original graph.

2. Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Clusters-Based Anonymization of Knowledge Graphs". Proceedings of the International Conference on Applied Cryptography and Network Security, 2020.

   Abstract: While knowledge graphs (KGs) are getting popular as they can formalize many types of users' data in social networks, sharing these data may reveal users' identities. Although many protection models have been presented to protect users in anonymized data, they are unsuitable to protect the users in KGs. To cope with this problem, we propose k-Attribute Degree (k-ad), a model to protect users' identities in anonymized KGs. We further present information loss metrics tailored to KGs and a cluster-based anonymization algorithm to generate anonymized KGs satisfying k-ad.

Finally, we conduct experiments on five real-life datasets to evaluate our algorithm and compare it with previous work.

3. Anh-Tu Hoang, Barbara Carminati, and Elena Ferrari. "Privacy-Preserving Sequentially Publishing of Knowledge Graphs". Proceedings of the IEEE International Conference on Data Engineering, accepted, 2021.

   Abstract: Knowledge graphs (KGs) are widely shared because they can model both users' attributes as well as their relationships. Unfortunately, adversaries can re-identify their victims in these KGs by using a rich background knowledge about not only the victims' attributes but also their relationships. A preliminary work to deal with this issue has been proposed in [22] which anonymizes both user attributes and relationships, but this is not enough. Indeed, adversaries can still re-identify target users if data providers publish new versions of their anonymized KGs. We remedy this problem by presenting the k^w-Time-Varying Attribute Degree (k^w-tad) principle that prevents adversaries from re-identifying any user appearing in w continuous anonymized KGs with a confidence higher than $\frac{1}{k}$. Moreover, we introduce the Cluster-based Time-Varying Knowledge Graph Anonymization Algorithm to generate anonymized KGs satisfying k^w-tad. Finally, we report the results of the experiments we have done on six real-life datasets to evaluate our algorithm and compare it with previous work.