Automatic Learning of Anonymization for Graphs and Dynamic Graphs

Maria Coralia Laura Maag

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et électronique (Paris)

Présentée par

Maria Coralia Laura NECULA MAAG

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Apprentissage automatique de fonctions d'anonymisation pour les graphes et les graphes dynamiques

soutenue le 8 avril 2015

devant le jury composé de :

M. Patrick GALLINARI, Professeur, Université Pierre et Marie Curie	Directeur
M. Ludovic DENOYER, Professeur, Université Pierre et Marie Curie	Co-Directeur
M. Fabrice ROSSI, Professeur, Université Panthéon-Sorbonne	Rapporteur
M. Benjamin NGUYEN, Professeur, INSA Val de Loire	Rapporteur
M. Bernd AMANN, Professeur, Université Pierre et Marie Curie	Examinateur
Mme. Maryline LAURENT, Professeur, Télécom SudParis	Examinatrice
M. Philippe JACQUET, Directeur de recherche, Alcatel-Lucent Bell Labs	Examinateur
M. Hakim HACID, Professeur associé, Zayed University, Émirats Arabes Unis	Examinateur

DOCTORAL THESIS
UNIVERSITY PIERRE AND MARIE CURIE

Speciality

Computer Science

École doctorale Informatique, Télécommunications et électronique (Paris)

Presented by

Maria Coralia Laura NECULA MAAG

For the degree of

DOCTOR of the UNIVERSITY PIERRE AND MARIE CURIE

Thesis subject :

Automatic Learning of Anonymization Functions for Graphs and Dynamic Graphs

defended the 8 of April 2015

Jury :

M. Patrick Gallinari, Professor, University Pierre and Marie Curie	Director
M. Ludovic Denoyer, Professor, University Pierre and Marie Curie	Co-director
M. Fabrice Rossi, Professor, University Panthéon-Sorbonne	Referee
M. Benjamin Nguyen, Professor, INSA Val de Loire	Referee
M. Bernd Amann, Professor, University Pierre and Marie Curie	Examiner
Mrs. Maryline Laurent, Professor, Télécom SudParis	Examiner
M. Philippe Jacquet, Research Director, Alcatel-Lucent Bell Labs	Examiner
M. Hakim Hacid, Ass. Professor, Zayed University, United Arab Emirates	Examiner

Abstract

Data privacy is a major problem that has to be considered before releasing datasets to the public or even to a partner company that would compute statistics or make a deep analysis of these data. Privacy is insured by performing data anonymization as required by legislation. In this context, many different anonymization techniques have been proposed in the literature. These techniques are difficult to use in a general context where attacks can be of different types, and where measures are not known to the anonymizer. Generic methods able to adapt to different situations become desirable. We are addressing the problem of privacy related to graph data which needs, for different reasons, to be publicly made available. This corresponds to the anonymized graph data publishing problem. We are placing from the perspective of an anonymizer not having access to the methods used to analyze the data. A generic methodology is proposed based on machine learning to obtain directly an anonymization function from a set of training data so as to optimize a tradeoff between privacy risk and utility loss. The method thus allows one to get a good anonymization procedure for any kind of attacks, and any characteristic in a given set. The methodology is instantiated for simple graphs and complex timestamped graphs. A tool has been developed implementing the method and has been experimented with success on real anonymized datasets coming from Twitter, Enron or Amazon. Results are compared with baseline and it is showed that the proposed method is generic and can automatically adapt itself to different anonymization contexts.

Remerciements

Je tiens tout d'abord à remercier mes directeurs de thèse, Prof. Patrick Gallinari et Prof. Ludovic Denoyer, pour leur disponibilité et leur soutien qui ont été essentiels à l'accomplissement de ce travail.

Je tiens également à remercier Philippe Jacquet pour son soutien et ses conseils qui m'ont beaucoup guidée dans mes travaux de recherche.

Je remercie ceux qui ont été à l'initiative de cette thèse de doctorat, notamment Hakim Hacid, Johann Daigremont et Bruno Aidan, ainsi que Xavier Andrieu pour son aide et ses conseils tout au long de ces années de thèse.

Je remercie l'ensemble de mes collègues présents ou passés au Bell Labs. Je remercie plus particulièrement Alonso Silva Allende pour avoir accepté le travail de relecture des premières versions de mon manuscrit ainsi que Gerard Burnside et Marc-Olivier Buob pour leurs conseils sur la présentation de ce travail.

Je remercie ma hierarchie présente ou passée au sein des Bell Labs, et plus particulièrement Messieurs Jean-Luc Beylat et Chris White pour m'avoir fourni le cadre propice à la realisation de ce projet.

Je remercie les Professeurs Benjamin Nguyen et Fabrice Rossi d'avoir accepté la charge d'être rapporteurs de ma thèse. Je remercie également les Professeurs Maryline Laurent, Bernd Amann, Hakim Hacid et Philippe Jacquet qui m'ont fait l'honneur d'avoir accepté de faire partie de mon jury de thèse ainsi que Laurent Viennot et Bernd Amann pour avoir accepté de faire partie de mon jury de mi-parcours.

Un grand merci à tous ceux qui ont assisté à ma soutenance et qui parfois sont venus de loin: j'ai été très touchée par votre présence.

Je tiens à exprimer ma profonde gratitude à ma famille, mon mari Stéphane, mes enfants Lucas et Antoine ainsi qu'à ma mère, Laurentia Veronica. Sans leur support cette thèse aurait été difficilement réalisable.

Contents

List of Figures

List of Tables

A mon grand-père, Ion Degeratu

Chapter 1

Introduction

1.1 General Context

Privacy is a major problem that has to be considered when releasing data. Datasets need to be released for different purposes. In the case of outsourcing, companies owning data are using third parties to develop algorithms which need to be tested on samples of real data. Datasets analysis is very important also for marketing applications. The tremendous need of open data available for research communities is another reason to release those datasets. Today it is difficult and in most of the cases it is impossible for a data owner to release complex datasets in a completely safe environment without any risk of individual's re-identification.

Protection of data is mandatory before any release as required by the legislation (e.g., Directive 95/46/EC of the European Parliament [October 1995]), but also to build confidence between data owner and the persons directly concerned by data disclosure. Among the common approaches to preserve data's privacy, **anonymization** is a real trend which opens several interesting and important challenges. **Data anonymization** is the process transforming the original dataset into a dataset not allowing individual's re-identification or other sensitive data re-identification.

Since the arrival of Web 2.0, the user has become the important actor on the Internet. No longer a passive consumer of published information, the Web user is nowadays the central focus through his relationships and interactions at the forefront. New technologies give to each user the possibility to produce and publish his own data. The amount of data available on the Internet increases therefore drastically. Data available for analysis becomes even more vulnerable than before. In parallel, tools analyzing data extracted from the Internet become more and more powerful. With the arrival of the multitude of social data sources and analysis techniques, the risk for the user to be subject of an

infringement to his private data has been considerably multiplied. Many anonymization processes are not effective when external information (as for example previous knowledge of the adversary or information published on social networks) is combined with the anonymized information.

Examples of **data attacks** in the past (Netflix attack or well known AOL data disclosure detailed below) have revealed the necessity to use more sophisticated techniques than simple identifiers anonymization.

In 2006 AOL shared a search record of around half million users. Journalists from New York Times (in Barbaro and Zeller [2008]) revealed the identity of the user 4,417,749 as being a certain Mrs. Thelma Arnold. The link between searches performed by the user 4,417,749 and external information concerning Mrs. Arnold had been obtained by data analysis. As a consequence, the CTO of the company and the researchers responsible for sharing data were all fired.

Netflix case is another example of privacy infringement because of insufficient data anonymization before release. The Netflix prize data is a dataset made available in the context of the Netflix prize competition. The purpose of this competition was to choose the best collaborative filtering algorithm to predict user ratings for films based on previous ratings. The training dataset provided by Netflix contained 100,480,507 ratings of 480,189 users on 17,770 movies. Netflix anonymized data before release to protect user's privacy by replacing all personal information of the users as well as their identifiers and by adding noise into data (by deleting ratings, inserting alternative ratings and dates or modifying rating dates). Based on the strong correlation between a user's profile in Internet Movie Database (IMDb) and the Netflix released dataset, Narayanan and Shmatikov [2006] propose a de-anonymization method capable of making the correlation between IMDb and the released anonymized dataset. In 2009 four Netflix users filed a lawsuit against Netflix, alleging that Netflix had violated U.S. fair trade laws and the Video Privacy Protection Act concerning wrongful disclosure of video tape rental or sale records (United States Congress [1988]) by releasing the datasets. In 2010 Netflix announced that the competition will not be pursued as a consequence of the lawsuit and of the Federal Trade Commission privacy concerns. The result of these privacy infringements was the serious limitation of the open access to data for research or any other purposes.

The actor willing to access private information is generally called an "adversary". Potential adversaries come from different categories as described in Narayanan and Shmatikov [2009]:

- Government-level agencies with the purpose of global surveillance (which are among the strongest adversaries)

- Commercial enterprises having as goal abusive marketing are interested in this kind of information
- The de-anonymized information could be helpful for phishing attacks (to perform those attacks with more personalized and better targeted information) and spamming
- De-anonymization performed on specific individuals by investigators, neighbors or colleagues is another category of attack

In the Netflix or AOL cases, if the released data were anonymized with stronger algorithms in order to take into account external information access as well as a large panel of possible attacks, these privacy breaches would not have been possible. The need for data access and the need for privacy resulted in an increasing importance of data anonymization process. The added value of data anonymization is that apart from tackling the sensitive issue of data privacy, it will facilitate information sharing between involved actors. By inserting perturbations in data, data owners are able to hide sensitive information that would hinder their competitive advantage if revealed to business opponents. Also, from the perspective of end users, which in their majority are reluctant in sharing personal information, the fact that data will be guaranteed as being anonymized will give them confidence in sharing their data.

1.2 Graph Data Anonymization Issue

Anonymization techniques have been proposed for different kind of data representations as described in Zhou et al. [2008] and Wu et al. [2010]. Techniques to anonymize data for which each record represents a separate entity (as in a relational database) have been developed and are currently used (e.g., Fung et al. [2010]). However, datasets have moved from traditional models to more complex ones, like for example graph or hypergraph structures.

For example, relational and communication data issued from social networks or telecommunication networks can be represented in complex structures. Communication interactions for example, can be represented as a graph with multiple oriented and timestamp labeled links. Anonymization techniques for tabular data cannot be successfully applied to these complex models (Aggarwal et al. [2011]).

When data can be represented in the form of a graph, even more information is available than in tabular data to identify an entity. The more the structure which models data is complex, the easier will be to identify entities inside it. As a consequence, methods to anonymize data in an efficient way will be more complex than for tabular data.

The work of Liu and Terzi [2008] describes the difficulty to anonymize graphs compared to anonymizing tabular data. Any adversary could use the topology structure of the graph for de-anonymization. Anonymization strategies will then modify the structure by adding remaining edges to the initial graph. This is called perturbation of the graphs structure. In Aggarwal et al. [2011] it is shown that for large sparse graphs it is difficult to preserve utility after perturbation as it exists in this kind of graphs re-identification signatures that are very robust to perturbations. To perturb this kind of graphs, the level of added noise necessary is very high and implies an important loss of the associated utility.
Depending on the anonymization problem we need to solve, specific evaluation metrics will be defined. However, developing metrics for quantifying loss of information for a specific task is a research work on its own. This work is generally complex as the utilities to be measured on a structure issued from communication data are diversified.

We address in this thesis the problem of **privacy related to graph data** which needs, for different reasons, to be **made publicly available**. This is called the **anonymized graph data publishing** problem. The reasons for data publication could be related to:

- the need to involve a third party in the analysis of data. Most of the time data owners do not have the time or the competences to perform these analyses. Therefore, an additional actor appears in the person of the data analyzer.
- the exposure of data to third party applications in order to provide new services based on this data (e.g., crowdsourcing applications, advertisers),
- the need for data publicly available for research reasons or technical challenges (e.g., medical research),
- testing algorithms developed by an external partner of the data owner, in a real environment.

We consider here the perspective of an anonymizer **not having access to the methods used to analyze** data. We suppose that the anonymizer is a trusted party of the anonymization process. Some approaches from the state of the art consider also the data publisher as an untrusted environment (Allard et al. [2011]). We are not tackling this case and we consider that the untrusted environment starts once data is published.

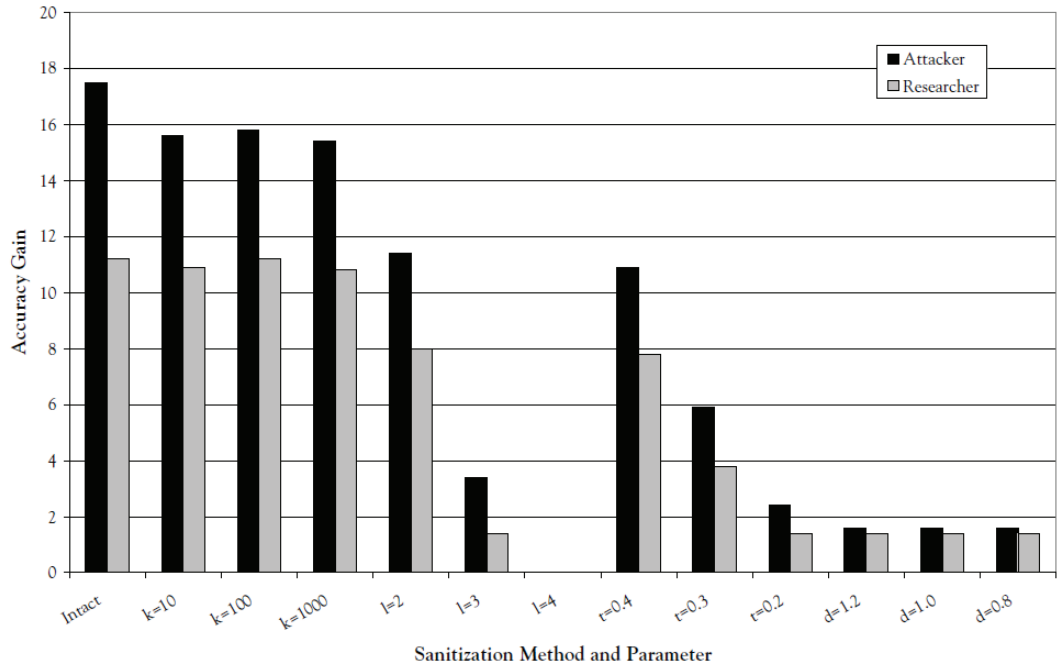


Figure 1.1: Effects on privacy and utility of anonymization with different parameters: black bars represent the privacy of the data smaller bars signifying more privacy; gray bars represent the utility of the data.

1.3 Contributions

Ohm [2010] describes several examples of datasets having been anonymized which have been easily de-anonymized and re-identified by scientists. Usually, only the knowledge on data anonymized without any external enrichment, is not sufficient to perform a de-anonymization. However, external data is easily available and in some cases makes the anonymization useless, as described before. A naive anonymization (e.g., replacing identifiers by a code) is insuring a very low level of anonymity. To protect people privacy, other techniques for anonymization are needed. Modifying original data could be necessary in most of the cases. The problem is that data modified **looses a part of its utility**.

The anonymization techniques have as main goal to find a **compromise between privacy and utility** of data. Figure 1.1 from Ohm [2010] is showing the relation between privacy protection and utility. Black bars represent the privacy, smaller bars meaning more privacy. Grey bars represent the utility of data.

By varying different anonymization parameters, privacy is protected with different degrees. However, the loss of utility is almost proportional to the increase of privacy protection. In this graphic, the best privacy protection and utility preservation would be represented by a short black bar (privacy protected) next to a long gray bar (utility

preserved).
Most of existing anonymization techniques are based on certain aspects of privacy preserving related to a certain utility to preserve. There is no general method for anonymization which could be applied to any privacy and utility aspects of data. Recent techniques based on differential privacy are promising but their privacy guarantees are not universal and attacks are still possible. Methods currently used to anonymize complex graph datasets for a third party release are usually using naive anonymization combined with noise addition or data modification related to a certain type of attack. Techniques providing differential privacy based guarantees are currently employed and they result most of the times into good guarantees for privacy. However, the main goal of the anonymization is to find a compromise between privacy and utility of the data. Usually, this balance is set manually. Differential privacy proposes a condition on the data delivery mechanism insuring good privacy guarantees. We propose a learning methodology for finding a good anonymization function in a given context. The two approaches are then complementary.
The main challenge of my thesis is to propose and validate an **automatic technique** able to learn the **best adapted anonymization function** for graph data publication **without direct access to the utility** to be preserved on data and to the methods used by the analyzer of data, by taking into account a **large and flexible panel of possible de-anonymization risks**.

My thesis **main contributions** are:

- Provide a state of the art of the existing methods in the domain of graph data anonymization.
- Show the vulnerability of graphs in particular when data can be decomposed in a multitude of subgraphs (e.g, in the case of timestamped graphs) and propose a system taking into account this vulnerability when performing data anonymization.
- Propose a generic method based on machine learning techniques able to learn efficient anonymization functions for different contexts. This context has two main components: (i) the data analyzer, (ii) a panel of risks. Our goal is to learn anonymization strategies able to adapt themselves to different situations. In our formulation, both the data analyzer and the risks will be represented as blackboxes.
- Instantiate the proposed methodology and the blackboxes for, in a first step, simple graphs and in a second step complex timestamped graphs issued from call logs.
- Propose and implement new methods for timestamped graphs de-anonymization mainly in order to instantiate the analyzer blackbox.

- Experiment the instantiation of the methodology on real datasets of simple graphs and call logs graphs with success. The results have been compared with methods issued from the state of the art.

1.4 Detailed Content

A part of my work (**Chapter 2**) has been dedicated to the analysis and comprehension of existing publications in the data anonymization field, and particularly in graph data anonymization. The first part of this chapter deals with the privacy question regarding graph data and potential attacks on this type of data. In the second part of the chapter, the data anonymization issue is described as well as the consequence of anonymization on the utility of data. Several techniques for specific utility loss evaluation existing in the literature are also described.
Research works dealing with the anonymization issue for particular representations of data like hypergraphs or temporal graphs are identified. A section is dedicated to the differential privacy concept and to its possible applications to data release problem. A separate section deals with machine learning techniques applied to anonymization. In many works, machine learning is used for de-anonymization purposes. A case of data anonymization using machine learning is described. The tradeoff between privacy and utility of data has been explored by a certain number of papers in the literature.

Chapter 3 is illustrating the vulnerability of graph data when data can be partitioned into subgraphs. Methods for anonymization are usually applied on the entire dataset and are robust for attacks made against the entire graph structure. We show that local attacks can still be performed in the released dataset by partitioning the anonymized dataset into subgraphs. A system is then proposed based on a data decomposer module performed on data before anonymization. The system described in this chapter has been filed as a patent (Hacid and Maag [2014]).

A second contribution of my thesis described in **Chapter 4** consists in proposing a new generic methodology to learn, for a given family of data, a given set of privacy attacks and data analysis process, what are the best parameters for the anonymization function in order to preserve the balance between utility loss and privacy risk. Existing algorithms or anonymization tools (described for example in Vinogradov and Pastsyak [2012]) are dependent of the settings of a certain number of parameters. The choice of those parameters is an essential step for the anonymization process and is important when dealing with complex data and with multiple privacy risks or utilities related to

the anonymized data. The problem of finding the best adapted parameters in a given context reveals to be a complex issue. We use two optimization methods to approximate the searched parameters: estimation of distribution algorithms and genetic algorithms. The proposed methodology has been published in Maag et al. [2014].

Chapter 5 is instantiating the proposed methodology for simple graphs. Panels of possible privacy risks are described corresponding to the instantiation of the privacy risks blackbox. The analysis blackbox is instantiated by using utility loss evaluation methods to simple graphs. Baseline methods used for comparison are also described. The results of the implemented instantiation for the methodology are evaluated on real datasets issued from Twitter and Amazon. The method adapts well to a new context (i.e. to changes performed in the blackboxes) and outperforms baseline methods. Results from this chapter have been presented and published in an international conference (Maag et al. [2014]).

The contribution described in **Chapter 6** deals with finding anonymization parameters for complex structures anonymization (such as graphs resulting from communication logs with a temporal component). Most of the work in anonymization dealing with relational data, tackled simple graphs or bipartite graphs, and the relations are in most cases single and not oriented. The instantiation of our parameter learning methodology on such structures has lead to the proposal and implementation of new techniques for privacy risks detection and utilities measurements.The obtained results are compared with baseline methods results. The experimentation is performed on datasets issued from Twitter and Enron corpus. The method behaves well and results are improved when diversifying the number and the type of learned parameters used for the anonymization function. This work is currently submitted for publication.

Chapter 7 contains the conclusion and the perspectives of my work. The results improve with the number of parameters used for learning for simple graphs as well as for complex graphs. A direction to continue this work would be to explore the behavior of the system when learning with diverse parameters and eventually time related parameters. Another interesting direction consists in studying how to offer guarantees based on differential privacy when using anonymization functions learned with the proposed methodology.

Chapter 2

State of the Art

Chapter 2 contains the state of the art regarding existing publications in data anonymization field. A first part deals with privacy from the perspective of graph data anonymization and the associated potential attacks. In the following part, data anonymization techniques are described as well as their consequences on the utility of data. Techniques usually employed to evaluate utility loss of data are also evocated. Anonymization approaches when dealing with particular representations of data like hypergraphs or temporal graphs are described. Last part of the chapter is dedicated to the differential privacy concept and to its possible applications to anonymized data release problem. Machine learning techniques applied to anonymization are evocated from the perspective of data de-anonymization, data anonymization and tradeoff exploration between privacy and utility.

Contents

2.1 Introduction

Privacy protection is an important research subject as it represents a critical challenge. A huge amount of research works have been treating this complex issue. We aim to address here the privacy protection related to data release when data is the result of people's interactions. Data containing private information about individuals can be retrieved and stored, following each country, according to particular legal conditions. In most of the cases, data release can be performed only if there are guarantees that no private information about individuals can be extracted. This is done through a process called "data anonymization" or "data sanitization". Mainly, there are two models of data sanitization as described in Dwork et al. [2006]: **interactive models** and **non-interactive models**.

- In the **interactive** version, a mechanism is provided such as the user can send queries on data and gets answers or noisy answers.
- For the **non-interactive** functioning, data owner publishes an anonymized version of data collected. The techniques applied on the dataset before publication are called "anonymization" techniques or "de-identification" techniques.

Different anonymization techniques already exist for tabular data and are applied with success on datasets where tuples are independent one of each other. The problem we are tackling here is the **data anonymization for datasets issued from interactions between individuals, before data publishing, without a detailed knowledge of the analysis to be performed on data**. This type of data can be represented as a complex structure (e.g., a multiple oriented timestamped graph). We are dealing with the case of the **non-interactive data sanitization for complex graph data**.

This chapter is organized as following: the first section discusses privacy protection for graph data and existing de-anonymization techniques. Section 2.3 identifies the existing anonymization techniques for graphs. Section 2.4 describes techniques used to evaluate utility loss when anonymizing graph data. Section 2.6 is dedicated to the differential privacy issue and its applications to data release problem. Last section provides description of machine learning techniques used in the context of data anonymization.

2.2 Privacy Protection for Graph Data

2.2.1 What to protect?

The first question we have to answer to is what part of data we need to protect in this type of complex structure. This section aims to make a synthesis of the answers to the question "What to protect?" when anonymizing complex data structures issued from individual's interactions.
Liu and Terzi [2008] classify data disclosure (i.e. data which should not be re-identified in the anonymized version of the dataset) to be avoided in graphs of a given network in three categories. However, other categories appear in the bibliography; we have to consider also that one type of disclosure can lead to another type of disclosure as described in Zheleva [2011]. We list hereafter different types of disclosure, this list being non-exhaustive:

- **Identity disclosure**: disclosure of the identity of a node (e.g., user present in a certain disease network). The identity disclosure occurs when the adversary is able to map a real life identity on a node in the anonymized data.
- **Link disclosure**: sensitive relationships between two individuals (e.g., a financial transaction having occurred between two nodes).
- **Content disclosure**: data associated with the node or a link is disclosed (e.g., political opinion of a node).
- **Affiliation link disclosure**: this kind of disclosure described in Zheleva [2011] is determining whether a person belongs to a certain group or not.

Others classifications have been made for example in Zhou et al. [2008]. In a social network represented as a graph, diverse information revealed can be considered as being an attempt to the user privacy as described in Zhou et al. [2008]:

- **Vertex existence:** an individual appearing or not in a certain network (e.g., network of millionaires, of a disease infection etc.).
  An infection network is a valuable resource to be analyzed in public health research. However, by making public this resource, the identity of a person present in this network could be revealed and the privacy of this person could be affected.

- **Vertex properties (Hay et al. [2007], Hay et al. [2008], Liu and Terzi [2008], Ying and Wu [2008]):** a related example is the degree of a node in a financial support network.
  In Hay et al. [2007] the structural re-identification of a node is studied. Once a node is identified in an anonymized graph (by using the degree of that node), the information is combined with the attributes of that node.

- **Sensitive vertex (Campan and Truta [2009], Zhou and Pei [2011], Zheleva and Getoor [2007]) or edge (Campan and Truta [2009], Ying and Wu [2008]) labels:** the disease label in a disease infection network or the labels of a user in a social network.
  The labels of a certain vertex in a graph can be identified if the vertex is uniquely identified in the graph. Even more, sensitive labels corresponding to a certain vertex can be revealed if the vertex is identified as part of a group having in common these sensitive labels.

- **Link relationship (Cormode et al. [2008]), Edge existence (Zhang and Zhang [2009]):** the link in a finance transaction network signifies that a transaction occurred between the two nodes which could be sensitive information.
  This same link in a bipartite graph representing the authors and the papers corresponding to a conference means that a given author has written a given paper; when the graph represents data coming from a pharmacy, the adversary could identify which user bought which medicine. In Cormode et al. [2008] the main goal is to protect this link by grouping the nodes and the entities in a bipartite graph. Zhang and Zhang [2009] proposes to swap and delete edges based on degree information to prevent re-identification.

- **Link weight (Liu et al. [2008]):** a social network weighted with the communication frequency between two individuals or a financial transactions network with the amounts of each transaction.
  This second example corresponds to information a company should not disclose. If another company is able to have this information, then this company could make an offer just below the price obtained from the financial transaction network. The paper Liu et al. [2008] is tackling the anonymization of weighted networks trying

to keep unchanged main properties of that network (as e.g., the shortest path) in order to preserve utility of the data.

- **Graph metrics**: metrics as betweenness, closeness, path length, centrality or reachability may be considered private data in certain cases.

Defining what to protect before any tentative of data anonymization is a very important step because as mentioned before, there is no anonymization technique capable of protecting all data components while preserving data's utility. New techniques based on differential privacy (described in 2.6) try to guarantee general data protection.

2.2.2 De-anonymization Techniques

The only access to the anonymized data, even if this data would be anonymized with naive techniques, is usually not enough for being able to re-identify nodes, edges or content. Additional knowledge is needed for that purpose. With the arrival of user created content and the arrival of social networks, this additional knowledge is available everywhere on the web. This knowledge could also be personal knowledge of the malicious user (not coming from the web).
We are going to call the person(s) aiming to de-anonymizing data the "Adversary". This person could perform the de-anonymization attack with the explicit objective of accessing the private data. However, this person could stumble upon the anonymized data unintentionally and his previous knowledge could make this data to be de-anonymized, once again in an unintentional manner.
According to the manner the adversary is de-anonymizing data, the de-anonymization techniques can be classified in: **active attacks** and **passive attacks**.

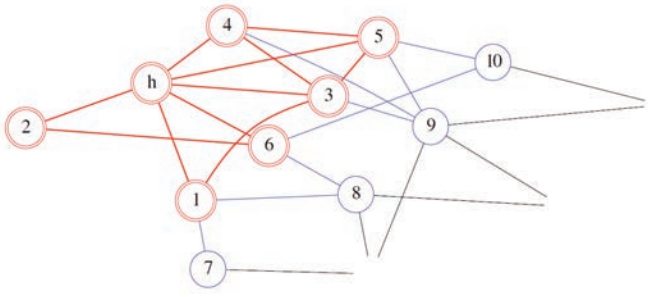


Figure 2.1: Fingerprint planted in the initial graph: each node represents a user. Fingerprint obtained by the "seed" subgraph formed by vertices h, 1, 2, 3, 4, 5, 6 can be uniquely re-identified using the degree-sequence of h.

- Peng et al. [2012] describes the case of an **active attack**; the adversary plants a "seed" into the target social network before its release. The "seed" is the equivalent of a specially designed subgraph with a particular fingerprint as illustrated in Figure 2.1. Peng et al. [2012] is tackling the special case of undirected graphs where the relation between users is mutual. The acquisition of the background knowledge to be combined with the anonymized data is performed by the adversary by creating or stealing a few accounts and connecting them with a few others accounts. The paper considers that the adversary hasn't complete control over the connections which is more realistic than the considerations made in previous works. The purpose of the adversary is to localize the seed after the anonymized graph being released, then to identify the seed's neighbors. The case of active attacks is related to malicious users having the explicit objective to de-anonymize data. The difficulty to perform an active attack comes from the fact that in most social networks, to create a new link the acceptation of the link from both of the terminating nodes is necessary. However, in communication networks or in Twitter like networks (based on a follower-followed structure) there is no need for mutual acceptation to create an embedded graph structure. In Narayanan and Shmatikov [2009] is described the fact that active attacks are easy to detect as real users never link back to sibyl nodes (new identities created for the attack). The paper concludes that a large-scale active attack is unlikely to be feasible.

- In the case of **passive attacks**, the adversary might not have the explicit objective of de-anonymizing the given data and no new link is created especially for the attack. However, some existing structures in social networks or communication graphs present typical patterns and can be sometimes unique so that the structure could be easily re-identified in the released anonymized network.

De-anonymization attacks on graph data can be grouped according to the exploited weak point in **four main families**: structural attacks, rich graphs attacks, dynamic data based attacks and learning on missing information attacks. Each type of attack is detailed further.

2.2.2.1 Structural Attacks

Structural attacks are based on the principle that if a certain query based on structural information of a graph over a dataset has a limited number of answers, this may lead to target re-identification and to privacy infringement.
Let $\mathcal{G}$ be a graph and $\mathcal{G}'$ be the anonymized version of the given graph.

Structural attack definition (Zou et al. [2009]):
If a query Q over $\mathcal{G}$ has a limited number of match vertices in $\mathcal{G}$, then target might be uniquely identified. If Q is based on structural information about the target in $\mathcal{G}$, this is called a ***structural attack****.*

An example of structural attack is the "friendship attack" described in Tai et al. [2011c]. This kind of attack is using the vertex degrees pair of two individuals and their friendship relation and is illustrated in Figure 2.2. Information that Bob and Carl are friends as well as their total number of friends can lead to their re-identification in the anonymized dataset. The paper introduces the notion of *k2-Degree Anonymity*, notion defined as

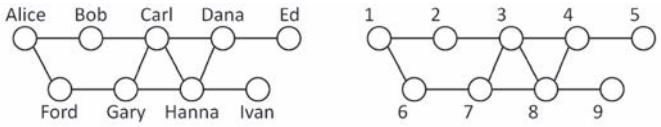


Figure 2.2: Friendship attack: Bob and Carl can be uniquely identified by their vertex degree pair (2,4).

following:
A graph $\mathcal{G}$ is ***k2-Degree Anonymous*** *if for every vertex with an incident edge of degree pair $(d1, d2)$, there exist at least $k-1$ other vertices such that each of the $k-1$ vertices also has an incident edge of the same degree pair.*
The main idea is to not allow a degree pair structure to be distinguished among $k-1$ other structures.

Structural attacks have been classified into four main types in Zou et al. [2009]:

- **Degree Attack**
  For example an adversary who knows that Bob has four neighbors. If in the anonymized graph there is only one vertex having four neighbors, then the node Bob is de-anonymized.

- **Sub-graph attack**
  This kind of attack consists in a sub-graph which can be uniquely identified in the global graph. The sub-graph to be identified is called the query graph.

- **1-Neighbor-Graph Attack**
  It can be considered as a special case of sub-graph attack. The adversary in this case knows the 1-neighborhood of the vertex and the connections in this neighborhood. If this structure can be uniquely identified in the global graph, then the node can also be identified.

- **Hub Fingerprint Attack**
  This kind of attack considers that the adversary has identified some hub in the anonymized network. Then, if the adversary knows how the vertex to be de-anonymized situates relatively with the identified hub, the vertex may then be identified.

According to the properties the adversary knowledge is related to, the survey Zhou et al. [2008] classifies in a different manner the possible attacks on a graph:

- **Identifying attributes of vertices**
  Relies on a set of attributes identifying in a unique manner an individual. In Campan and Truta [2009] not only structural information of data represented as a graph is taken into account. They assume a much richer data model and pay special attention to the node attribute data. The attributes related to a node are then classified in identifiers (e.g., name, id), quasi-identifiers (e.g., zip code) and confidential or sensitive attributes (e.g., diagnosis or income).

- **Vertex degree**
  The vertices degrees are information easy to collect by adversaries (Hay et al. [2007], Liu and Terzi [2008], Ying and Wu [2008], Tai et al. [2011a]). In Hay et al. [2007] the example is given for a user Bob for which the adversary knows that he has at least three neighbors. This restrains to only a small number the collection of nodes representing the user Bob.

- **Link relationship (Campan and Truta [2009], Cormode et al. [2008], Zheleva and Getoor [2007])**
  Channels people use to communicate with each other can be sensitive information (phone, email, IM). In Cormode et al. [2008] the example is given for private data related to associations between entities. As an example, for a particular pharmacy, the list of its clients is not sensitive information. For this same pharmacy, the list of the medicines sold out is not sensitive information. However, the association between these two non sensitive information is private information at it may indicate a health issue for the clients of that pharmacy.

- **Neighborhoods (Hay et al. [2007], Ying and Wu [2008], Hay et al. [2008], Zhou and Pei [2011], Zhou and Pei [2008])**
  A neighborhood based attack can have as target for example a victim having four good friends who know each other. In Zhou and Pei [2011] the knowledge of the adversary is in the 1-neighborhood of the victim. The adversary knows the relations existent between the neighbors of the victim and based on this knowledge in some cases it is possible to re-identify in a unique manner the given vertex.

- **Embedded subgraphs (Backstrom et al. [2007])**
  The attack can be performed by an adversary embedding a specific subgraph into a social network before the network being released. This is assimilated to an active attack. The idea in Backstrom et al. [2007] is that the adversary first chooses an arbitrary set of users to be attacked. Then he creates a small number of accounts with edges to the targeted users. He also creates a specific pattern of links between the new created accounts (that could be easily recognized in a global graph). Once the anonymized data is released, the adversary can easily identify the particular pattern embedded in the released graph.

- **Graph metrics (Hay et al. [2007], Ying and Wu [2008])**
  Graph metrics (betweenness, closeness, centrality, path length, reachability, etc) can also be used to breach the privacy. In Hay et al. [2008] an example is given on how to use a hub for de-anonymization purposes. A hub is a node in the network with high degree and high betweenness centrality (which is the proportion of the shortest paths in the network that include the node). A hub is a challenge for anonymization for being easy to distinguish in a given network.

2.2.2.2 Attacks on Rich Graphs

The possible re-identification methods in a graph are as diversified as the complexity of the representation of a graph. According to the complexity of the dataset, data to be anonymized can be represented as a simple graph, as an oriented graph, as a graph with multiple links, as an hypergraph etc. Recent papers bring examples of how re-identification can be made by using minimal knowledge on the victims especially when the anonymized graph structure is rich.
Li and Shen [2011a] introduce an attack based on the property of a hyperedge rank. The anonymization method related to this attack tackles the hyperedge rank anonymization problem.
In Li and Shen [2010] the weights present in a graph are used for identity disclosure (the weights for each adjacent edge to a vertex as well as the sum of those weights). A weighted graph is introducing much more information than a simple graph making the anonymization even more difficult than in a simple graph. Li and Shen [2010] propose a solution for the weight anonymization problem.

2.2.2.3 Dynamic Data Based Attacks

In Bhagat et al. [2010] and Tai et al. [2011b] the special case of dynamic data is evocated. Indeed, when publishing several times an instance at a time t_1 and t_2 of a network, the

adversary holding certain knowledge on the evolution of the network between the two releases is capable to re-identify nodes in the anonymized graph data.
As an example described in Tai et al. [2011b], Figure 2.3 illustrates two sequential releases of a network of patients in a certain hospital. By having the knowledge that user Bob has gone to the hospital for the first time in the period between the two releases, the adversary is able to re-identify Bob in the anonymized data.

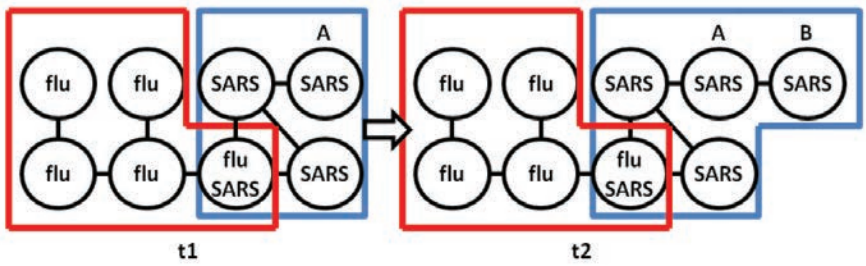


Figure 2.3: Sequential releases of a dynamic social network: Bob can be re-identified between t_1 and t_2 as being the only person having visited the hospital for the first time.

Bhagat et al. [2010] is illustrating the need to use a link prediction algorithm in order to take into account the future evolution of the released network when anonymizing it.

2.2.2.4 Learning Missing Information in Partially Anonymized Graph

This type of attack addresses released networks containing only partial information. Indeed, in most of the social networks, some profiles are public and some others profiles are private. The social network could be then released only by hiding those private profiles. The research study Zheleva and Getoor [2009] describes a method to infer those sensitive attributes and to disclose hidden group memberships in a partially anonymized network. This disclosure is mainly performed by using learning methods and classification models in order to determine the missing labels in the partially anonymized network.

2.3 Anonymization Techniques for Graphs

Three main types of anonymization techniques on graphs can be distinguished:

1. Structural methods based on k-anonymity concept,
2. Randomization techniques,
3. Generalization techniques.

We detail hereafter each one of the family methods.

2.3.1 Structural Modification Based on k-Anonymity Concept

The first family is related to methods performing structural modifications on the graph in order to obtain the k-anonymity property (see definition below). This family is mainly related to the structural attacks that can be performed on a graph and it contains methods providing the anonymized graph with k-anonymity concerning a panel of properties (e.g., vertices degrees, neighborhoods). Several approaches to modify a graph for k-anonymization have been described in the literature.

k-anonymity definition

A certain structure is k-anonymous with respect to a structure query if there exist at least $k-1$ other structures that match the given structure query.

Problem definition for k-anonymity (Liu and Terzi [2008])

Input: a simple graph $\mathcal{G}(\mathcal{V},\mathcal{C})$ to be anonymized. $\mathcal{V}$ is the set of vertices in $\mathcal{G}$ and $\mathcal{C}$ is the set of edges in $\mathcal{G}$.
Use a set of graph-modification operations (e.g., adding or deleting edges) on $\mathcal{G}$ to construct the anonymized graph $\mathcal{G}'$.
Output: Anonymized graph $\mathcal{G}'(\mathcal{V}',\mathcal{C}')$, with $\mathcal{V}'$ the set of vertices in $\mathcal{G}'$ and $\mathcal{C}'$ the set of edges in $\mathcal{G}'$ so that:

- $\mathcal{G}'(\mathcal{V}',\mathcal{C}')$ is k-anonymous (in Liu and Terzi [2008] according to the node degree),
- $\mathcal{G}'$ is structurally similar to $\mathcal{G}$.

The cost of anonymization is then defined according to the difference in the similarity between $\mathcal{G}$ and $\mathcal{G}'$. The similarity between $\mathcal{G}$ and $\mathcal{G}'$ is considered from a structural perspective; the cost (which is the similarity delta between $\mathcal{G}$ and $\mathcal{G}'$) is proportional with the modifications performed on the graph $\mathcal{G}$ to obtain the graph $\mathcal{G}'$.
In Liu and Terzi [2008] for instance, $\mathcal{G}$ is modified by adding new edges to obtain $\mathcal{G}'$ and the paper focuses on identity disclosure. The problem definition of the graph anonymization is stated and a two steps solution is proposed. In a first step a new degree sequence for the graph to be anonymized is constructed such that this sequence is k-anonymous. In the second step given the new degree sequence, a graph is constructed (if possible) maximizing the structural similarity with the initial graph. The cost is in this case defined as being proportional with the number of edges added to the original graph $\mathcal{G}$. The problem to be solved is how to minimize this cost.
The k-anonymization problem has always a solution (the worst solution being transforming the graph into a complete graph with all the nodes having the same degree).

The problem related to the worst solution is that the cost is high and the utility is lost. Hay et al. [2007] consider the anonymity of a social network by taking into account the structural similarity between nodes from the corresponding social graph. In Hay et al. [2007] the queries are related to the degrees of the nodes and its neighbors.
However a graph being k-anonymous according to a structure query does not necessarily protect the privacy of the given nodes. The k-anonymization is usually defined according to a given structure query. If the structure query changes, even if the graph is k-anonymized for example according to its nodes degrees, this anonymization will not be robust according to a neighborhood query structure attack.
Narayanan and Shmatikov [2009] argues on the fact that the auxiliary information the adversary has is not restricted to the neighborhood of a single node and this information is a global one which extends the types of possible queries structures.

Additionally, k-anonymity does not protect the nodes from the identification of certain attributes. The notion of **l-diversity** has been introduced related to graphs.

L-diversity definition (Zheleva [2011]):
A set of records in an equivalence class C is l-diverse if it contains at least l "well-represented" values for each sensitive attribute. A set of nodes V satisfies l-diversity if every equivalence class $C' \subseteq V$ is l-diverse.

Even if a given node cannot be uniquely identified, if the k similar nodes have sensitive attributes in common, then their privacy will not be protected.

Structural modification based on k-anonymity concept can be classified in two main categories: **optimization based methods** and **greedy modification approaches**, methods detailed further.

Optimization Graph Construction Methods

Optimization methods are based on the principle of finding the best solution to the given anonymization problem. An example of optimization method is given in Liu and Terzi [2008] in which the k-degree anonymization problem is tackled. In this paper the notion of k-degree anonymity is defined as mentioned earlier. The solution proposed has two steps:

- In the first step of the solution, a degree sequence of $\mathcal{G}$ is formed in a descending order. The optimization part consists in finding the optimal degree sequence

constructed from the original degree sequence and complying with the k-degree anonymity condition by performing a minimum number of edge addition operations.

- The second step of the solution constructs a graph with the optimal degree sequence resulting from step 1.

Greedy Graph Modification Approaches

Greedy graph modification approaches have as main principle the modification of local structures with the goal of obtaining results close to the optimal one.
In Zhou and Pei [2008] a greedy algorithm has been used to anonymize a social network to protect it against neighborhood attacks. There is no addition of fake vertices as this operation is considered as a change in the global structure of the graph. The algorithm has two main steps:

- First, all the neighborhoods in the social graph are extracted.
- In the second step of the algorithm, the neighborhoods of the vertices extracted are organized into groups of similar neighborhoods. They are then modified in order to respond to the k-anonymity requirement for neighborhoods.

As the distribution of degrees in a graph is following a power law, the most vulnerable nodes are the ones with high degrees as they neighborhoods are complex. Therefore the algorithm starts the graph modification with the high degree vertices.

In Zou et al. [2009] the main idea is to protect privacy by creating k homomorphic (thus indistinguishable) components. The algorithm K-Match (KM) has several steps

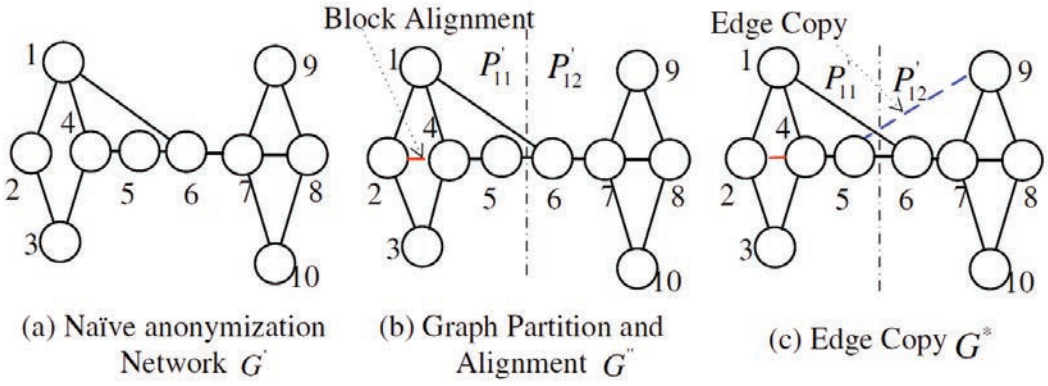


Figure 2.4: Graph partition and edge copy: edge between nodes 2 and 4 and crossing edge between 5 and 9 are added in the anonymized graph.

(illustrated in Figure 2.4):

1. The input of the algorithm is the original graph $\mathcal{G}$ and the parameter k.
2. The original graph is first transformed into $\mathcal{G}'$, a naive anonymized graph by replacing real identifiers with numbers.
3. $\mathcal{G}'$ is partitioned into blocks which are clustered into groups of at least k blocks. This graph partitioning part is very important for reducing the cost of the anonymization operation. Finding the optimal graph partitioning is an NP-complete problem. A greedy algorithm (Kuramochi and Karypis [2005]) is used to partition the graph $\mathcal{G}'$ based on frequent sub-graph identification.
4. Graph alignment is performed on all the blocks of each group
5. For all crossing edges between blocks, edge-copy is performed (Figure 2.4). New edges are added in order to obtain synmetric blocks. Original graph $\mathcal{G}$ is a subgraph of anonymized graph $\mathcal{G}'$.

2.3.2 Randomization Techniques

The second family of anonymization techniques for graphs is based on randomization techniques. The original graph is modified randomly by adding noise either by adding, deleting, switching edges or vertices and their attributes. Several methods have been described in Zhou et al. [2008]:

Randomized edge construction methods

Methods consisting in performing m edge deletions followed by n edge insertions (Hay et al. [2008]) based on a randomized edge construction method.

Randomized spectrum preserving methods

The spectrum of a graph is equivalent to the set of eigenvalues of the adjacency matrix of the graph; the spectrum is considered to have close relation with graph characteristics. The approach in Ying and Wu [2008] is similar to Hay et al. [2008] as the method consists in adding or deleting edges. The additions/deletions are performed on a random base but the effects on the graph spectrum are taken into account when choosing which edges to add or remove.

Randomized weights perturbation methods

This problem is tackled in Liu et al. [2008] and data utilities considered are the lengths of shortest paths between vertices and the shortest path between vertices in a selected subset. Two methods are proposed:

- Use of a Gaussian randomization multiplication (with a noise matrix of mean 0 and a given standard deviation) to perturb the weights of the edges.
- Use of a greedy perturbation algorithm with the goal of keeping the shortest path after perturbation as the original shortest path before the perturbation.

Note that differential privacy (described in 2.6) is considered in many works as a special case of randomization techniques. Most of the anonymization techniques respecting the differential privacy guarantees are based on adding random noise (e.g., adding Laplacian noise to the dataset).

2.3.3 Generalization Techniques

These anonymization techniques are based on the idea of clustering vertices and edges into groups and then form a super-vertex. The inconvenient of the clustering based methods is that the graph may be shrunk after anonymization and local structures will be difficult to analyze.
There are four main classes of clustering-based approaches.

Vertex clustering methods

Vertex clustering methods consist in delivering an anonymized graph which is a generalized graph of the original one, with a supernode instead of an original group of nodes. In Hay et al. [2008] the nodes of the graph are partitioned into disjoint sets. These nodes are considered as supernodes since they are nodes of a generalized graph. The partitioning of nodes is performed such that the resulting generalized graph maximizes utility and preserves privacy.

Edge clustering methods

Edge clustering methods consist in delivering a representation of the original graph wherein the relational information exist between clusters of vertices. This method consists in leaving the set of edges intact. The edges will only exist between the clusters of vertices. Figure 2.5 illustrates this approach and is described in Zheleva and Getoor [2007]:

Figure 2.5: Edge clustering methods: relational information only between clusters of vertices.

Vertex and Edge Clustering Methods

Vertex and edge clustering methods consist in partitioning original graph into clusters then combining nodes into a generalized node and edges between clusters into a single edge. In Campan and Truta [2009] data of the graph to be anonymized is clustered. For each cluster, the corresponding subgraph is extracted and the nodes of the subgraph are collapsed into a single node. The information about the number of nodes in the cluster is attached to this generalized node as well as the number of edges in the original cluster. Then the inter-cluster edges will be collapsed into a single edge and the structural information released will limit to the total number of edges collapsed into a single edge between the two clusters.

2.4 Utility Loss Evaluation

The new graph obtained after anonymization is compared to the original graph in terms of utility loss. The utility is a notion very difficult to define and quantify regarding graphs representing a given network. The utility depends directly of the use to be made with the anonymized data. Compared to tabular data, when anonymizing graphs it is very difficult to quantify the loss of utility of data to be protected. A change in a local area of a graph is modifying properties and behavior for the entire graph.
In Wu et al. [2010] a classification is described of the utility loss evaluations when modifying graph data. The article is identifying three main types of utility to be evaluated when modifying data in a graph structure: graph topological properties, graph spectral properties and network queries aggregation.

2.4.1 Graph Topological Properties

Considering the general graph properties, some papers compare the new graph according to the properties of this resulting graph compared to the original graph. For example, important applications of social network data are issued from the analysis of the graph properties (Zhou et al. [2008]). Properties compared may be as listed above:

- **Degree distribution**
  In Tai et al. [2011c] the degree distribution is preserved thanks to the first step of the algorithm where the vertices with similar degrees are clustered.

- **Degrees centralities**
  After anonymization usually the degrees centralities are decreased (Liu and Terzi [2008], Tai et al. [2011c]) as in an anonymized graph strong leaders and influential vertices may be de-anonymized.

- **Clustering coefficients**
  The clustering coefficient is decreased in Tai et al. [2011c] as the algorithm may connect distant vertices and disconnect vertices in the same clique.

- **Average path length**
  The average path length in Tai et al. [2011c] is increased as some vertices in different cliques are connected and some vertices in the same cliques are disconnected.

- **Number of edge changes from $\mathcal{G}$ to $\mathcal{G}'$**
  In many algorithms, the additions and deletions of edges are almost equal and they increase linearly to the value of k for the k-anonymity based algorithms.

Song et al. [2011] contains a description of the utility metrics used in graphs. The utilities described reflect the properties preservation for the social network associated with the graph and we are listing them hereafter:

- **Diameter**
  The longest path of the graph.

- **Radius**
  The minimum graph eccentricity (the maximum distance between a vertex and any other vertex in the graph) of any vertex in the graph.

- **Density**
  The ratio of the number of edges to the number of possible edges in a graph.

- **Degree centrality**
  It is related to a vertex v and it represents the number of vertices adjacent to v in the graph.

- **Closeness centrality**
  It is related to a certain vertex v and it represents the inverse of the average distance of the vertex to all the other vertices in the graph. It measures how close a vertex is to all other vertices. The closeness centrality measure is proportional

with the importance of a node in a graph; the closer a vertex is to other vertices, the faster the information can be exchanged. Therefore this measure is usually used in the context of information diffusion or influence calculation level in social networks.

- **Betweenness centrality**
  This measure is related to a given vertex v and it represents the ratio of the number of shortest paths in the graph that pass by v to the total number of shortest paths in the graph. This type of measure is also used for influence calculation or information diffusion.

- **Eigenvector centrality**
  This measure indicates each vertex importance according to its connections to important vertices, similar to the concept used in Google's Pagerank algorithm.

- **Global clustering coefficient and local clustering coefficient**
  The global clustering is defined for the whole graph and depends on the number of triangles and connected triples in the graph.
  The local clustering coefficient is a measure associated to each vertex in the graph. It is proportional to the number of connected vertices in its neighborhood compared to all the possible edges of the neighborhood.

- **Mean geodesic distance**
  It represents the average shortest path length of a graph.

- **Algebraic connectivity**
  It is the second smallest eigenvalue of the Laplacian matrix of the graph and it suggests how well connected a graph is. If the graph is not connected, the value of this measure is 0.

- **Earth movers's distance**
  It is used to measure the distance between two distributions and it represents the amount of work necessary to transform one distribution into another. It can be used to compare degree distributions of graphs for example.

According to the measures described above, the anonymization cost needs to be quantified. For example, in a k-anonymization algorithm, this cost can be quantified as being proportional to the number of edges deleted and inserted in the anonymized graph.
According to the utility needed to be preserved, these measures and the associated cost are usually defined before anonymization. In Li and Shen [2011b] the utility needed is the structure of the communities in the hypergraph. The anonymization cost related to

a hyperedge is defined as the difference between the ranks after and before anonymization.
Another example of cost function is the combination between the number of edges added/deleted, the number of vertices linked with an external neighbor and the proportion of labels generalized in the graph.

2.4.2 Graph Spectral Properties

Another method to quantify the loss in the utility of the data consists in the evaluation of the changes in the spectrum of the graph. In Ying and Wu [2008] it is shown the close link between the spectrum of a graph and the graph characteristics.
The spectrum of a graph is defined as being the eigenvalues of the adjacency matrix or of other related matrix (as the Laplacien). Ying and Wu [2008] is describing an anonymization method based on add, delete or switch operations on the edges of the graph, operations guided by the spectrum of the modified graph. The idea is to add, delete or switch edges by modifying as little as possible the associated spectrum of the graph.
In Ying and Wu [2009] the main idea is to re-generate a new synthetic graph matching a certain list of properties associated to the real social network. Among those properties are those correlated with the spectrum of the graph.

2.4.3 Network Queries Aggregation

In the case of the network queries aggregation, the measure of the utility loss is proportional to the delta between the answer to a set of queries on data of the graph $\mathcal{G}$ and the graph $\mathcal{G}'$.
In Cormode et al. [2008] the evaluation method consists in listing a set of types of queries (SQL like queries) and compare the answers before and after anonymization on a bipartite graph. The queries are grouped in 3 categories:

1. Graph structure only queries (if the anonymized graph is isomorphic with the original graph, the answer to this type of queries will be perfectly accurate).
2. Attribute predicate on one side only consisting in computing the aggregate of nodes satisfying certain properties.
3. Attributes predicate on both sides consisting in computing the aggregate of nodes on both sides and their relations satisfying required properties.

In Zhou and Pei [2008] the utility is evaluated according to the answer to aggregate queries. An example of aggregate queries is the computing for a label l_1 of its nearest vertex of label l_2. The utility loss is calculated by randomly picking 10 labels pairs and then calculating the average of the error rate when computing the shortest path between the pairs of labels in the original and anonymized graph.

2.5 Complex Graphs Anonymization

2.5.1 Hypergraphs

A hypergraph is a generalization of a graph in which an edge can connect any number of vertices. Due to the power of hypergraphs to represent multi-relations among objects, they are more and more used in publishing data. However, the privacy of data is not well preserved as a particular structure in a hypergraph can be easily re-identified. Anonymizing data by simply removing identifiers is not sufficient for a graph and is even less sufficient for a hypergraph. In a hypergraph, as an edge can connect to more than one node, the neighborhood of a given vertex has a higher probability to be unique than in a simple graph.
In a graph, an edge has always the degree equal to two and this degree is not powerful in a de-anonymization task. In a hypergraph representing a social network, the degrees of the hyperedges are elements allowing an adversary to re-identify a certain vertex in the anonymized hypergraph.

For example, in Li and Shen [2011b] a social network is modeled by using a hypergraph and the groups a user is part of are the hyperedges of the hypergraph. The main privacy attack considered in this paper is the identity disclosure. A potential attack on the anonymized data in a hypergraph is based as mentioned above on the edge rank property. This paper shows that in a hypergraph representing social network from Facebook where each member is a vertex and edges are represented by interest groups, naive data anonymization is not enough to preserve the privacy of a given vertex. The rank sequence of a vertex is defined as being the set of ranks of the incident edges of that vertex. Based on the rank sequence, an attack is described by identifying a unique rank sequence for a given vertex which leads to the identity disclosure for that given node. Based on this, Li and Shen [2011b] describes a rank-based hypergraph anonymization (RHA) in two steps:

1. a rank anonymization (RA) which consists in modifying the rank set to insure that each element cannot be distinguished

2. a hypergraph construction (HC) consisting in reconstructing a hypergraph with the same vertex set and the perturbed rank set

The main goal of the operation is to minimize the information loss between the original hypergraph and the new anonymized hypergraph and to find a compromise between utility and efficiency regarding privacy preservation. The RA algorithm described has as main goal the community preservation when modifying the hypergraph for anonymization.

2.5.2 Temporal Graphs

Another representation of datasets as complex graphs is related to temporal graphs. Datasets issued from communication logs or relational data is often labeled with temporal information. This information can then be exploited for de-anonymization purposes by a potential adversary.
Most of the prior works on relational data anonymization propose static privacy approaches capable of protecting data released at a certain time t. The work described in Tai et al. [2011b] is an approach for protecting data in a dynamic context when several releases are performed during a certain time. However, to the best of our knowledge there is no anonymization model taking into account the analysis in time of graphs with multiple oriented timestamped edges.
Sherkat et al. [2013] is providing a study concerning the timestamped event sequence anonymization based on time and event generalization. The paper considers two types of attacks: sequence identification and event prediction. For the first type of attack, k-anonymity based privacy evaluation is used. The idea is not to allow the association of any sequence in the original database of timestamped event sequences with less than k sequences in the published dataset.
Sharad and Danezis [2013] is tackling the de-anonymization from the perspective of call detail records publishing. It is showed that call detail records in the case of the dataset D4D (Data for Development, Orange) released by Orange (French telecom operator) can still be de-anonymized by using the 1-hop neighborhood of the published nodes.

2.6 Differential Privacy for Graph Data Anonymization

Differential privacy is a quite recent concept aiming at providing certain guarantees for the privacy of released data. It has been used in a first time for interactive data delivery. Recently, approaches to provide differential privacy for non-interactive data release have been described in the literature.

2.6.1 Principle

Differential privacy is not a condition on the dataset but is a condition on the data release mechanism. Differential privacy related anonymization techniques could be classified within the randomization techniques, as the mechanism generates noisy views of the original dataset.

Differential privacy applied on tabular data is resumed in Dwork et al. [2006] as following:

For a database access protocol (corresponding to the delivery mechanism), an adversary $\mathcal{A}$ and a particular database x, the random variable $\mathcal{T}_{\mathcal{A}}$ is denoted the transcript, corresponding to the output of the delivery mechanism.

A mechanism is ϵ-indistinguishable if for all pairs $x, x' \in D^n$ which differ in only one entry, for all adversaries $\mathcal{A}$, and for all transcripts t:

$$\left| ln \left(\frac{Pr[T_A(x) = t]}{Pr[T_A(x') = t]} \right) \right| \leq \epsilon \tag{2.1}$$

Therefore, when ϵ is small, the definition is equivalent to the requirement for all the transcripts t being such as:

$$\frac{Pr[T_A(x) = t]}{Pr[T_A(x') = t]} \in 1 + \epsilon \tag{2.2}$$

As mentioned in Leoni [2012], the ϵ-differential privacy for a mechanism is such as the same mechanism executed on two databases differing in one row will probably output the same result.

To limit the possible linkability attacks, the system has to keep track of the queries sent to the system and a certain "privacy" credit amount is accorded to the user. One of the main concerns regarding differential privacy, pointed out by the opinion G29 [April 2014], is the fact that it is mandatory to consider the combination between all possible queries, an error being to consider each query independently.

This kind of condition works well on tabular data for interactive database interrogation. When tuples of the database are independent, the differential privacy is a strong condition and allows hiding participation of any tuple in the database.

The paper Kifer and Machanavajjhala [2011] shows that privacy guarantees of differential privacy degrade when applied to correlated data. Kifer and Machanavajjhala [2011] is arguing that it is not possible to guarantee the privacy without a certain number of assumptions on the data delivery mechanism. The paper is pointing out privacy leaks when, for example, previous knowledge like deterministic statistics are known on the database. This kind of statistics could add correlations into data. Adversary knowledge should be taken into account even in the context of differential privacy.

This same paper points out that for social networks where correlations exist between

nodes the database excluding a given node has important implications on neighbor nodes. In the case of social networks, the ϵ-differential privacy would need an extremely small ϵ corresponding to a very large amount of noise added to any query. This wouldn't provide any information about the social network and the utility of data would be completely lost. In this paper it is also shown that by using only a naive ϵ-differential privacy on the queries, an attacker could infer the participation of a certain user to the social network. There are some attempts to provide differential privacy guarantees to graph data. Task and Clifton [2012] is one of the first approaches to use differential privacy in a graph context, i.e. in a context where different entries are interacting among each other. In this paper, the node-privacy is satisfied if the differential privacy is satisfied for all pairs of neighbor graphs. Neighbor graphs are defined as graphs differing according to one vertex and the edges adjacent to this vertex. This induces important restrictions on the queries the system is able to compute on the graph. The paper approach is related to an interactive differential privacy mechanism (the dataset is not publicly released in this case). The problem we tackle deals with data publishing. Some attempts where made to apply the principle of differential privacy to data release, attemps described further.

2.6.2 Differential Privacy for Data Release

In O'Hara [2011] it is claimed that differential privacy is limited to the interactive functioning. In this report on privacy and transparency for the UK government, the differential privacy approach is seen as a mechanism able to set a tolerable level of privacy for interactive mechanisms. Both, the privacy and transparency report for the UK government (O'Hara [2011]) and the Opinion 05/2014 on Anonymization Techniques (G29 [April 2014]) are describing and evaluating differential privacy only for the interactive mechanism.

However, recent work encourages researchers to pursue in the direction of using differential privacy for non-interactive settings.

Differential privacy for data publishing consists either in releasing a new database composed of synthetic individuals or in releasing a perturbed version of the original dataset. As mentioned above, the condition on the differential privacy says that the query output obtained from a database D should be slightly different from the query output produced when querying a neighbor database D'. For the non-interactive model existing research considers a unique query on the original graph performed only once and using the entire privacy budget (Sala et al. [2011]).

In Leoni [2012] the mechanism on which differential privacy is applied is of type $M : \mathcal{D} \to \mathcal{R}$, the differential privacy condition applying for example on a count of individuals filling a certain condition. When considering a mechanism of type $M : \mathcal{D} \to \mathcal{D}$, privacy

is more difficult to preserve.

In Sala et al. [2011] a new graph $\mathcal{G}'$ is generated from the original dataset $\mathcal{G}$ according to a desired level of ϵ-differential privacy. Mir et al. [2013] consider anonymization for call detail records, but the calls are considered as row entries, no interaction information is kept in the dataset. The algorithm described in this paper, uses distributions (like commute distance, calls per day) to produce synthetic CDRs for a number of synthetic users and length of simulated time.

Wang et al. [2013] revisit the definition of the differential privacy, definition which in classical approaches treats the dataset as a collection of rows, each row corresponding to an individual record. The definition revisited for graphs aims to ensure that including or excluding links between individuals makes no statistical difference on the result. In this context, the neighboring graphs are two graphs differing according to one edge. In this case, differential privacy ensures that the probability to have a particular output is almost the same whether having or not a particular edge in the original graph.

In Ahmed et al. [2013] an anonymization method for graph data publishing is proposed, approach satisfying the differential privacy guarantees.

In most of the existing research papers the impact of differential privacy anonymization on the utility of data is proved through experimentation.

Differential privacy, just as k-anonymity or l-diversity has its strong parts but also has identified weaknesses. In Cormode [2010] it is shown that even under differential privacy, disclosure can take place. In the described attack, instead of learning information at an individual level, properties of the population are learned. These properties allow predicting private information about an individual.

Even if differential privacy is a promising direction for anonymization, privacy is not completely guaranteed against all types of powerful attacks.

In the literature, differential privacy and k-anonymity based techniques have most of the time been viewed as completely different approaches to ensure data's privacy. Several studies (Li et al. [2011], Comas [2013], Soria-Comas et al. [2014]) bring a link between k-anonymity and differential privacy.

The idea in Li et al. [2011] is to bring differential privacy's strong guarantees into the practical k-anonymization methods. Main criticisms brought to the differential privacy are the important amount of noise to add into the dataset in order to achieve differential privacy guarantees and the fact that the utility is guaranteed only for a restricted type of queries. In Soria-Comas et al. [2014] it is shown that the amount of noise necessary to bring the differential privacy guarantees to the dataset can be reduced by performing the method on a k-anonymous version of the dataset.

2.7 Machine Learning in Data Anonymization Process

2.7.1 Machine Learning used for Data De-Anonymization

Machine learning techniques have been used in the past mainly for de-anonymization purposes. Learning missing information in partially anonymized graphs has been described by Zheleva and Getoor [2009]. This data disclosure is performed by using learning methods and classification models in order to determine the missing labels in partially anonymized graph.
In the work of Sharad and Danezis [2014], machine learning (decision forest) is used to evaluate anonymization techniques. The anonymization technique is modeled as a blackbox and the de-anonymization process is associated to a learning problem. The paper proposes to replace manual de-anonymization techniques by a generic learning algorithm. The learning algorithm is provided with examples of nodes being normally unlinkable. The conclusion of the paper is that good de-anonymization attacks can be produced even by using an automated algorithm based on machine learning.

2.7.2 Machine Learning used for Data Anonymization

Machine learning techniques have been used in the past also for data anonymization. In Szarvas et al. [2007] machine learning techniques are used in the context of medical records anonymization. The main idea is to use machine-learning named entity recognition on semi-structured documents in order to identify personal health information to be anonymized.

2.7.3 Exploring the Privacy-Utility Tradeoff

Anonymization problem can be modeled as the search of a tradeoff between privacy and utility. Machine learning techniques have been used in the past for finding or approximating this tradeoff. Li and Li [2009] propose a framework able to take into consideration the privacy-utility tradeoff. The concepts used are similar to concepts evaluating risk-return tradeoff in financial investment. The evaluation of the privacy is considered as an individual concept and is measured separately for every individual. Utility is considered as an aggregated concept and its measure is performed accumulatively.
In Vinterbo [2004], the privacy problem for health data is formalized as an optimization problem balancing privacy and data utility requirements. The problem of minimizing information loss while satisfying a set of privacy requirements is proved to be NP-hard. A theoretical analysis is provided of the relationship between data utility for machine

learning and privacy disclosure control by generalization.
Mivule and Turner [2013] describe the idea to use KNN classification as an indicator for an optimal balance between privacy and utility.
The idea of finding a tradeoff between privacy and utility in graph data has been described in the literature in Song et al. [2011]. In a more recent paper Wainwright et al. [2012] describes a theoretical method to obtain a precise tradeoff between privacy and utility of data. The optimal convergence rates are learned by using stochastic gradient descent procedures. This method supposes a well known tradeoff function.
Optimization algorithms have already been used in the past by Bayardo and Agrawal [2005] for achieving data privacy through k-anonymization. This paper proposes the use of optimization algorithms to explore the space of possible anonymizations. K-anonymity condition can lead to NP-hard problems and to significant computational challenges. Bayardo and Agrawal [2005] propose the use of optimization techniques in order to find good k-anonymizations in a reasonable time.

2.8 Conclusion

An important research work has been performed in the past in the graph data anonymization field. In most of the approaches described, the privacy risks the datasets is protected from are known in advance. The idea of finding a balance between privacy protection and utility loss has been evocated in the past but no generic method to find this compromise is described. Most of the works deal with simple graphs and only a limited number of papers deal with complex graphs.

Chapter 3

Temporal Graphs Anonymization Issue

This chapter addresses the data anonymization issue from the perspective of graph data subject to decomposition in a multitude of subgraphs. It has been shown that even if the dataset respects anonymization constraints, subgraphs resulting from dataset decomposition can still be de-anonymized. A solution and a framework taking into account this issue is proposed. The proposed solution has been filed as a patent in Hacid and Maag [2014].

Contents

3.1 Introduction

We address in this chapter the sensitive issue of data anonymization in the context of data represented as a graph and subject to data decomposition in a multitude of subgraphs. Anonymization, in its different forms, is quite powerful in a "centralized" and "closed" environment, e.g., when data is in one place and cannot be coupled with other data, i.e. auxiliary information. This mechanism suffers sometimes with the large amount of data, impacting sampling techniques. Our first main contribution was to

propose an improvement of the anonymization process through local, dynamic, and temporal partitioning of a centralized dataset. This makes its reconstruction more complex ensuring a higher robustness of the anonymization process. The proposal was mainly applied on graph data as we focused from the beginning on communication data.

A sensitive case in which data released can be decomposed in a multitude of sequences relates to data coming from call logs like e.g., phone calls, tweets, messages or e-mails exchanged between two parties. This data can be represented as a graph with multiple, oriented and timestamped edges. For phone calls for example, data are recorded in CDRs. A Call Detail Record (CDR) is a file containing metadata corresponding to a telecommunication transaction. As described in Petersen [2002], CDRs are used for accounting and administrative purposes and they contain data as for example information about the caller, the called person, the duration of the call, the starting time of the call, the location information, the route of the call etc. This type of information is protected by law as we already mentioned in Chapter 1.

Research work has been accomplished for CDRs anonymization, but the CDR data was limited to information like for example the initiator of the call and the location data (in Mir et al. [2013]). The destination of the call was, in Mir et al. [2013], not present in the CDRs and no graph structure was considered.

D4D (Data for Development) (Orange) was a contest launched by Orange in 2012 which released four anonymous datasets of CDRs. The CDRs were collected by the Orange Ivory Coast subsidiary with the goal of bringing benefits for the concerned population. In Sharad and Danezis [2013] it is shown that the anonymization strategy used for call logs in the D4D dataset is weak and re-identification based on nodes 1-hop neighborhood degree distribution is possible.

In the past, it has been shown in well known cases such as Netflix or AOL, that by combining released anonymized information with external data, re-identification becomes a reality. Relational and communication data issued from social or telecommunication networks can be represented in very complex structures. Communication interactions for example, may be represented as a graph with multiple oriented and timestamp labeled links. Liu and Terzi [2008] describe the difficulty to anonymize graphs comparing to anonymizing tabular data. Any adversary could obtain any topology structure of the graph for de-anonymization. When the graph has multiple timestamped labeled oriented edges, it is even easier to re-identify an entity than in simple graph data. The more the structure modeling data is complex the easier it will be to re-identify entities inside it.

Related to that, targeted methods to anonymize data in an efficient way for a certain type of attack will be even more complex. It is much harder to perturb a structure around a given vertex in a complex structure. Additionally, the impact of that perturbation can then spread to the whole graph structure.

Recently, differential privacy based methods have been applied to simple graph structures in the interactive mechanism (e.g., Task and Clifton [2012]) or non-interactive mechanism (e.g., Sala et al. [2011], Ahmed et al. [2013], Wang et al. [2013]).
When adding a temporal component to the simple graph, each edge corresponds to a period of time. Next sections describe the vulnerability we identified in data subject to be partitioned in subgraphs as well as one solution we propose.

3.2 Graphs Risk for De-Anonymization based on Subgraphs Partitioning

The problem when anonymizing the initial data as a whole in a static manner is that a malicious user could split the anonymized data in subgraphs, and then de-anonymize each subgraph corresponding to the given timeslot.

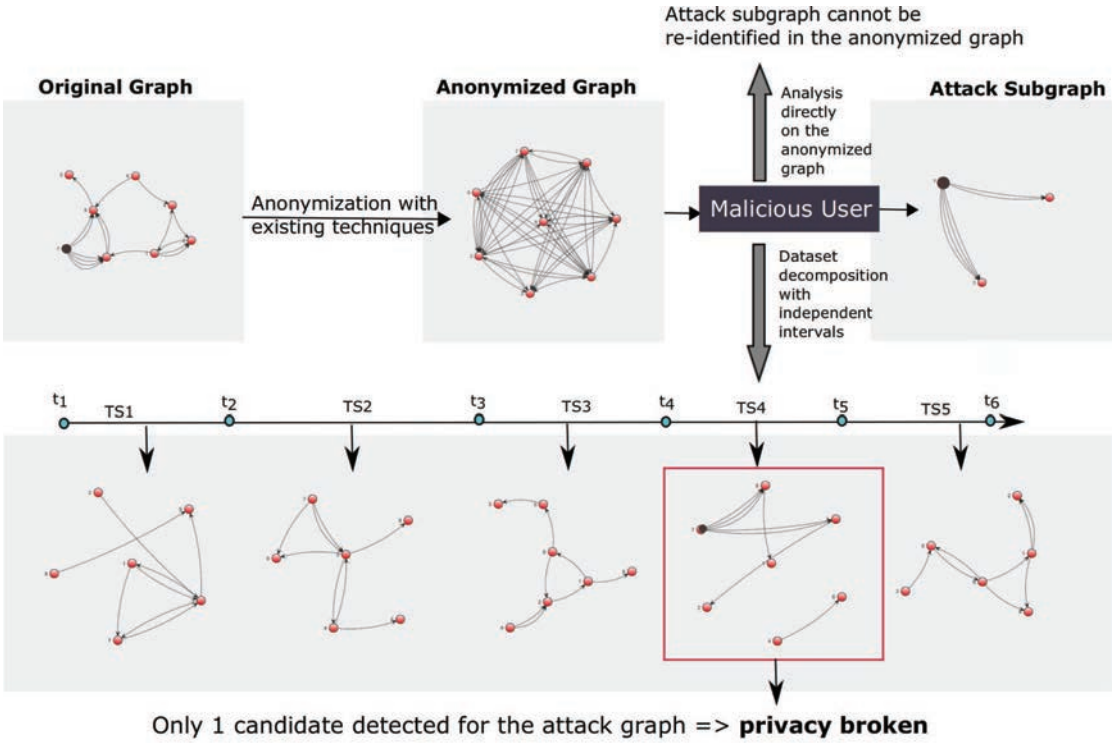


Figure 3.1: Main problem of existing anonymization techniques w.r.t. the time (i.e. dynamics) dimension: even if the anonymized graph respects anonymization constraints (e.g. k-anonymity), when decomposing it in subgraphs, subgraph in timeslot TS4 can be de-anonymized.

We illustrate this in Figure 3.1. In this example the malicious user (the attacker) performs an active attack by embedding a particular subgraph in the initial data. The

malicious user retrieves then the anonymized data, and by using the temporal partitioning in subgraphs, he is capable to localize in a unique manner the embedded attack subgraph in the anonymized data. In this example, the anonymized graph respects anonymization constraints but its projection on the timeslot (t_4, t_5), TS4, does not respect those constraints. An attack inserted in this time interval can be re-identified in the anonymized data.

The main idea to respond to the identified data's vulnerability is based on the "divide and conquer" principle: partition the datasets dynamically, apply the anonymizations locally, and combine the whole.
As mentioned before, the existing techniques all operate on a centralized and "one-block" of data which is anonymized as a whole. We propose here to perform the anonymization on subsets of the dataset independently of the other subsets and then group the anonymized subsets all together for harmonization. This implies to have a strategy for partitioning the initial dataset into several subsets, applying an anonymization on each dataset, and finally recompose the datasets to form one anonymized dataset. By doing this, we ensure to make harder the de-anonymization process by preventing having regular/unique patterns in the datasets which are used to de-anonymize data in general.

3.3 Anonymization by Data Partitioning

Our idea for anonymization is then to introduce a dynamic property inside the anonymization process to prevent uniform anonymizations. By uniform anonymizations we mean anonymization performed on a whole dataset considered as the unit of interest.
The dataset is decomposed in a plurality of subsets and then the chosen anonymization strategy (which could be based on k-anonymization techniques or on ϵ-differential privacy) is applied to each subset.
The proposed approach is composed of three main steps:

1. Decomposition of the dataset into several subsets;
2. Anonymization of each subset independently;
3. Aggregation of the anonymized subsets.

These three steps are illustrated in Figure 3.2. During the first step, the dataset is decomposed into several subsets. We propose to operate a non-uniform decomposition in order to make it more complicated for potential attackers to get insights regarding the decomposition. Thus, we can consider that the decomposition in itself provides an

anonymization parameter which can be added to the classical parameters.
We propose the following strategies to decompose the dataset:

- Independent intervals: in this strategy, we target the division of the initial dataset into n subsets with respect to the density. We defined the density as being the number of calls in a period of time. The particularity of this division is that each time interval is independent of the other and each interval must have the same density in terms of activity. The density includes the amount of users combined with their amount of interactions, e.g., calls.

- Cross intervals: in this strategy, we consider that intervals intersect between them. This means that an interval may have a very small portion which is also included in the next interval. However, it is expected that all the intervals may have the same, or similar, densities in terms of activity.

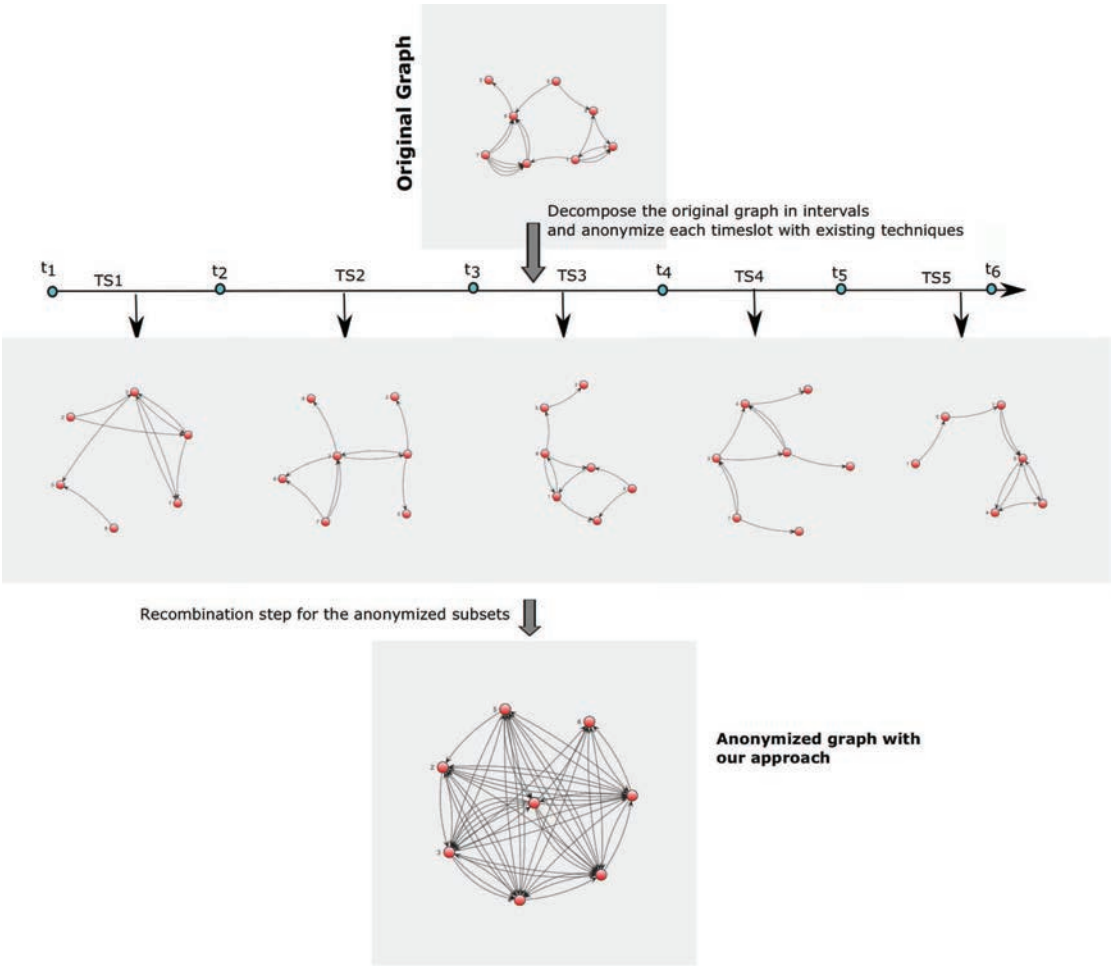


Figure 3.2: Temporal graphs anonymization reinforcement technique, approach in three main steps: (i) data decomposition in subsets (ii) anonymization of each subset independently and (iii) aggregation of the anonymized subsets.

This step produces a set of intervals, which if combined, will reconstruct the whole initial dataset. Once these intervals built, it is then the time to apply locally on each subset, an anonymization technique (as illustrated in 3.2). During the anonymization, any technique can be used to ensure this process. In fact, the idea behind is that an anonymization applied locally becomes much harder to break down since the anonymization works mainly on the available data at a certain time. All state of the art techniques can be used at this stage to perform this local anonymization (e.g., k-anonymity, l-diversity, ϵ-differential privacy).
Finally, a recombination step is performed on the anonymized subsets in order to hide the different decompositions. This recombination is intended to hide the decomposition parameter, preventing to infer the local anonymization.

3.4 System Architecture

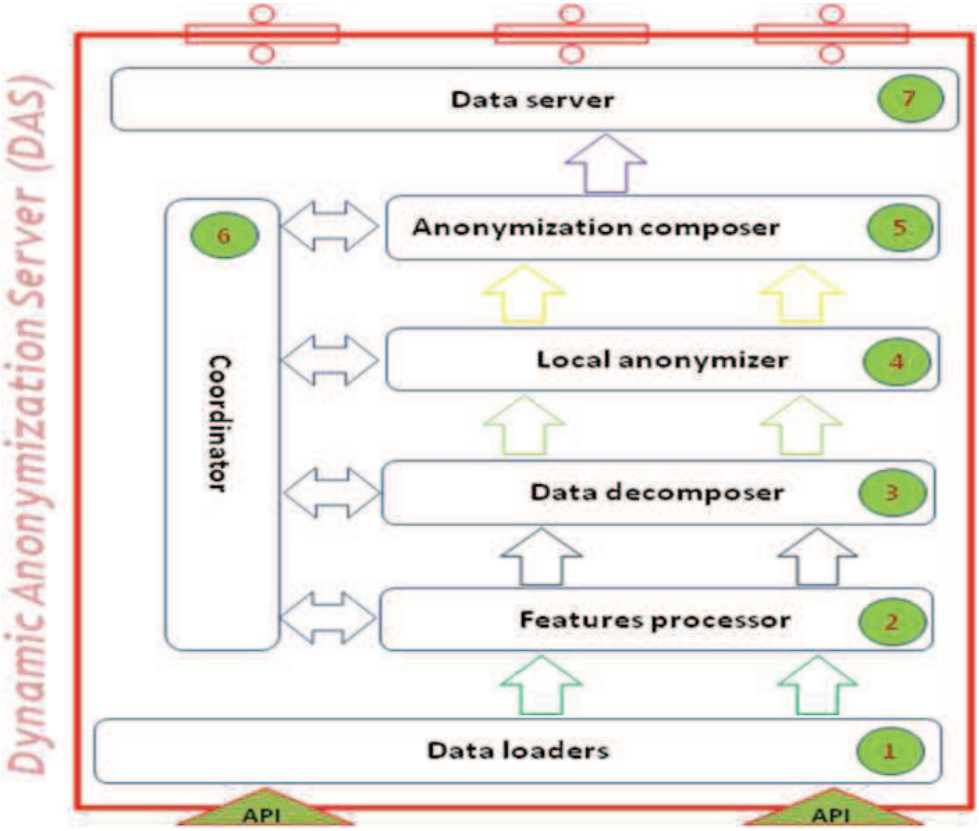


Figure 3.3: General architecture of the dynamic anonymization server (DAS).

We also propose an architecture integrating the described solution the idea being to propose a framework for anonymization taking into account the complexity of the data and its possible decompositions. This framework aims to provide good anonymizations for

complex data in a real environment. We will call this framework "Dynamic Anonymization Server" (DAS). The DAS will be responsible for executing the whole anonymization reinforcement process. From a positioning and exploitation point of view, the framework can fit as a strategic and crucial partner between data owner and the analysis provider. The general architecture of the DAS is depicted in Figure 3.3. The whole system communicates with other systems through APIs which ensure an ability to read data and send outputs to other systems. The input of the system is a set of data (in our case, graph data) and the output is another set of data with the anonymization operated on it. The resulting data is expected to keep a high analytical quality while hiding sensitive data regarding individual and/or businesses. The different components of the proposed system are the following:

1. Data Loaders: these components are responsible for loading data from data owners. They are composed of adapters able to read data under different formats such as relational databases, flat files, spreadsheets, XML, etc. The only role this component ensures then is that of importing data to be anonymized.

2. Features Processor: this component takes as an entry (i) a dataset, (ii) an analytical objective (i.e. associated features such as betweenness centrality for influence, density for communities, etc.), and (iii) a set of information to be hidden. As an output, it provides a value associated to each object in the dataset as a preparation for further computation.

3. Data Decomposer: this component is responsible for decomposing the dataset into several subsets depending on the time dimension. It encapsulates both the strategies described beforehand for the usage of time. As a result, this component provides several subsets of data decomposed either independently (without any intersection) or with some intersections.

4. Local Anonymizer: this component applies a specific anonymization strategy on each subset obtained from the previous component.

5. Anonymization Composer: this component aggregates all the local anonymizations obtained from the previous component. Basically, this component ensures that there is not useless information added into the anonymization and ensures that only one anonymization is provided as an output of the system.

6. Coordinator: this component ensures that the different components communicate between them correctly and ensures that the anonymization is optimal. Optimal here means that the resulting anonymization is not expected to decrease the quality of the expected analysis to be operated on data. It also ensures that no critical personal information is released in the computed final anonymized dataset.

7. Data Server: this is a communication server which is responsible of managing the transmission of data to the analyzer. This is operated through APIs. Intuitively, this component ensures that the access is made by people who have the right to access data through right management processes, passwords, etc.

Figure 3.3 illustrates the idea from a vision where the system is used independently. Intuitively, the system can also be embedded into the data owner actor as a service that he can operate on data.
However, it cannot be positioned at the analyzer level since the objective behind this process is to hide sensitive data from the analyzer, who is a potential attacker on the privacy of data.
Algorithms and analysis embedded in the analyzer part may be very complex. Additionally, the analyzer may not wish to disclose the analysis he performs on data. In the case of advertisement agencies for example, the analyzer does not want to disclose the analysis in order to retain a competitive advantage over a concurrent company.

3.5 Conclusion

Starting from existing anonymization methods, we have concluded that even if the global dataset cannot be de-anonymized, when splitting this dataset into subsets (by using time dimension or other kind of characteristics), we are able to re-identify isolated structures in the resulting subgraphs. An intentional attack inserted in the original dataset on a time interval, can then be re-identified even if the whole dataset has been anonymized. Our first approach described in this chapter was to apply existing anonymization techniques to each resulting subgraph and then to re-compose the dataset with the anonymized subgraphs. In this initial approach, we consider that the DAS contains all the elements allowing it to perform the anonymization process. However, three main issues were identified:

- In real systems, the analytical objective of the dataset is outside of the DAS component. In many cases, data must be externalized because the analysis to be performed on it is too complex. Data owner or data anonymizer cannot perform this analysis. The element “Features processor” able to quantify an analytical objective of the dataset is an external interface and the anonymizer has no knowledge about its internal functioning. We have modeled in the next chapters this component as an external blackbox.
- The set of information to be hidden by the “Features Processor” as described above is a static information. The information revealed by an anonymized dataset

depends on the given dataset and on the external information an Adversary can have access to. Therefore, its quantification should be done dynamically by a module able to evaluate privacy risk for a given dataset. For this reason we have modeled further this module also as an external blackbox.

- The "Features processor" component has as goal to find a tradeoff between the analytical objective and the information to be hidden (privacy risk of the dataset). This component is suitable for machine learning techniques in order to automatically find the tradeoff between utility loss and privacy risk in a given context (subject treated further in this document).

Starting from the points described above, we have defined and implemented a new methodology based on machine learning able to learn the best anonymization function in a given context. This methodology is described in Chapter 4.
My research work had as main challenge the anonymization before data release of call logs, i.e. multiple timestamped graphs. However I have first implemented and evaluated the proposed methodology on simple graphs as described in Chapter 5, mainly for being able to provide a comparison with existing works. Chapter 6 provides the implementation of the methodology on timestamped graphs.

Chapter 4

Anonymization Methodology Based on Machine Learning

This chapter describes a methodology which aims at automatically finding a tradeoff between utility and privacy in a given context. After a general description of the approach, the first part of the chapter gives the notations and definitions used in the document.
In a second part we describe the optimization problem to be solved and the corresponding optimization methods. The methodology proposed in this chapter has been published in Maag et al. [2014].

Contents

4.1 Introduction

Most of the anonymization techniques described in the literature have as main goal to find a compromise between privacy and utility of data as described in Ohm [2010]. When adding perturbation in data, the privacy is better protected but the utility of data is decreasing. The difficulty of an anonymization algorithm is to optimize the balance between utility loss and privacy protection of the data. Starting from this requirement, we propose a novel approach based on machine learning which will consist in **automatically finding a good anonymization procedure**, given a set of possible attacks, and a set of characteristics to preserve. The proposed method aims to automatically adapt to different objectives of utility and potential attacks. This adaptivity is the main difference with existing anonymization methods. This method determines the anonymization procedure based on a set of training graphs that will be used to discover how to both preserve the anonymity given the possible attacks, and to minimize the utility loss i.e. the loss over the characteristics of the graphs. In this sense, the method is generic, and can be applied as soon as the attacks and the measures of anonymization quality and of utility are available during the training phase. In the following we will consider that they are provided as blackboxes. The blackbox corresponding to the utility measures is provided by the external analyzer. The hypothesis is realistic as the third party representing the data analyzer is usually implementing an API allowing the data owner to retrieve the results of the measures performed on the data. The blackbox corresponding to the panel of attacks is also considered as being implemented externally and is supposed to combine a large panel of de-anonymization methods as well as external data retrieved from the Internet as e.g. social networks. This blackbox is implementing different techniques and combines them with data an adversary could have access to in a de-anonymization attempt.

Figure 4.1 shows a general framework for anonymization. Data owner collects data about a certain user (1). The user shares data on the Internet, as for example on social networks (2). Data owner needs for some reason to release data to a third party, to a data analyzer or in some cases to release open data for research purposes (4). As required by legislation but also to keep clients confidence, this data has to be anonymized before release (3). We define an adversary as being a person aiming at re-identifying anonymized data in the released dataset. The adversary could access external data made public by the user on the Internet (5). In combination with the anonymized data, the adversary could then re-identify the identity of the user in the anonymized dataset (6). In most of the cases for which graph data need to be published, analysis to be performed on the released data is either confidential or is too complex to be implemented by the data owner or by the anonymization mechanism. We make the hypothesis that the anonymizer has **no access to the analysis algorithms**. Only the access to the

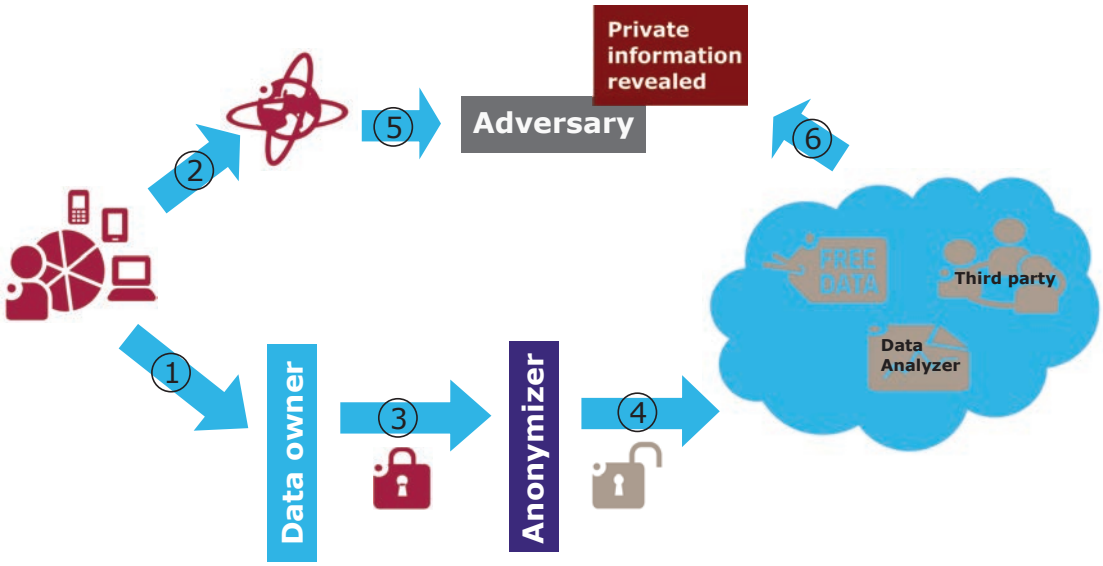


Figure 4.1: Anonymization general framework: private information can be revealed if the adversary combines external data with anonymized data.

analysis result can be obtained, the analysis methods being inside the blackbox and not being accessible. This means that the properties to be preserved in the anonymized graph are unknown to the data anonymizer.

The anonymization problem for data release consists in determining how to modify data such that re-identification cannot be performed and data is still useful for the purpose for which it has been released. Based on these considerations, we model the anonymization problem as an optimization problem with the goal of finding a tradeoff between the loss of data's utility and the gain in data's privacy. Our main idea is to use machine learning on a small part of a dataset in order to automatically find the best anonymization procedure for a given context.

The specific context is represented by the analysis to be performed on data (which is external and not known by the Anonymizer) and the possible attacks, both represented by blackboxes. This implies an objective function to minimize unknown.

Another hypothesis is that the entire data is not known in advance **so that new data can be anonymized with the function learned on the training data.**

4.2 Methodology

With respect to classical approaches to anonymization, we propose a totally different methodology where **the anonymization procedure will be automatically found**. This approach is based on Machine Learning techniques and consists in finding a good anonymization procedure by using *training data* – i.e. training graphs here. The underlying idea is summarized below:

1. Instead of defining a specific anonymization function, the user defines a family of parameterized functions, the parameters controlling the behavior of the anonymization procedure (see chapters 5 and 6 for examples of such families).

2. The system is provided with the ability to measure if a graph is well anonymized, and also a way to compute if the anonymized graph preserves the desired characteristics (utility loss measure). These two measures can usually be computed from blackboxes as explained in chapters 5 and 6.

3. Given a set of training graphs, the machine learning algorithm will test different anonymization functions corresponding to different parameter values and will be able to evaluate both, the quality of the anonymization and the utility loss.

4. The algorithm will be able to find the "best" parameters values, i.e. the values that correspond to an anonymization procedure that obtains the best balance between anonymization quality and utility loss.

Instead of defining a specific anonymization function, we define a family of parameterized functions. Graph dataset to be anonymized is split in a multitude of subgraphs. We use a part (e.g., 10%) of the projected subgraphs as training set to learn which model fits better in a certain context. The learned anonymization function is then applied on the rest of the subgraphs (ex. 90%) corresponding to the testing set.

4.3 Notations and Definitions

Notations described in this chapter mainly focus on graph anonymization problem for simple graph. Additional notations dealing with complex graphs are introduced in Chapter 6. Table 4.1 contains a synthesis of the notations used in the current document.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{C})$ be the graph describing the relational data. Let S be the number of vertices in the graph $\mathcal{G}$ and U the number of edges in graph $\mathcal{G}$. The vertices of $\mathcal{G}$ are defined as $\mathcal{V} = (v_1, ..., v_S)$ with each vertex v_i representing an interaction entity. We denote with d the degree of a certain vertex in the graph.
Edges between users represent interactions. The interactions set is $\mathcal{C} = (c_1, ..., c_U)$ with $c_k \in \mathcal{C}$, $c_k = (o(c_k), d(c_k))$, $o(c_k)$ and $d(c_k)$ being the vertices connected by the edge c_k. Let $\mathcal{G}' = (\mathcal{V}', \mathcal{C}')$ be the anonymized graph. The anonymization function receives as input $\mathcal{G}$ and outputs an anonymized graph $\mathcal{G}'$.

We introduce next the anonymization function f_θ, the utility loss Δ and the privacy risk R used to evaluate the compromise between privacy protection and utility loss.

Table 4.1: Notations Summary Table

Symbol	**Significance**
$\mathcal{G} = (\mathcal{V}, \mathcal{C})$	Graph containing the relational data
$\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{T})$	Graph containing relational timestamped data
$\mathcal{V} = (v_1, ..., v_S)$	Set of vertices in $\mathcal{G}$
S	Number of vertices in $\mathcal{G}$
$\mathcal{C} = (c_1, ..., c_U)$	Set of edges in $\mathcal{G}$
U	Number of edges in $\mathcal{G}$
$\mathcal{G}' = (\mathcal{V}', \mathcal{C}')$	Anonymized graph
f_θ	Anonymization function
$\theta = (\theta_1, ..., \theta_n) \in \mathbb{R}^n$	Parameter vector for function f
n	Number of parameters for function f
Δ	Utility loss
R	Privacy risk
$M : \mathcal{G} \to \mathbb{R}^n$	Measure which applied on a graph $\mathcal{G}$ gives a measure result
$L(\theta, \mathcal{G})$	Approximation of the utility loss when using f_θ on $\mathcal{G}$
$L(\theta)$	The total loss evaluating how much information has been lost by providing $\mathcal{G}'$ anonymized with θ instead of $\mathcal{G}$
$\mathcal{A}$	Set of possible attacks on $\mathcal{G}$
$a \in \mathcal{A}$	Attack
$R(\mathcal{G}, \mathcal{G}', a)$	The probability to re-identify original data from $\mathcal{G}$ in anonymized data $\mathcal{G}'$
$Q(\theta)$	The global privacy risk for all anonymization results using f_θ among all the possible original graphs
f_{θ^*}	Approximation of the best possible anonymization function
$\mathcal{L}(\theta)$	Objective function to minimize
m	Number of sampled graphs $\mathcal{G}_1, ..., \mathcal{G}_m$ used in the training set
λ	Hyper-parameter used for the balance between privacy risk and utility loss
$\mathcal{L}_i(\theta)$	The empirical loss for the graph $\mathcal{G}$ and for an instantiation of the function $f_\theta(\mathcal{G}_i)$
$\mathcal{L}^{emp}(\theta)$	The approximation of the empirical loss corresponding to parameter θ
$\mathcal{D}$	Maximum degree of a graph $\mathcal{G}$
k	Used for the k anonymity evaluation
N	Size of the initial population of θ used for learning
P	Percentage of best parameters conserved by the optimization algorithm
θ^*	Learned parameters vector
T	Number of times the empirical loss is computed (for the Monte-Carlo method)
d	Degree of a given node
p	Number of groups the vertices are split into for the anonymization method
$\mathcal{T}$	Set of timestamps
T_{min}	Minimum timestamp in graph $\mathcal{G}$
T_{max}	Maximum timestamp in graph $\mathcal{G}$
$O : \mathcal{G} \times \mathcal{T} \times \mathcal{T} \to (\mathcal{V}, \mathcal{C}, \mathcal{T})$	Operator used to project graph data on a time interval

4.3.1 Anonymization Function

A data anonymization process will correspond to a stochastic parameterized function f_θ with parameter θ such that f_θ defines a distribution $P(\mathcal{G}'|\mathcal{G},\theta)$ which corresponds to the probability that the anonymization procedure applied on graph $\mathcal{G}$ returns the anonymized graph $\mathcal{G}'$. Let n be the size of parameter θ. The result of $f_\theta(\mathcal{G})$ is thus a graph $\mathcal{G}'$ that is sampled following $P(\mathcal{G}'|\mathcal{G},\theta)$. When applied several times on the same graph $\mathcal{G}$, the result $\mathcal{G}'$ returned by the anonymization function parameterized with the same θ is likely to be different.

4.3.2 Utility Loss

Instead of providing $\mathcal{G}$ to the third-party, we will provide an anonymized version $\mathcal{G}'$ of $\mathcal{G}$. The measures made on the anonymized graph $\mathcal{G}'$ will thus be noisy versions of the desired measures on $\mathcal{G}$ resulting in a loss of information. The utility loss is evaluated according to the analysis to be made on data (which is not known by the anonymizer). The utility loss is the gap between the analysis result on the original data $\mathcal{G}$ and the analysis result on the anonymized data $\mathcal{G}'$. The analyzer will perform a set of analysis on the original graph and the anonymized graph provided by the anonymizer. The loss of information resulting from the use of $\mathcal{G}'$ instead of $\mathcal{G}$ will be accessed by the data anonymizer through an external API provided by the data analyzer.
We define the analysis to be made on data as being equivalent to a certain measure $M(\mathcal{G})$ which applied on a graph $\mathcal{G}$ gives a measure result represented as a real number or as a vector. The gap between the measure $M(\mathcal{G})$ obtained on the original data and the same measure $M(\mathcal{G}')$ obtained on the anonymized data is the loss in the utility of the data due to the anonymization process and is denoted $\Delta(M(\mathcal{G}), M(\mathcal{G}'))$. The total loss in the utility of data when using the anonymization function parameterized with θ corresponds to the result of the utility loss measured among all possible anonymizations as defined bellow:

$$L_M(\theta,\mathcal{G}) = \int_{\mathcal{G}'} P(\mathcal{G}'|\mathcal{G},\theta)\Delta(M(\mathcal{G}), M(\mathcal{G}'))d\mathcal{G}' \tag{4.1}$$

The total loss among all the graphs to be anonymized is the integral of the loss obtained when considering for each graph $\mathcal{G}$ all possible anonymized graphs $\mathcal{G}'$:

$$L_M(\theta) = \int_{\mathcal{G}}\int_{\mathcal{G}'} P(\mathcal{G})P(\mathcal{G}'|\mathcal{G},\theta)\Delta(M(\mathcal{G}), M(\mathcal{G}'))d\mathcal{G}'d\mathcal{G} \tag{4.2}$$

Evaluating the utility loss over all possible $\mathcal{G}'$ is impossible so that we will make use of Monte-Carlo methods (Eckhardt [1987]). Monte-Carlo methods are based on repeated and numerous simulations on random samples according to a given distribution in order to obtain a result close to the one ideally obtained by taking into account all the possibilities. Rather than evaluating the integral, the utility loss is evaluated at a series of random points sampled from $P(\mathcal{G}'|\mathcal{G},\theta)$. Let T be the number of anonymizations used to evaluate the utility loss for a graph $\mathcal{G}$. The evaluation of the utility loss for an anonymization function parameterized with θ becomes:

$$L(\theta,\mathcal{G}) \approx \frac{1}{T} \sum_{(\mathcal{G}'_1,....\mathcal{G}'_T)} \Delta(M(\mathcal{G}), M(\mathcal{G}'_k)) \tag{4.3}$$

The total loss to be evaluated is the utility loss over all the graphs in the dataset is:

$$L(\theta) = \int_{\mathcal{G}} P(\mathcal{G}) L(\theta,\mathcal{G}) d\mathcal{G} \tag{4.4}$$

We use once again the Monte-Carlo method to evaluate the sum over a random series of graphs to be anonymized. Let m be the number of sampled graphs forming the training set. The total loss of the utility can be approximated to:

$$L(\theta) \approx \frac{1}{m} \sum_{(\mathcal{G}_1,....\mathcal{G}_m)} L(\theta,\mathcal{G}) \tag{4.5}$$

The total loss will be approximated by:

$$L(\theta) \approx \frac{1}{m} \sum_{(\mathcal{G}_1,....\mathcal{G}_m)} \left(\frac{1}{T} \sum_{(\mathcal{G}'_1,....\mathcal{G}'_T)} \Delta(M(\mathcal{G}_i), M(\mathcal{G}'_k)) \right) \tag{4.6}$$

The total loss is then a function $L(\theta) \in [0, +\infty)$ reflecting how much information has been lost by providing $\mathcal{G}'$ instead of $\mathcal{G}$.

4.3.3 Privacy Risk

We define an attack $a \in \mathcal{A}$ as corresponding to an algorithm used to evaluate the privacy preservation in the anonymized graph $\mathcal{G}'$. This type of algorithms aim at evaluating if the access to the anonymized graph $\mathcal{G}'$ can lead to a re-identification of certain nodes or edges. Section 2.2.2 contains a state of the art of possible attacks on graph data. Usually the algorithm used suppose that the adversary has an external knowledge about the initial dataset and combines this knowledge with the anonymized dataset in order to re-identify anonymized data.

$\mathcal{A}$ corresponds to a set of possible attacks on the anonymized graph $\mathcal{G}'$. We define the privacy risk for graph $\mathcal{G}$, anonymized graph $\mathcal{G}'$ and attack a as being the probability $R(\mathcal{G}, \mathcal{G}', a) \in [0; 1]$ to re-identify original data in the anonymized data according to a given attack. The global privacy risk is the privacy risk considered for all anonymization results, among all the possible original graphs and with all possible types of attacks as defined hereafter:

$$Q(\theta) = \int_{\mathcal{G}} \int_{\mathcal{G}'} \int_{\mathcal{A}} P(\mathcal{G}'|\mathcal{G}, \theta) P(\mathcal{G}) P(a) R(\mathcal{G}, \mathcal{G}', a) da \; d\mathcal{G}' d\mathcal{G} \tag{4.7}$$

Once again we are going to use the Monte-Carlo methods to evaluate the privacy risk in a series of random points. The lowest R is, the more $\mathcal{G}'$ is a good anonymization of graph $\mathcal{G}$. A privacy risk equal to 0 corresponds to a perfect anonymization according to a given set of considered attacks. R is strictly dependent of the set of considered attacks $\mathcal{A}$.

4.4 Optimization Problem: Balance between Utility Loss and Privacy Risk

As explained before, we consider that the anonymizer and data analyzer are not the same entity. The anonymizer does not implement the analysis to be made on data, but can access the measure of the gap between the analysis performed by the data analyzer on graphs $\mathcal{G}$ and $\mathcal{G}'$. This analysis component is acting as an external blackbox for the anonymizer. The privacy risk evaluation is also considered as an external blackbox capable, for a pair of original graph $\mathcal{G}$ and anonymized graph $\mathcal{G}'$, to return the privacy risk corresponding to a release of $\mathcal{G}'$ instead of $\mathcal{G}$.
Let us consider a particular family of parameterized anonymization functions f_θ with $\theta = (\theta_1, ..., \theta_n) \in \mathbb{R}^n$ a vector of parameters. Our goal is to find the "best" possible anonymization function denoted f_{θ^*} such that:

- f_{θ^*} is robust against the attacks, i.e the R values over generated anonymized graphs are low
- f_{θ^*} preserves the utility measures over anonymized graphs, i.e Δ values are low

In other words, the best anonymization function f_{θ^*} must correspond to a balance between anonymization and utility loss.

Let us define the objective function $\mathcal{L}(\theta)$ as:

$$\mathcal{L}(\theta) = \int_{\mathcal{G},\mathcal{G}',\mathcal{A}} \left(\Delta(M(\mathcal{G}), M(\mathcal{G}')) + \lambda R(\mathcal{G}, \mathcal{G}', a)\right) P(a)P(\mathcal{G})P(\mathcal{G}'|\mathcal{G}, \theta)da \; d\mathcal{G}d\mathcal{G}' \quad (4.8)$$

where λ is a hyper-parameter chosen by hand. The term $\Delta(M(\mathcal{G}), M(\mathcal{G}')) + \lambda R(\mathcal{G}, \mathcal{G}', a)$ measures the balance between the utility loss obtained by using $\mathcal{G}'$ instead of $\mathcal{G}$ and the quality of the anonymization, $\mathcal{L}(\theta)$ is the average over all possible pairs $\mathcal{G}, \mathcal{G}'$ and all possible attacks in $\mathcal{A}$. While $\mathcal{L}(\theta)$ cannot be evaluated, it can be approximated using Monte-Carlo sampling on a training set of sampled graphs $\mathcal{G}_1, ..., \mathcal{G}_m$ where m is the number of graphs used for learning. As explained before, rather than evaluating the integral, the utility loss is evaluated at a series of random points.
Let $\mathcal{L}_i(\theta)$ be the empirical loss for the graph $\mathcal{G}_i$ and for an instantiation of the function $f_\theta(\mathcal{G}_i)$:

$$\mathcal{L}_i(\theta) = \Delta(\mathcal{G}_i, f_\theta(\mathcal{G}_i)) + \lambda \frac{1}{|\mathcal{A}|} \sum_{a_k} R(\mathcal{G}_i, f_\theta(\mathcal{G}_i), a_k) \quad (4.9)$$

The integral to be minimized defined in (4.8) can be approximated by the *empirical loss* defined as following:

$$\mathcal{L}^{emp}(\theta) = \frac{1}{T} \sum_{j=1}^{T} \left(\frac{1}{m} \sum_{\mathcal{G}_i} \mathcal{L}_i(\theta) \right) \quad (4.10)$$

where T corresponds to the number of samples used for the Monte-Carlo method. In the following we will denote θ^* the learned estimation for parameter θ. The learning problem to solve can thus be written as:

$$\theta^* = \arg\min_\theta \mathcal{L}^{emp}(\theta) \quad (4.11)$$

Solving this optimization problem corresponds to finding **the best anonymization procedure** w.r.t the empirical loss $\mathcal{L}^{emp}$. This will be done by using optimization methods described further in section 4.6.

4.5 Summary

All the defined functions are summarized in Figure 4.2. The risk for de-anonymization $R(\mathcal{G}, \mathcal{G}')$ is measured by the blackbox "Panel of Privacy Risks" corresponding to a catalog of possible attacks on the anonymized data. Utility loss $\Delta(M(\mathcal{G}), M(\mathcal{G}'))d\mathcal{G}'$ is obtained from a second blackbox called "Data Analyzer" containing a panel of measures for which data is externally needed. Both blackboxes receive as input original data $\mathcal{G}$ and anonymized data $\mathcal{G}'$ the output being the values $R(\mathcal{G}, \mathcal{G}')$ and $\Delta(M(\mathcal{G}), M(\mathcal{G}'))$ used

by the learning module.

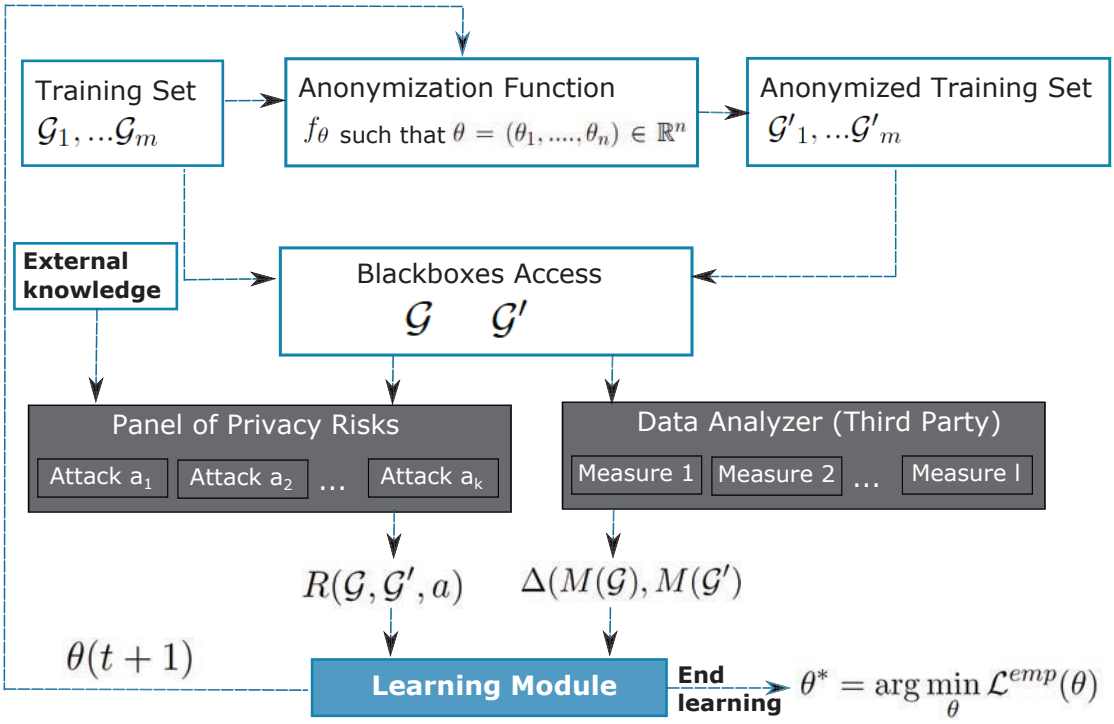


Figure 4.2: Learning process.

The "Learning Module" aims to learn on a training set models capable to behave well on unseen data. The dataset is split in subgraphs forming a training set (e.g. 10% of data) and a testing set (e.g. 90% of data). Training set is formed by m subgraphs $\mathcal{G}_1, ...\mathcal{G}_m$. In order to form this training set, for simple graphs we retrieve chronologically the relations present in the graph. For multiple timestamped graphs, we retrieve in chronologic order the interactions occurred. Only the training set will be transmitted to the blackboxes in its original format in the learning phase. The learning module will dispose of a set of representative graphs $\mathcal{G}_i$ for the dataset and of the access to the blackboxes (and therefore to the values $R(\mathcal{G}_i, \mathcal{G}'_i)$ and $\Delta(\mathcal{G}_i, \mathcal{G}'_i)$ for an anonymized instance $\mathcal{G}'_i$). In practice this access could be materialized by an external API the analyzer exposes to the anonymizer and a set of attacks implemented for learning purpose.
For each anonymization function f_θ candidate for the anonymization process, each graph of the training set $\mathcal{G}_1, ...\mathcal{G}_m$ is anonymized resulting in $\mathcal{G}'_1, ...\mathcal{G}'_m$. The learning module then updates the set of parameters θ in the anonymization function until θ^* is learned after a certain number of iterations. A solution to find the function f_θ corresponding to the minimum of the empirical loss could be obtained by computing the value of the empirical loss in all possible values of the searched parameters θ^*. This could be possible eventually when only one parameter is used. However, when dealing with a certain

number of parameters optimization techniques are mandatory to find the anonymization function minimizing the empirical loss.

The described problem corresponds to a complex multivariante optimization problem. The anonymization function is a stochastic function. The anonymization function being stochastic, the loss is also a stochastic function. Additionally, the loss is not known in advance by the anonymizer, but it is approximated using the access to the blackboxes. For a known loss, classical minimization techniques based for example on gradient descent could have been used. In our case these techniques are not suitable because the loss is not known, therefore we use for learning a set of optimization methods described in the next section.

4.6 Optimization Methods

4.6.1 Estimation of Distribution Algorithm

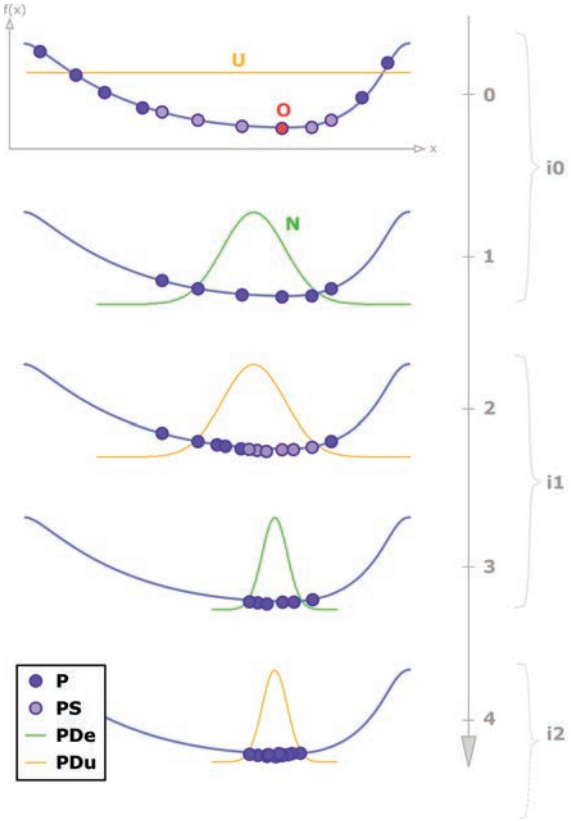


Figure 4.3: Estimation of distribution algorithm: for each step candidates solutions are generated. At step 0, population is initialized from a uniform distribution over admissible solutions (P). The most promising candidates (PS) are selected and a new population is generated at each step following a normal distribution N with the distribution parameters PDe.

Estimation of Distribution Algorithms (EDAs), sometimes called probabilistic model-building genetic algorithms (PMBGAs), are stochastic optimization techniques that explore the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. Hauschild and Pelikan [2011] provides a survey of the different types of EDAs and of their advantages. The principle of a typical EDA algorithm is illustrated in Figure 4.3 (source: Wikipedia [2013]). EDA is initializing the population from a uniform distribution over admissible solutions. The algorithm evaluates the function to optimize $f(x)$. At each step it regenerates new candidates sampling promising candidate solutions from the best candidates selected at the previous step. As showed in the figure, the algorithm ends when the model generates solutions close to the optimal solution. We have made the choice to use EDA for solving the optimization problem for its capacity to converge to a good approximated solution in a limited amount of time. Convergence for this type of algorithms is discussed in Zhang and Muhlenbein [2004] or Wright and Pulavarty [2005]. It is showed that the algorithm is able to find the optimum solution with high probability.

The use of this learning algorithm in our context is detailed further in this document. The implementation of the solution is described in Algorithm 1. In our specific case,

Algorithm 1: Use of EDA in the Anonymization Context

Data: $(\mathcal{G}_1, ..., \mathcal{G}_m)$, access to external blackboxes Δ, R
Result: $\theta_1^*, ..., \theta_n^*$

1 init $Model(t_0)$ to N possible solutions, $0 < \theta_i < \theta_{max}$;
2 $Model(t_0) = (\theta_1, ..., \theta_n)_1, ..., (\theta_1, ..., \theta_n)_N$;
3 init T;
4 init endLearning;
5 **while** *endLearning!=0* **do**
6 **foreach** $\mathcal{G}$ *in* $(\mathcal{G}_1, ..., \mathcal{G}_m)$ **do**
7 **foreach** $(\theta_1, ..., \theta_n)_k$ *in* $Model(t)$ **do**
8 **for** $index \leftarrow 0$ **to** T **do**
9 $\mathcal{G}' = f(\mathcal{G}, (\theta_1, ..., \theta_n)_k)$;
10 $\mathcal{L}(\theta_k) = \Delta(\mathcal{G}, \mathcal{G}') + \lambda \frac{1}{|A|} \sum_{a_k} R(\mathcal{G}, \mathcal{G}', a_k)$;
11 **end**
12 **end**
13 **end**
14 Compute $\mathcal{L}^{emp}$;
15 $BestFitting(\theta)$ = select P% in $Model(t)$ minimizing $\mathcal{L}^{emp}$;
16 mean = Mean($BestFitting(\theta)$);
17 variance = Variance($BestFitting(\theta)$);
18 $Model(t+1)$ = gaussianValues (mean, variance);
19 $endLearning--$;
20 **end**

EDA is initialized to a population of size N of random possible candidate solutions to a given problem between 0 and θ_{max} (lines 1 and 2 of the algorithm). The population is

then scored using a function which in our case will correspond to the balance between utility loss and privacy risks (line 10). For each graph in the training set, we calculate the ranking function $\mathcal{L}(\boldsymbol{\theta_k})$ T times (lines 8 to 11), each iteration corresponding to a result of the anonymization function parameterized with $\boldsymbol{\theta_k}$. $\mathcal{L}^{emp}$ is the mean of all $\mathcal{L}(\boldsymbol{\theta_k})$ obtained among all the iterations and all the graphs in the training set. The candidate solutions $\boldsymbol{\theta_k}$ are ranked according to the result obtained for each one of them for $\mathcal{L}^{emp}$. From the ranked population a truncation selection is performed with a threshold P (for example P could be equal to 40% of the population) and parameters corresponding to the lowest values for $\mathcal{L}^{emp}$ are selected. The algorithm then constructs a probabilistic model which attempts to estimate the probability distribution of the selected solutions. We make the hypothesis that each θ_i in the population is distributed according to a Gaussian model and we regenerate a new population based on the mean and variance of the most promising candidates selected among the initial population (lines 16 to 18). The new solutions are replacing entirely the old population (line 18). The process is repeated a certain number of times ("endLearning" parameter).

The idea behind this algorithm is to focus, at each step, on the parameter space region which corresponds to good anonymization functions. This region is determined by the P best functions found at the iteration. While different choices can be made for the sampling model $\mathcal{M}$, a classical choice is to consider that $\mathcal{M}$ is composed of independent Gaussian densities. It means that, at time t, $\mathcal{M}$ is defined by a set of Gaussian distribution $\mathcal{N}(\mu_i^t, \sigma_i^{t2})$ such that $\theta_i \sim \mathcal{N}(\mu_i^t, \sigma_i^{t2})$.

4.6.2 Genetic Algorithms

Another solution to the optimization, close to EDA, is the use of genetic algorithms. Genetic algorithms (Whitley [1994], Golberg [1989]) are optimization search heuristics. The model is initialized to a random population N of possible solutions to the problem between 0 and θ_{max}, similar to the initial model used for EDA and described in section 4.6.1.

Each candidate solution $(\theta_1, ..., \theta_n)$ has a set of characteristics that can be altered at each iteration. Each iteration corresponds to a new generation of candidate solutions. The objective function corresponding to the optimization problem to be solved is evaluated for each generation. Best candidates are selected and for each individual the genome is changed to form a new generation. The changes in the genome consist in recombinations and random mutations. This technique has been used in a differential privacy context in Zhang et al. [2013].

The algorithm we use is described in Algorithm 2. The first steps of the algorithm are similar to the EDA algorithm (line 1 to 15). In our case, we use for the generation

Algorithm 2: Use of a Genetic Algorithm in the Anonymization Context

Data: $(\mathcal{G}_1, ..., \mathcal{G}_m)$, access to external blackboxes Δ, R
Result: $\theta_1^*, ..., \theta_n^*$

```
1  init Model(t_0) to N possible solutions, 0 < θ_i < θ_max ;
2  Model(t_0) = (θ_1, ..., θ_n)_1, ..., (θ_1, ..., θ_n)_N;
3  init T;
4  init endLearning;
5  while endLearning!=0 do
6      foreach G in (G_1, ..., G_m) do
7          foreach (θ_1, ..., θ_n)_k in Model(t) do
8              for index ← 0 to T do
9                  G' = f(G, (θ_1, ..., θ_n)_k) ;
10                 L(θ_k) = Δ(G, G') + λ 1/|A| Σ_{a_k} R(G, G', a_k);
11             end
12         end
13     end
14     Compute L^emp;
15     BestFitting(θ) = select P% in Model(t) minimizing L^emp;
16     Calculate NoiseRange = 5%* range (BestFitting(θ));
17     foreach i = 0; i < N; i++ do
18         Randomly choose θ_parent1 and θ_parent2 from BestFitting(θ);
19         mean = Mean (θ_parent1, θ_parent2);
20         variance = Variance (θ_parent1, θ_parent2);
21         θ_i = gaussianValues(mean, variance) + randomNoise(NoiseRange);
22         add θ_i to Model(t + 1);
23     end
24     endLearning − −;
25 end
```

of each new child, the combination of two random parents from the best candidates selected (line 18 of the algorithm). Each new child is generated according to the mean and the variance of its parents (line 19 to 21). Noise equivalent to 5% of the candidates scale is added to each new child (line 21). Similar to EDA, the process terminates when certain conditions are met or after a certain number of generations (iterations).

4.7 Conclusion

We have described in this chapter the general methodology we propose for graph anonymization based on machine learning. The robustness and the utility of the anonymized data has been compared with existing anonymization methods for simple graphs. Chapter 5 contains the description of the results obtained when applying the methodology on simple graphs. We have then applied the described methodology on call logs represented as multiple oriented timestamped graphs. The approach as well as the obtained results

are described in Chapter 6.

Chapter 5

Simple Graphs Anonymization

This chapter deals with the application of the methodology described in Chapter 4 on simple graphs. First, the optimization method for simple graphs problem is introduced. The second part contains the experimentation performed on real datasets. In order to instantiate the methodology on simple graphs, a series of privacy risks and utility loss evaluations are defined. Experiments are performed on real datasets coming from Twitter or Amazon and obtained results outperform when compared with baseline. Results from this chapter have been published in Maag et al. [2014].

Contents

5.1 Introduction

Chapter 3 pointed out the weakness of the anonymization when the dataset has a temporal component. In Chapter 4 we have described a new methodology based on machine learning able to adapt to rich datasets. However, most of the existing research work as described in Chapter 2 deals with anonymization of simple graphs structures.
In a first step, I have applied the proposed methodology on simple graph data in order to compare to existing anonymization methods. Therefore, this chapter focuses on the experimentation of the anonymization methodology on non labeled simple graphs for privacy protection against nodes re-identification. Simple graph data representation corresponds to most of the published relational data (e.g. friendship relations issued from social networks).

In the literature, three main types of simple graph anonymization techniques can be distinguished: (i) structural methods based on the *k-anonymity* concept, (ii) randomization techniques and (iii) generalization techniques. Most of the existing anonymization techniques have been proposed for one particular type of attack (e.g. the case of k-anonymity based structural methods described in Liu and Terzi [2008]) or one particular set of characteristics to preserve (e.g. randomization methods with graph spectrum preservation described in Ying and Wu [2008]). They are thus very specific to these strong choices and **cannot be adapted easily to other kinds of attacks and measures**. Concerning the third family of methods based on generalization techniques, the difficulty to analyze local structures limits their utilization mainly because of the use of hypernodes and hyperedges in data representation as described in 2.3.3.

The contributions of this chapter are the following:

- We apply the generic approach described in Chapter 4 to simple graphs data. The methodology is able to discover a good anonymization function, given a set of possible attacks and characteristics to preserve proper to simple graphs. As described previously, the methodology is based on modeling the anonymization problem as an optimization problem that embeds the balance between privacy risk and utility loss.
- We use a learning algorithm (*Estimation of Density Algorithm*) to efficiently find the anonymization function best adapted in a given context.
- We instantiate this algorithm for a generic anonymization procedure based on adding edges in the graphs to anonymize, the quantity of added edges depending on the degrees of the nodes.

- We compare our approach to classical approaches used on simple graphs and show that our method is generic and can automatically adapt itself to different anonymization contexts while baseline methods issued from the state of the art are specific to a particular context.

In the following, section 5.2 contains a brief reminder of the optimization problem from Chapter 4. Section 5.3 contains a brief description of the anonymization method used to evaluate the proposed methodology on simple graphs. Experiments and evaluations are presented in Section 5.6 for two datasets retrieved from Twitter and Amazon.

5.2 Optimization Problem for Simple Graphs

Simple graphs are defined in Berge [1973] as being multigraphs responding to two conditions: (i) they do not have loops and (ii) any two vertices have at most one edge connecting them.
As described in Section 4.4, the learning problem we have to solve is to find best parameters θ^*, for a given parameterized anonymization method f, such as:

$$\theta^* = \arg\min_{\theta} \left(\frac{1}{T} \sum_{j=1}^{T} \left(\frac{1}{m} \sum_{\mathcal{G}_i} \Delta(\mathcal{G}_i, f_\theta(\mathcal{G}_i)) + \lambda \frac{1}{|A|} \sum_{a_k} R(\mathcal{G}_i, f_\theta(\mathcal{G}_i), a_k) \right) \right) \tag{5.1}$$

Solving this optimization problem corresponds to finding **the best anonymization procedure** w.r.t the empirical loss $\mathcal{L}^{emp}$. The goal is to find the parameters θ^* corresponding to the minimum of the sum between privacy risk and utility loss. If this function was a known function, classical machine learning techniques like gradient descent based techniques could have been used. This function is a stochastic function and only its values in certain points can be approximated.
A solution to find its minimum could be obtained by computing the value of the function in all possible values of the searched parameters θ^* and then retrieve the minimum. This could be possible with the cost of time when only one parameter is used. However, when dealing with a certain number of parameters, optimization techniques are mandatory to approximate the minimum of this function.
From the techniques described in 4.6, we have chosen to implement Estimation of Distribution Algorithm - EDA to determine the parameters corresponding to the probability of adding noise in order to minimize the utility loss and to maximize privacy preserving. This optimization method is described in 4.6.1.

5.3 Anonymization Method

As explained previously, our technique relies on a specific family of parameterized anonymization functions. In this chapter, we propose a new generic anonymization procedure based on adding noise in simple graph data. This method shares some common points with the ones described in Hay et al. [2008] or Ying and Wu [2008]. We have made the choice to use a randomization method, as among the three main categories described in the state of the art (Chapter 2), this category consists in a class of methods not strictly related to one type of privacy protection (like the structural methods are).
To limit the modifications of the graph, we choose to add noise by only adding edges in the graph, the vertices remaining the same. Contrarily to classical methods where the quantity of noise is determined by one hyper-parameter which is chosen manually, we propose a generic method where the quantity of noise depends on the degree of the vertices. The idea behind this algorithm is to consider that a good anonymization function will certainly add different level of noise depending on node's degree.
In our approach, we make the assumption that the best quantity of noise could differ among the vertices of the graph according to their degrees. Therefore, we split the vertices into n groups according to their degree and we associate to each group a noise adding parameter $\theta_{add_{degree}}$.
As denoted previously, S is the number of nodes in the graph and U the number of edges. For each pair (v_i, v_j) of non connected vertices (a total of $S^2 - U$) we add an edge connecting them with a probability corresponding to the product $\theta_{Degree_{v_i}} * \theta_{Degree_{v_j}}$ between the two modification parameters corresponding to the vertices degrees.
The proposed approach makes use of n parameters denoted $\theta_1, ..., \theta_n$. Different n values will be tested in the experimental section to explore the ability of our technique to find good anonymization procedures[1]. Each vertex will be associated to one of these n parameters, given its degree. Let $\mathcal{D}$ be the maximum degree of the original graph $\mathcal{G}$. The parameter θ_i will be associated to all nodes of degree d such as:

$$\left|\frac{\mathcal{D}}{n}\right|(i-1) < d \leq \left|\frac{\mathcal{D}}{n}\right| i \tag{5.2}$$

except for the parameter θ_n for which corresponding nodes have $d \leq \mathcal{D}$. The vertices with the lowest degree will be associated to θ_1, while the vertices with the highest degrees will be associated to θ_n, the other vertices being associated to the other θ_i. Given each pair (v_i, v_j) of possible not-already connected vertices, where v_i is associated to θ_i and v_j is associated to θ_j, the probability of adding an edge between v_i and v_j will be $\theta_i \times \theta_j$.

[1] Note that, the goal here is not to propose the "best" possible family of procedures, but to explore the ability of the machine learning technique to automatically find the best function in a family of possible methods.

The resulting anonymization algorithm is given in Algorithm 3. The anonymized graph

Algorithm 3: Anonymization Method: A new edge is inserted between v_i and v_j in the anonymized database with a probability corresponding to the product $\theta_{Degree_{v_i}} * \theta_{Degree_{v_j}}$ of the parameters corresponding to each vertex in the pair according to its degree.

Input: $\mathcal{G}$, θ
Result: $\mathcal{G}'$

```
1  copy data from G to G';
2  select the vertices from G';
3  foreach vertex v_i in G' do
4  |  assign a unique generated identifier to v_i ;
5  end
6  select the vertices from G;
7  foreach vertex v_i in G do
8  |  foreach vertex v_j in G do
9  |  |  if ∄ edge(v_i, v_j) then
10 |  |  |  θ_{add v_i v_j} = θ_{Degree_{v_i}} * θ_{Degree_{v_j}}
11 |  |  |  if generatedBoolean(θ_{add v_i v_j}) then
12 |  |  |  |  insert into G' edge(v_i, v_j);
13 |  |  |  end
14 |  |  end
15 |  end
16 end
```

$\mathcal{G}'$ is initialized to the original dataset (line 1). The first part of the algorithm (lines 3 to 5) consists in a naive anonymization, i.e. the real identifiers are replaced with a new numerical identifier. If the dataset used for experiments is already naively anonymized, which is the case of most of the datasets, this part of the algorithm is not useful.

For each possible pair of vertices in the original dataset (two loops "for" at lines 7 and 8 of the algorithm), if the vertices are not already connected (line 9), a boolean is generated with the probability $\theta_{Degree_{v_i}} * \theta_{Degree_{v_j}}$. If the generated boolean is true (line 11), then a new edge is created between v_i and v_j.

Note that, when considering $n = 1$, the probability of adding an edge does not depend on the considered vertices, and thus the obtained method is a random anonymization method with a uniform parameter similar to the one used in Ying and Wu [2008].

In order to evaluate the proposed generic approach, we will compare our method with baseline models in different configurations. Each configuration will be associated to one or many possible attacks, and to one or many possible measures to compute on the graphs. These different configurations will allow us comparing the methods in different situations that can be met in real applications and aim at showing the genericity of the proposed algorithm.

In the following, sections 5.4 and section 5.5 describe a catalogue of possible attacks and utility measures used for evaluation. We then describe the baseline method allowing

us to compare to the state of the art in section 5.6.1 and detail the used datasets corresponding to the graphs to anonymize in section 5.6.2. Results are presented and commented in section 5.6.

5.4 Privacy Risks

We consider for evaluation of the privacy risk R two types of attacks. The first type of attack (vertex degree attack) is a reference in the simple graph anonymization field. The second attack (based on the knowledge on one's neighbors) is associated to social network data disclosure and to the information a possible adversary has access to.
We propose to focus on the following risks: (i) k-degree anonymity (RD-k) with $k = 5$ and $k = 20$. (ii) k-neighborhood degree anonymity based on neighbors degree sequence (RN-k) with $k = 5$ and $k = 20$.
These two types of risk are based on the k-anonymity notion described in Liu and Terzi [2008]: *A certain structure is k-anonymous with respect to a structure query if there exist at least* $k - 1$ *other structures that match the given structure query.*

5.4.1 k-Degree Anonymity

We next define a privacy risk estimation method based on one type of attack inspired from the existing work in anonymization field based on graph structural attacks.
k-anonymity condition is applied when dealing for example with nodes degrees, nodes neighborhoods or groups degree sequence for nodes. Based on this concept, we evaluate in a first approach the anonymized graph according to its conformity to the k-anonymity condition for the nodes degree.
The k-degree anonymity is based on a degree attack described Liu and Terzi [2008] and evaluates the number of nodes re-identifiable in $\mathcal{G}'$ by their degree. The definition of the associated risk denoted R_{RD-k} is thus:

$$R_{RD-k}(\mathcal{G}_i, \mathcal{G}'_i) = \sum_{v_j \in \mathcal{G}_i} \frac{1}{n_j} \tag{5.3}$$

where $n_j < k$ and n_j = number of vertices in $\mathcal{G}'_i$ with the same degree as v_j, including v_j.
We generate a hashtable for each graph containing as key the degree present in the graph and as values the list of vertices having the related degree as illustrated in Figure 5.1. In the given example the graph conforms to the 2-anonymity condition according to the vertices degree. However, the associated graph would not conform to the 3-anonymity

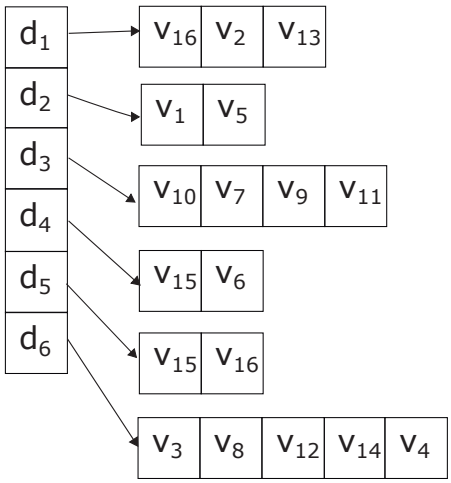


Figure 5.1: Vertices degree hashtable: it stores for each degree d_i present in the graph the corresponding list of vertices of degree d_i.

condition for the vertices degree. Indeed, for the degrees d_2, d_4 and d_5 there are only 2 vertices with similar degrees.

We have based our privacy evaluation algorithm on these considerations. Algorithm 4 describes how we evaluate the privacy risk. We are using two hashtables objects as the one illustrated in Figure 5.1 to store the vertices degrees. These hashtables are initialized at lines 1 and 2 of the algorithm. A threshold k is given as input at the beginning of the algorithm and it will be used for the k-anonymity based privacy level obtention.

- For each node in the original dataset (loop "for" at line 4) , we are analyzing its degree. If the degree does not exist in the hashtable "verticesOriginal", a new list of corresponding vertices is created (line 7). Otherwise, the existing vertices list is retrieved (line 9). The vertices list is updated with the new vertex (line 11) and the global hashtable is updated with the new structure (line 12).

- In the second step of the algorithm for each degree entry in the degree hashtable of the original graph (loop "for" at line 14 of the algorithm), we are comparing (line 19) the vertices list (line 15) with the one corresponding to the same degree (line 18) in the anonymized degree hashtable. If in the original dataset there are less than k corresponding vertices to one degree (line 19), we consider that a risk for privacy infringement exists. If in the anonymized data the corresponding list for the degree has also less than k elements (line 20), we consider that a privacy infringement occurred for the common elements between the two lists. For each element in common (line 22), we increment the value of the global privacy infringement risk. At this level, we are using the k-anonymity notion when incrementing

Algorithm 4: k-Degree Based Privacy Risk Estimation

Input: $\mathcal{G}$,$\mathcal{G}'$, k parameter
Output: Privacy risk estimation

```
1  Hashtable verticesOriginal = null;
2  Hashtable verticesAnonymized = null;
3  initialize privacyRiskSum = 0;
4  foreach vertex v_i in V do
5  |  calculate d_i;
6  |  if d_i is not in verticesOriginal then
7  |  |  Hashtable verticesList = null;
8  |  else
9  |  |  verticesList = verticesOriginal.getValue(d_i);
10 |  end
11 |  insert v_i in verticesList;
12 |  insert (d_i, verticesList) in verticesOriginal ;
13 end
14 foreach d_i in verticesOriginal do
15 |  verticesListOG_i = verticesOriginal.getValue(d_i);
16 |  verticesListAG_i = verticesAnonymized.getValue(d_i);
17 |  sizeDegreeOG_i = size(verticesListOG_i);
18 |  sizeDegreeAG_i = size(verticesListAG_i);
19 |  if sizeDegreeOG_i < k then
20 |  |  if sizeDegreeAG_i < k then
21 |  |  |  foreach v_j in verticesListOG_i do
22 |  |  |  |  if v_j is in verticesListAG_i then
23 |  |  |  |  |  privacyRiskSum = privacyRiskSum + (1/sizeDegreeOG_i)
24 |  |  |  |  end
25 |  |  |  end
26 |  |  end
27 |  end
28 end
```

the privacy risk value. We are considering the privacy infringement as being invers proportional to the number of vertices associated to the degree for which the privacy infringement risk exists.

We are not evaluating the anonymized graph as a standalone graph according to its k-anonymity. The anonymity is evaluated by comparing the vertex sequences for each degree between the anonymized graph and the original one. The k-anonymity notion is used for the evaluation of the privacy risk. Indeed, we are considering that the privacy risk is increasing when identifying fewer nodes associated with a certain degree.

5.4.2 k-Neighborhood Anonymity

The evaluation is similar to the k-degree privacy evaluation; the main difference is that instead of using the node degree for the nodes classification, we are using the neighbors degree sequence. This kind of attack has been described in Zhou and Pei [2011].
To evaluate the graph privacy according to the neighbor's degree sequence, we generate for each vertex the corresponding sequence in a descending order of the degrees. A hashtable containing all the degrees sequences as keys and an array with the corresponding nodes as values is generated.
The privacy is evaluated similar to the evaluation described in 5.4.1 corresponding to the k-degree anonymity privacy risk. The privacy is incremented when the same sequence of vertices corresponding to the same neighbors degree sequence, of size less than k is found both in the original graph and in the anonymized graph.
Privacy values obtained are normalized with the worst possible values for each graph. This "worst value" is obtained when calculating the privacy risk for a $\mathcal{G}'$ equal to $\mathcal{G}$.
The k-neighborhood degree anonymity is based on a family of attacks (Hay et al. [2008]) for the particular case where the adversary knowledge is about the neighbors degree sequence of a node. The privacy risk corresponds to the number of nodes re-identifiable in $\mathcal{G}'$ by their neighborhood sequence (the ordered sequence of their neighbors degrees). The definition of the associated risk denoted R_{RN-k} is:

$$R_{RN-k}(\mathcal{G}_i, \mathcal{G}'_i) = \sum_{v_j \in \mathcal{G}_i} \frac{1}{n_j} \tag{5.4}$$

where $n_j < k$ and n_j = number of vertices in $\mathcal{G}'_i$ with the same neighborhood degree sequence as v_j, including v_j.

5.5 Utility Loss Evaluation

A panel of utility metrics used in graphs can be found in Song et al. [2011]. Three measures are used for this, inspired from the literature to be computed on the anonymized graphs. The choice of these particular metrics has been guided by their use in influence analysis which is an important field of social networks mining mainly for marketing purposes. These measures will be used as blackboxes to test if our method is able to adapt to different utilities, and thus if our automatic anonymization procedure can preserve important characteristics of the input graphs. The three measures are described below.

5.5.1 Clustering Coefficient Based Utility Loss (CC)

Clustering Coefficient is a measure used relatively often in the case of social networks analysis. Connectivity between users forming a group is an important notion which, depending of the context, should be preserved in the anonymized data. Community detection is an important component of the social network analysis and is based, among other measures, on Clustering Coefficient measure on the social graph. We are using here the local clustering coefficient corresponding to each vertex. This notion has been introduced by Watts and Strogatz [1998] in the work aiming to determine if a graph is a small world or is not. Local Clustering Coefficient aims to quantify how close a vertex's neighbors are to being a clique. It is defined as being the fraction of the existing edges between the vertex's neighbors divided by the totality of possible edges between the vertex's neighbors. Clustering coefficient notion is described also in Newman [2003], Barrat and Weigt [2000] or Song et al. [2011].
Figure 5.2 illustrates three graphs.

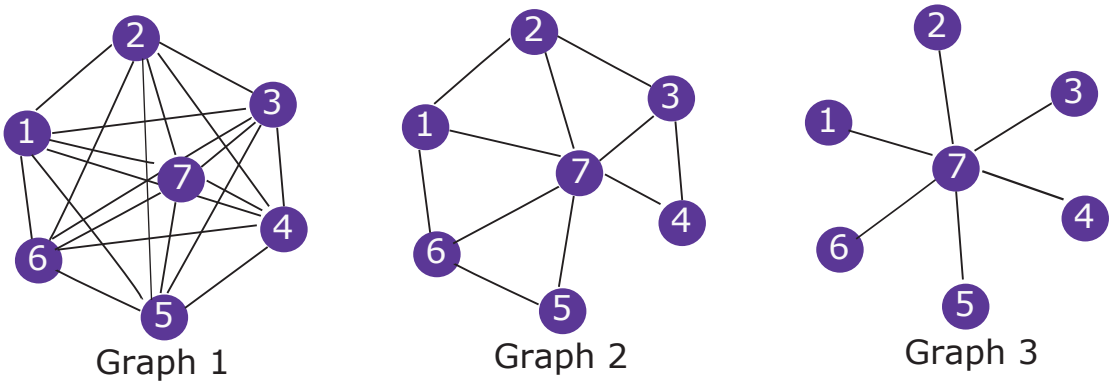


Figure 5.2: Example for local clustering coefficient: three graphs used to illustrate how the clustering coefficient is computed.

The total number of possible edges between the six neighbors of the node 7 is:

$$Edges_{max} = \frac{6(6-1)}{2} = 15 \tag{5.5}$$

For the graph 1, the local clustering coefficient of node 7 is:

$$CC(7)_{Graph1} = \frac{15}{15} = 1 \tag{5.6}$$

and it corresponds to a clique. For the graph 2, the local clustering coefficient of node 7 is:

$$CC(7)_{Graph2} = \frac{5}{15} \approx 0.3 \tag{5.7}$$

For the graph 3, the local clustering coefficient of node 7 is:

$$CC(7)_{Graph3} = \frac{0}{15} = 0 \tag{5.8}$$

corresponding to a situation where the neighbors do not know each other, which is rarely the case when dealing with social networks data.

We aim at modifying the graph for anonymization with a low impact on the clustering coefficient. The Δ value corresponding to this measure between the original dataset $\mathcal{G}$ and the anonymized dataset $\mathcal{G}'$ is denoted Δ_{CC}. This value is defined as being the sum of the absolute difference for each vertex in the original graph and in the anonymized graph:

$$\Delta_{CC}(\mathcal{G}, \mathcal{G}') = \sum_{v \in \mathcal{G}_i, v' \in \mathcal{G}'_i} |CC(v) - CC(v')| \tag{5.9}$$

v' corresponding to the node v in the anonymized graph.

5.5.2 Page Rank Based Utility Loss (PR)

Page Rank is the algorithm used by Google to rank websites when displaying search results in its search engine. The rank value obtained by the algorithm corresponding to a page, indicates the importance of that page. In Page Rank applied to WWW, a graph is created based on the existing links in the web: each page is a vertex and a link is created between two pages if one page references the other one. A page is considered as being important if it receives more links from other pages.

Page Rank was originally designed for oriented graphs, but it also can be applied on undirected graphs, which is our case. Page Rank applied on undirected graphs is close to the degree distribution of the graph as mentioned in Perra and Fortunato [2008]. Starting from the hypothesis that a vertex is more influential if it is linked with an influent vertex, Page Rank algorithm has been used for influence analysis in collaboration networks and is one of the basic measures used for Social Network Analysis (SNA).

This utility measure described in Brin and Page [1998] is based on the Page Rank values of the nodes of the graph and it has been recently used to identify influential nodes. The loss associated with this measure Δ_{PR} corresponds to the sum of the absolute difference for each vertex between Page Rank score in the original graph and the anonymized graph:

$$\Delta_{PR}(\mathcal{G}, \mathcal{G}') = \sum_{v \in \mathcal{G}_i, v' \in \mathcal{G}'_i} |PR(v) - PR_(v')| \tag{5.10}$$

5.5.3 Two-hop neighborhood based utility loss (THN)

This utility measure evaluates the number of two-hops neighbors that a vertex is capable to reach. This type of information is necessary when analyzing information diffusion in an online social network (as for example in Guille and Hacid [2012]). The number of nodes that can directly be reached by a given vertex after only two hops is one possible metric used for collaboration networks or social networks analysis. The influence value computed for a dataset could directly depend on this value.
We consider that a modification in the number of two-hops neighbors could significantly impact the analysis made on data. Each vertex is assigned with a score corresponding to the number of two-hops neighbors it is capable to reach. The corresponding loss Δ_{THN} is the sum of the absolute difference for each vertex between score in the original graph and the anonymized graph:

$$\Delta_{THN}(\mathcal{G}, \mathcal{G}') = \sum_{v \in \mathcal{G}_i, v' \in \mathcal{G}'_i} |THN(v) - THN(v')| \tag{5.11}$$

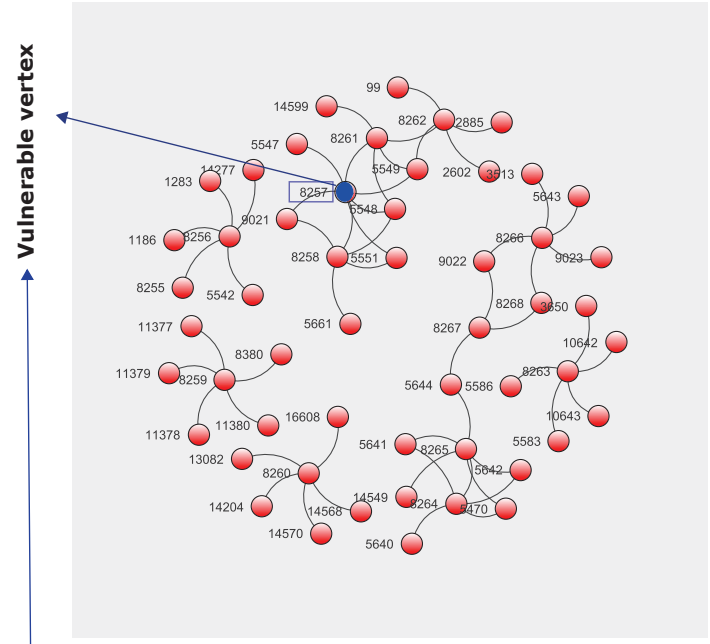
In order to give an equal importance to the different measures in the optimization problem, we will consider normalized versions of these measures - i.e. the measures have been scaled between 0 and 1. The normalization is done against the worst value obtained in a given context.

5.6 Experiments

This section describes the experiments and the results obtained when testing the instantiation of the methodology on simple graphs resulting from real datasets. The algorithms have been developed in Java. In order to evaluate the Clustering Coefficient measure, we have used the implementation from the library Jung (more information can be found at JUNG Library). JUNG (the Java Universal Network/Graph Framework) is a Java library providing tools to model, analyze and visualize graph data. The clustering coefficient measure is implemented in the "Metrics" package of the library and it returns a map between the vertices of the graph and their corresponding clustering coefficients. Similar to the Clustering Coefficient computation, in the experimentation part we have used the library Jung (JUNG Library) to compute Page Rank corresponding to each vertex in the graph.

5.6.1 Baseline (BL)

As a comparison, we have used a state of the art method for graph anonymization based on k-degree anonymization and described in Liu and Terzi [2008]. The graph is k-anonymous with regard to its vertices degrees, if for each vertex degree there are at least $k-1$ others vertices with the same degree.



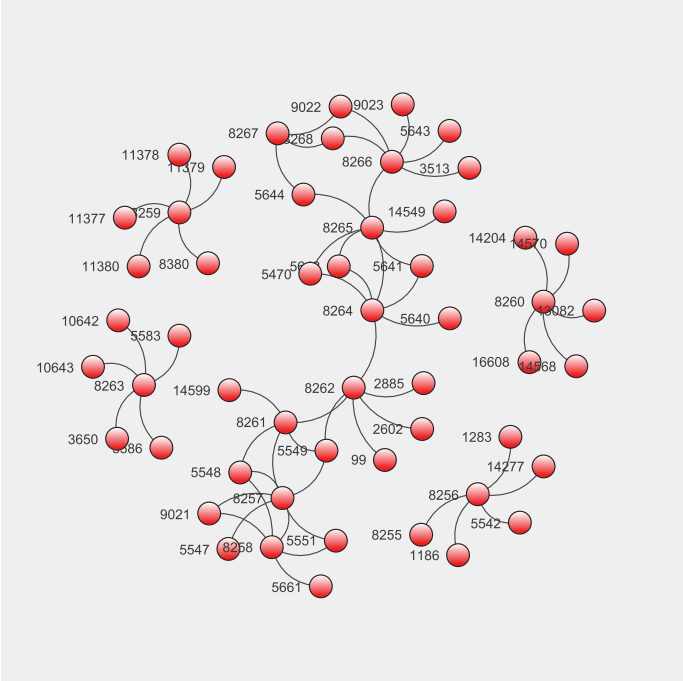
Original graph degrees sequence:

7	8257
6	8265, 8262, 8261
5	8266, 8264, 8263, 8260, 8259, 8258, 8256
3	5549, 5548, 8267
2	5642, 9021, 5641, 5470, 5551, 8268, 5644, 9022
1	11380, 5640, 5547, 2602, 5542, 13082, 2885, 14549, 16608, 11379, 11378, 3513, 11377, 8380, 14570, 8255, 5661, 1283, 10643, 10642, 14568, 14599, 1186, 14277, 5586, 99, 5583, 3650, 14204, 9023, 5643

Figure 5.3: Baseline: original graph and the corresponding vertices degree hashtable; node 8257 is the most vulnerable in the graph according to its degree.

This anonymization solution works in two steps for the obtention of a k-degree anonymous graph and is a reference in the literature when anonymizing data according to the k-degree de-anonymization risk. In a first step a new degree sequence for the graph to be anonymized is constructed such that this sequence is k-anonymous and such that the degree anonymization cost is minimized. The anonymization cost is quantified as the number of edges added in the graph by the algorithm. In a second step, given

the new degree sequence, a graph is constructed (if possible) maximizing the structural similarity with the initial graph.



Anonymized graph degrees sequence:

7	8257, 8265, 8262
6	8261, 8266, 8264
5	8263, 8260, 8259, 8258, 8256
3	5549, 5548, 8267
2	5642, 9021, 5641, 5470, 5551, 8268, 5644, 9022
1	11380, 5640, 5547, 2602, 5542, 13082, 2885, 14549, 16608, 11379, 11378, 3513, 11377, 8380, 14570, 8255, 5661, 1283, 10643, 10642, 14568, 14599, 1186, 14277, 5586, 99, 5583, 3650, 14204, 9023, 5643

Figure 5.4: Baseline: anonymized graph according to the k-degree anonymity, example for $k = 3$. Nodes of high degrees have been modified (8265 and 8262) in order to acquire the same degree as 8257.

Figure 5.3 illustrates the original graph being anonymized according to the k-degree anonymity for $k = 3$. The original graph presents vulnerability for the node 8257. Figure 5.4 illustrates the anonymized graph for $k = 3$. Nodes of high degrees have been modified (8265 and 8262) in order to acquire the same degree as 8257. In the resulting anonymized graph, there are 2 other vertices with the same degree as 8257. Note that other anonymization techniques could be used, but to the best of our knowledge, most of the existing methods would also be specific to a particular privacy risk and utility loss.

5.6.2 Datasets

We have used for evaluation two naively anonymized datasets: one extracted from Twitter and the second one extracted from Amazon. Both datasets have also been anonymized based on the baseline method with k=5 (5-degree anonymization) and k=10 (10-degree anonymization).

5.6.2.1 Twitter Dataset

The Twitter dataset contains around 1,530,000 anonymized messages corresponding to a period of several months between April 2009 and November 2009.
We consider that users A and B are linked if they exchanged messages containing "@user" indication. Based on this consideration, we have generated from the anonymized crawled data the graph constructed from Twitter messages. We have generated 40 graphs of 10000 edges each corresponding to 40 different time sequences. We use a proportion of 10% of the generated graphs for learning. The method is then evaluated on the rest of the graphs.

5.6.2.2 Amazon Dataset

The second dataset corresponds to an Amazon product co-purchasing network collected in March 2003 (Leskovec et al. [2007]). The network was collected by crawling Amazon website. It is based on "Customers Who Bought This Item Also Bought" feature of the Amazon website. If a product i is frequently co-purchased with product j, the graph contains an edge from i to j.

5.6.3 Results

We first (Section 5.6.3.1) analyze the results obtained with the simplest possible anonymization function with only 1 parameter ($n = 1$). This first set of experiments allows us to check that the use of EDA is a good optimization solution. We then explore the quality of the obtained anonymization functions for different numbers of parameters – Section 5.6.3.2. The genericity of the proposed algorithm and its ability to adapt to different anonymization contexts – i.e. different privacy risks and utility losses – is evaluated in Section 5.6.3.3.

5.6.3.1 Simple Anonymization Algorithm – $n = 1$

When the number of parameters of the anonymization function is low – $n = 1$ here – one possible way to optimize the empirical loss defined in Equation 4.10 is to make an exhaustive parameter value exploration which consists in testing all the values using a grid search.

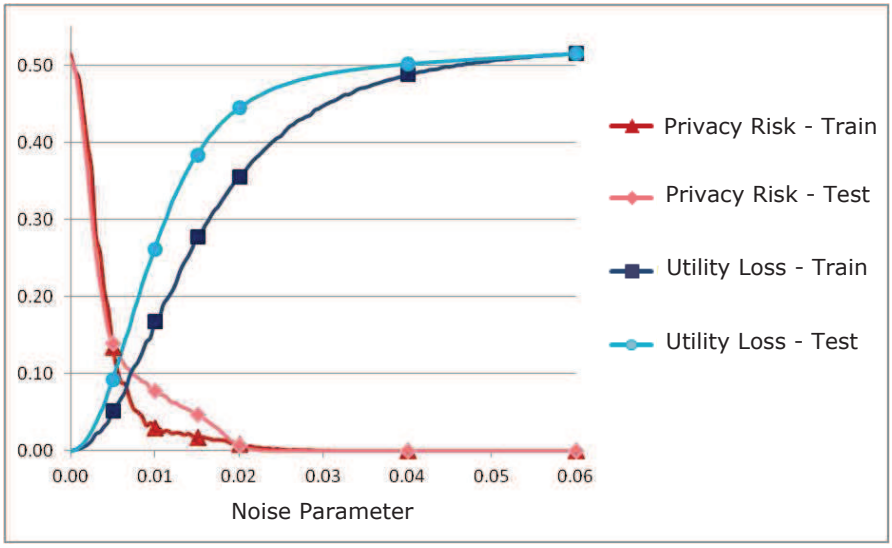


Figure 5.5: Privacy risk and utility loss variation: training set and test set means when performing an exhaustive parameter value exploration.

We pick one measure (CC) and one privacy attack (RN5) among those described in Sections 5.4 and 5.5 to show in Figure 5.5 the function's behavior according to the noise parameter variation. The privacy risk R_{RN5} and the utility loss Δ_{CC} are illustrated for different noise level values θ_1 on the training and testing sets. This corresponds to a classical anonymization configuration where one wants to avoid attacks based on 5-neighborhood anonymity, while being able to compute the Clustering Coefficient on the resulting anonymized graph. One can see that, the higher the noise level, the lower the privacy risk is and the higher the utility loss. This is quite intuitive and illustrates the privacy risk versus utility loss dilemma since it shows that adding a lot of noise tends to make the anonymization more robust, but greatly damages the utility one wants to compute. The best anonymization technique corresponds to the minimum of the Empirical Loss, which is illustrated by the two crosses in Figure 5.6.

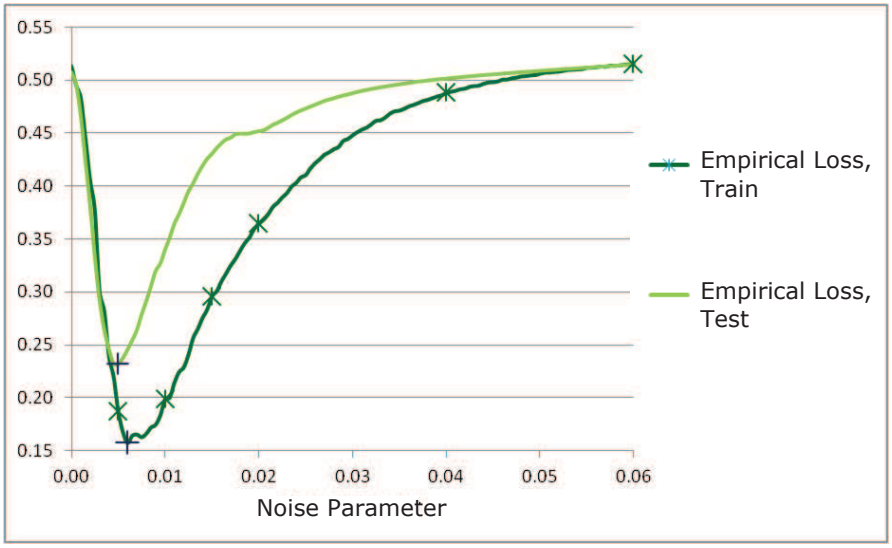


Figure 5.6: Empirical loss evolution when adding noise: training set and test set means when performing an exhaustive parameter value exploration.

The learning technique we propose aims at finding this minimum automatically on the training set.We evaluated the utility loss and privacy risk on the 5-degree and 10-degree anonymized graphs. As expected, the k-degree anonymization is providing a perfect anonymization when dealing with RDk privacy evaluation. For the remaining two methods (RN5 and RN20) the anonymization is not robust as illustrated in Table 5.1.

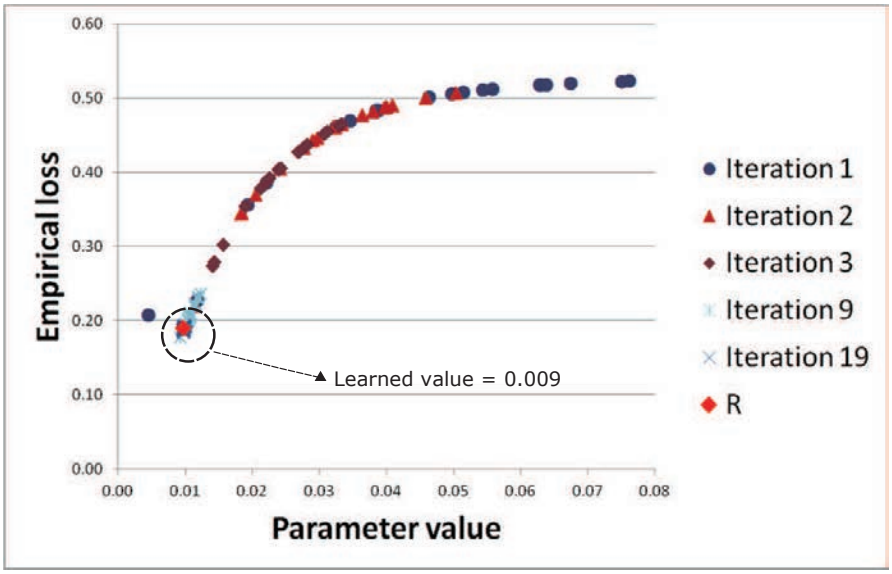


Figure 5.7: EDA behavior with one parameter ($n = 1$). At the first step of the algorithm the population is initialized to a uniform distribution over admissible solutions. After 19 iterations the learned value is close to the optimum one.

Figure 5.7 allows us to analyze the behavior of the EDA algorithm when $n = 1$. The points illustrate the different candidates that have been generated considering that

$N = 20$ and $P = 30\%$. We can see that the algorithm is able to find a parameter value equal to 0.009 (point R on the figure) – that is close to the parameter corresponding to the "real" minimum of the empirical loss (represented by the cross of the training set in Figure 5.6), showing the ability of the algorithm to find a good anonymization procedure.

When dealing with one parameter, it is possible in a limited time to calculate the sum behavior in almost all possible values of the parameter as in this example (only 1000 points have to be analyzed). However, when dealing with several parameters or when the result needs to be obtained in real time, the use of EDA optimization algorithm is mandatory.
Figure 5.7 shows the EDA behavior corresponding to the same privacy risk, utility loss and dataset represented in Figure 5.6 (RN5, CC and Amazon dataset). We randomly initialized the admissible solutions for the add probability between 0 and 0.1 to 20 values. The algorithm converged here in 18 steps.
The parameter corresponding to the minimum of this empirical loss corresponds to the best parameter choice. This parameter obtained by using EDA algorithm is equal to 0.009 and is very close to the one obtained when using the exhaustive method.
This result is encouraging for the method application in the case of 2 or more parameters.

5.6.3.2 EDA Learning With Multiple Parameters

Figure 5.8 shows an example of EDA behavior when finding minimum of the empirical loss when using two parameters (in this example parameters corresponding to add and delete edges operations). In this example, the utility loss is computed based on clustering coefficient and the privacy is evaluated by using risk neighborhood method.
Now, we consider the evaluation when using the anonymization method described in Section 5.3. We then evaluate the learning algorithm for different possible number of parameters $n = 1, 2, 5$ and 10. The EDA is used on the training set, and results in Table 5.1 correspond to the evaluation on the testing graphs that are not used during training. The privacy risk used for learning is the average of $RD - 20$, $RN - 5$ and $RN - 20$ risks corresponding to the 2 attacks presented in Section 5.4, and the utility loss used for learning is the average of the 3 losses presented in Section 5.5. For comparison we also illustrate the values for RD5 for which the baseline anonymization is perfect.

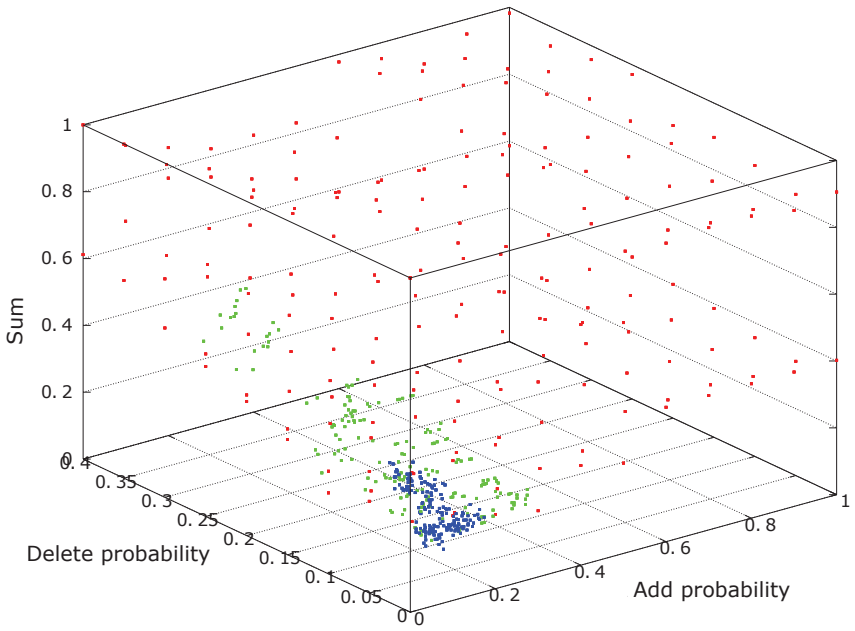


Figure 5.8: EDA behavior with two parameters ($n = 2$). EDA is initialized to a large set of possible candidates (red points). After several iterations, EDA converge to the optimal solution represented by the blue points.

The results correspond to a configuration where one wants to anonymize considering multiple possible attacks, and to preserve multiple possible characteristics. This is the case in most of the real cases as the external attacks can be very diverse and the analysis to be made on data is complex and associates a multitude of measures.

Table 5.1 contains the evaluation of the anonymization method corresponding to learned 1, 2, 5 or 10 parameters for the two datasets. Parameters are learned on 10% of the two datasets and the table contains the evaluation of the learned parameters on the remaining 90% of the datasets.

First, when considering the last column which corresponds to the balance value between privacy and utility loss, we can see that the higher the number of parameters we consider, the better the result is. It means that our learning algorithm is able to learn complex anonymization procedures, and these complex procedures result in better anonymization results than simpler ones. When comparing the best obtained methods ($n = 10$) with the baseline models on the Amazon dataset, we can see that the learned procedure is able to be as good as BL in term of $RD - 5$ privacy (for which BL has been proposed) but it is much better for the other privacy measures ($RD - 20$, $RN - 5$ and $RN - 20$) that are not considered by BL. This shows the ability of our algorithm to consider simultaneously different privacy measures. Moreover, the results corresponding to the Twitter dataset show that the utility losses obtained by

Method	Utility Loss			Privacy Risk				Empirical Loss
	CC	PR	THN	RD5	RD20	RN5	RN20	
BL, k=5 Am	0.0005	0.0001	0.0001	0	0.26	0.9706	0.9710	2.2023
BL, k=10 Am	0.0014	0.0003	0.00013	0	0.056	0.945	0.946	1.9489
n=1 Amazon	0.4604	0.4786	0.0224	0.008	0.008	0.178	0.182	1.3294
n=2 Amazon	0.5877	0.4477	0.0620	0	0	0.1079	0.111	1.3163
n=5 Amazon	0.7689	0.3504	0.1250	0	0	0.0101	0.0106	1.2651
n=10 Amazon	**0.8142**	**0.2022**	**0.0787**	**0**	**0**	**0.0074**	**0.0081**	**1.1106**
BL, k=5, Tw	0.045	0.012	0.004	0	0.037	0.763	0.765	1.626
BL, k=10 Tw	0.108	0.043	0.008	0	0	0.59	0.595	1.344
n=1 Twitter	0.609	0.890	0.040	0.001	0.001	0.084	0.085	1.709
n=2 Twitter	0.173	0.359	0.005	0.22	0.280	0.310	0.317	1.444
n=5 Twitter	0.111	0.257	0.004	0.16	0.187	0.345	0.352	1.256
n=10 Twitter	**0.086**	**0.184**	**0.003**	**0.15**	**0.167**	**0.341**	**0.348**	**1.129**

Table 5.1: Evaluation of proposed anonymization method versus classical structural k-degree methods: best results minimizing the empirical loss are obtained with our methodology for the highest number of parameters (n=10) for Amazon as well as for Twitter.

our approach are comparable to the ones obtained by the baseline, while providing graphs that are anonymized for different possible attacks showing the genericity of our technique.

Dataset	n	Learned $[\theta_1, ..., \theta_n]$
Amazon	1	[0.009]
Amazon	2	[0.007 0.06]
Amazon	5	[0.005 0.052 0.032 0.056 0.076]
Amazon	10	[0.001 0.013 0.029 0.035 0.043 0.035 0.029 0.042 0.034 0.042]
Twitter	1	[0.015]
Twitter	2	[0.004 0.015]
Twitter	5	[0.003 0.016 0.015 0.011 0.021]
Twitter	10	[0.003 0.020 0.020 0.013 0.016 0.014 0.007 0.005 0.010 0.021]

Table 5.2: Learned parameters when varying the number of parameters for the two datasets. Parameters learned corresponding to high degrees are higher than parameters learned for low degrees.

Table 5.2 represents the parameters learned when varying the number of parameters n for the two datasets. Parameters are sorted according to the nodes degrees, parameters on the left correspond to small degrees and parameters on the right correspond to big degrees. As illustrated in table 5.2 vertices with low degree do not need an important noise addition for anonymization. High degree vertices need more noise in order to preserve their privacy. This result is intuitive as by modifying the degree of a node with many connections, all the connected nodes will change their degree sequence. The consequence will be a decrease in the privacy risk related to neighbors degree sequence (RNk used in the learning phase) by performing a limited number of graph modifications.

5.6.3.3 Adaptation

The last set of experiments aims at showing that our method is not specific to some particular privacy risks and utility losses and can automatically adapt itself to different anonymization configurations.
Table 5.3 shows the results obtained when the learning algorithms only consider the $RD-5$ privacy risk instead of the average of all the 4 proposed risks. In that case, one can see that the obtained results are worse than the results obtained with the baseline models which is not surprising since the BL model has been developed for this particular situation. It means that, when knowing which attacks to avoid, the use of specific methods is certainly the best choice.

Anonymization method	Utility Loss			Privacy Risk	Empirical Loss
	CC	PR	THN	RD5	
BL, k=5, Amazon	0.0005	0.0001	0.0001	0	0.0007
BL, k=10, Amazon	0.0013	0.0003	0.00001	0	0.00161
n=1, Amazon	0.4249	0.4623	0.0006	0.005	0.8928
n=2, Amazon	0.5106	0.4052	0.0031	0	0.9189
n=5, Amazon	0.5779	0.2304	0.0037	0	0.812
n=10, Amazon	**0.5407**	**0.1996**	**0.0026**	**0**	**0.7429**
BL, k=5, Twitter	0.045	0.012	0.004	0	0.186
BL, k=10, Twitter	0.161	0.045	0.095	0	0.301
n=1, Twitter	0.035	0.065	0.008	0.673	0.781
n=2, Twitter	0.043	0.070	0.009	0.519	0.641
n=5, Twitter	0.059	0.096	0.014	0.3	0.469
n=10, Twitter	0.047	0.092	0.016	0.261	**0.416**

Table 5.3: Results obtained when considering only the $RD-5$ privacy risk and the three different utility losses during learning.

We consider the case when different possible attacks can be encountered, but only one characteristic has to be preserved – Table 5.4. In this case, our model clearly outperforms the baseline methods when considering the balance between utility loss and privacy risk (last column).

5.6.3.4 Learning Convergence

At last, we study how many evaluations we have to make during the training phase. We have performed different trainings with different values of N and P.

Anonymization method	Utility Loss	Privacy Risk				Empirical Loss
	THN	RD20	RN5	RN20	RD5	
BL, $k = 5$, Amazon	0.0001	0.26	0.9706	0.971	0	2.2017
BL, $k = 10$, Amazon	0.00013	0.056	0.945	0.946	0	1.947
$n = 1$, Amazon	0.0084	0	0	0	0	0.0084
$n = 2$, Amazon	0.0075	0	0.00002	0.00002	0	0.0075
$n = 5$, Amazon	0.01138	0	0	0	0	0.0114
$n = 10$, **Amazon**	**0.0083**	**0**	**0**	**0**	**0**	**0.0083**
BL, $k = 5$, Twitter	0.004	0.037	0.763	0.765	0	1.569
BL, $k = 10$, Twitter	0.008	0	0.59	0.595	0	1.193
$n = 1$, Twitter	0.39	0.004	0.114	0.117	0.005	0.63
$n = 2$, Twitter	0.396	0.006	0.113	0.115	0.007	0.637
$n = 5$, **Twitter**	**0.387**	**0.002**	**0.115**	**0.118**	**0.002**	**0.624**
$n = 10$, Twitter	0.398	0.008	0.11	0.112	0.006	0.634

Table 5.4: Results obtained when considering the four different privacy risks and only one utility loss during learning.

Figure 5.9 illustrates the performance obtained with respect to the number of candidates

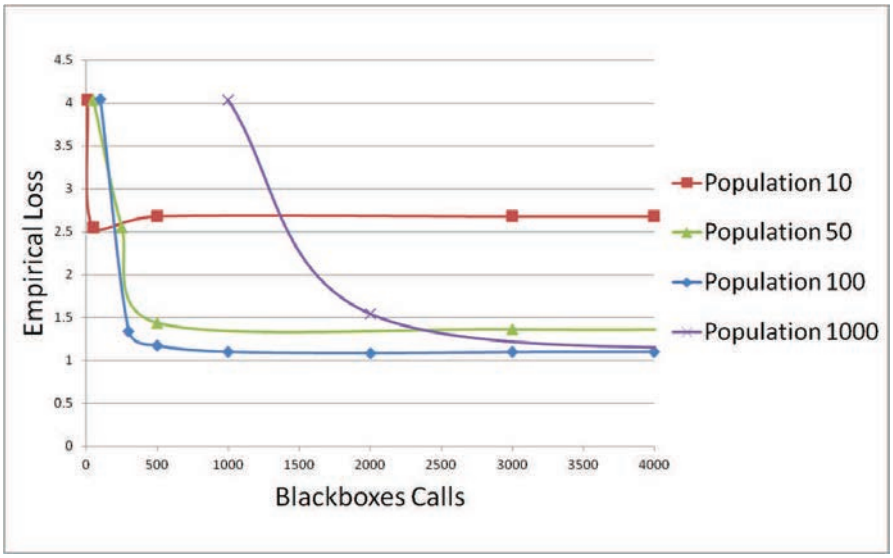


Figure 5.9: EDA efficiency versus number of calls to the blackboxes.

that have been generated on the Amazon dataset for n=5.

The result shows that for $N = 100$ a good anonymization procedure is obtained by computing about 300 candidates which is quite low, allowing our method to be used in real-world situations.

5.6.3.5 Algorithms Complexity

Novel algorithms proposed in this chapter are related to the anonymization procedure described in Algorithm 3 and to the method used to evaluate the privacy risk when releasing an anonymized version of the original data described in Algorithm 4 .

The anonymization algorithm adds randomly new edges between all non connected vertices, a total of potential $S^2 - U$ new edges. The add operation is performed according to the θ_i parameter associated to each node. This θ_i parameter corresponds to a probability and could take values between 0 and 1. However, values superior to 0.5 for this parameter correspond to an important modification of the graph and to a loss of its utility. In our experiments the initial model has been initialized with maximum values of 0.1 for θ and it converges rapidly to values less than 0.1 However, the theoretical complexity of this algorithm corresponds to $O(S^2)$ but in practice this complexity is never achieved.
The Algorithm 4 corresponds to the evaluation of the privacy risk for a given anonymized version of the original data. In a first step an iteration through all the vertices is performed corresponding to a complexity $O(S)$. In the second step of the algorithm for each original vertex if privacy is infringed, the privacy risk is incremented. This algorithm corresponds to a $O(S^2)$ complexity.

5.7 Conclusion

In this chapter we propose the application of the anonymization method based on machine learning, on simple graphs. Our methodology aims to find the best compromise between privacy protection and graph data utility loss in a given context. The advantage when compared to existing methods is that our proposed methodology is able to find the best compromise when dealing with a large panel of privacy attacks or analysis to be made on data. Moreover, the knowledge on how this analysis is performed is not needed in order to find an anonymization strategy. The tool implementing the methodology has been developed in Java and tested on real data with successful results. The methodology has been tested on simple non-oriented graphs. Results from this chapter have been presented and published at the 28th IEEE International Conference on Advanced Information Networking and Applications (Maag et al. [2014]). The next step consists in exploring the behavior of the method when dealing with complex structures issued from communication logs having a temporal component and multiple oriented edges.

Chapter 6

Adaptive Temporal Graphs Anonymization for Data Publishing

This chapter applies the methodology described in Chapter 4 to the temporal graphs anonymization problem. Some additional notations and definitions specific to temporal graphs are first introduced. For the instantiation of the methodology on temporal graphs, new privacy risks are described as well as a set of utilities specific to call logs. Experiments performed on real datasets coming from Enron or Twitter outperform baseline and show that method adapts well to a new environment. Results from this chapter have been submitted for publication.

Contents

6.1 Introduction

When starting the research work on data anonymization and particularly on Call Detail Records anonymization, an increased risk of re-identification has been identified in this type of data because of its temporal component. Chapter 3 aims at pointing out these risks. Starting from that identification, we proposed a generic methodology for anonymization based on machine learning in Chapter 4. Existing research work being available especially related to simple graphs anonymization, we first apply this methodology on simple graphs (Chapter 5). Comparison with existing methods let us believe that the proposed methodology could present a good behavior also on the original problem, which is the multiple oriented timestamped graphs anonymization. Similar to simple graphs data, call logs data anonymization should be performed to prevent users re-identification and at the same time to keep as much as possible of the utility of data. Call Detail Records (CDR) necessity for anonymization is described in the introduction of Chapter 3. CDRs are used in a multitude of contexts. In Lin and Wan [2009], CDRs representing mobile communications are used for clustering users according to their behavior. Operator's marketing strategies are then adapted to the customer type. Call logs contain a huge amount of information about the user's behavior and can be successfully used for marketing purposes.

D4D (Data for Development) (Orange) is a contest launched by Orange in 2012 which released four anonymous datasets, with the objective of bringing a benefit for the concerned population. The dataset was delivered to the scientific community in an anonymized format and it contained calls data collected by the Orange Ivory Coast subsidiary. Research projects participating at the contest are diverse and show the multitude of usages of the delivered data and the positive impact of data sharing. Van den Elzen et al. [2013] was the winner of the best visualization price accorded by the contest. The visual analytic tool developed is capable to explore complex patterns (like shown in Figure 6.1). The goal is to be able to determine by analyzing datasets, weather-driven events (like for example heavy rainfall in important cocoa areas) or social and political events. Event detection by analyzing static datasets is the first step before implementing tools able to predict those events. However, access to call logs in real time pose a privacy

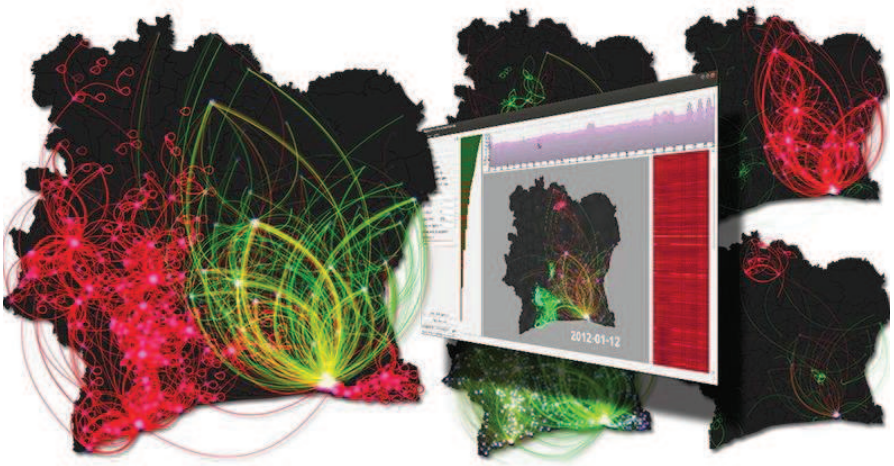


Figure 6.1: Exploration and analysis of massive mobile phone dataset.

problem. Anonymization techniques applied on CDRs as soon as they are created could be a solution to this issue.

Another application of the dataset analysis has been described in Berlingerio et al. [2013]. The "AllAboard" platform describes a system able to extract the needs of the population in terms of mobility. Mobile's phones location data is used to track users and to determine the travelers flows in the city. The system proposes a model improving the existing one and reducing waiting times.

The best scientific prize winner of the D4D contest was the project described in van den Elzen et al. [2013]. By analyzing the call detail records between individuals, a mapping of the different existing communities has been obtained. Figure 6.2 shows the communities corresponding to high volume of calls between individuals. A similar study has been performed in Belgium by Guigoures and Boullé [2011] and clusters have been obtained from call detail records corresponding to linguistic communities. In Guigourès et al. [2013] it is showed how mobile phone call logs can be used in order to study space-time correlations between calls. The temporal evolution of mobile users behavior can then be inferred corresponding to each geographic part of the country.

D4D initiative shows how statistics made on anonymous data can be used to improve the well-being of the population. Projects having used this data had as goal for example to improve public transportation, to propose new models for disease spreading based on mobility data or to calculate a "social distance" between users and detect communities. Even if the release of call logs datasets has undeniable benefits, the privacy infringement risk is a reality for persons present in the dataset. In Sharad and Danezis [2013] it has been shown that the released dataset for the D4D challenge, even if anonymized, can be de-anonymized by using neighborhood knowledge of the users present in the dataset.

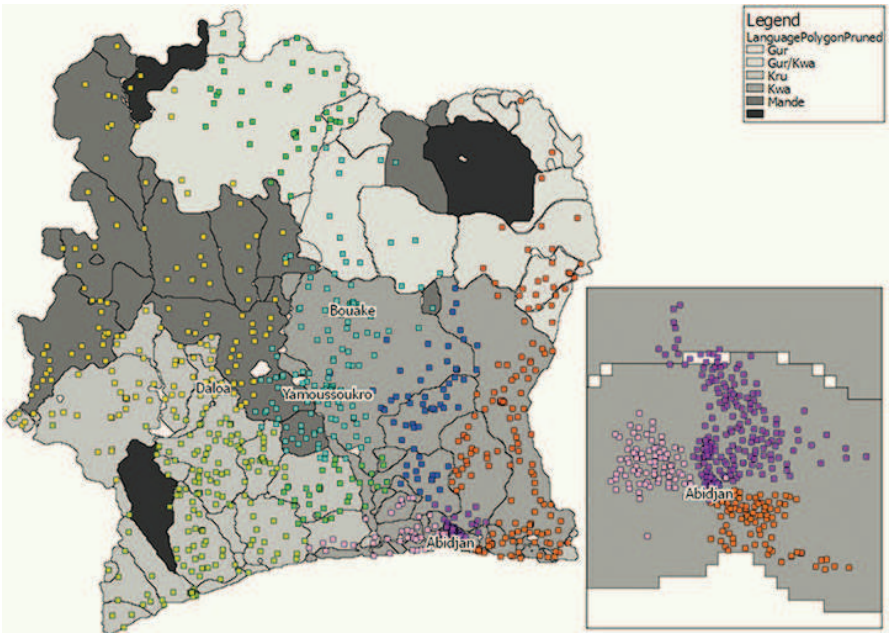


Figure 6.2: Analyze social divisions using cell phone data.

Anonymization techniques have been developed for different kinds of data representations. Techniques to anonymize data for which each record represents a separate entity (as in a relational database) have been developed and are currently used, partially with a successful result. However, datasets have moved from traditional models to complex models. The anonymization techniques for tabular data cannot be successfully applied to those complex models. Most of the existing work in anonymization dealing with call log data, tackled the anonymization for data represented as a simple graph. The proposed techniques are usually targeting a specific type of attack (e.g., Liu and Terzi [2008]) or a certain type of utility to preserve (e.g., Ying and Wu [2008]). When releasing complex timestamped datasets, in practice, noise addition is used in most of the cases to perturb structures used for re-identification. Existing techniques are **not easy to adapt** to a multitude of possible attacks or measures to be preserved on data. In addition, most of the techniques are based on an **analysis of the entire set of data** to be anonymized. For call detail records, in some specific cases new data is continuously created and it has to be properly anonymized. Moreover, in the case of big datasets, the dataset as a whole is too important to be entirely analyzed. The contributions in this chapter are the following:

- We apply the generic approach for call logs anonymization based on machine learning techniques described in Chapter 4 to call logs.
- The method does not need access to the entire set of data for determining the anonymization function. It then fits well to dynamic data and newly created data

is correctly anonymized.

- We define new de-anonymization techniques based on the temporal component of the graph to be anonymized for privacy evaluation.
- We compare our method with baseline methods and show its effectiveness on real datasets.

6.2 Machine Learning for Call Detail Records Anonymization

6.2.1 Notations and Definitions

We address the problem of call log and interaction data anonymization. Call log data are modeled as graphs with multiple, oriented and timestamped edges.
Let $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{T})$ be the graph containing the timestamped relational data. Similar to simple graphs, the vertices of $\mathcal{G}$ are defined as $\mathcal{V} = (v_1, ..., v_S)$ with each vertex v_i representing a communication entity. Timestamped communication events having occurred between two vertices form a set $\mathcal{C} = (c_1, ..., c_U)$ with $c_k \in \mathcal{C}$, $c_k = (o(c_k), d(c_k), t(c_k))$ where $o(c_k) \in \mathcal{V}$ is the origin of the call, $d(c_k) \in \mathcal{V}$ is the destination of the call and $t(c_k) \in \mathcal{T}$ is the timestamp of the call. Graph $\mathcal{G}$ has multiple oriented edges with a label $t(c_k)$ for each edge corresponding to the timestamp of the event. Let $\mathcal{G}' = (\mathcal{V}', \mathcal{C}', \mathcal{T}')$ be the anonymized graph. The anonymization function receives in input $\mathcal{G}$ and outputs the anonymized graph $\mathcal{G}'$. The anonymization function f_θ, the utility loss Δ and the privacy risk R used to evaluate the compromise between privacy protection and utility loss have been introduced in section 4.3.

6.2.2 Learning Problem

Our methodology described in Chapter 4 consists in learning a parameterized function on a set of subgraphs extracted from the dataset and then applying the learned function on the rest of the dataset. In order to form the dataset for training and for testing, we define the following projection operator:

$$O : \mathcal{G} \times \mathcal{T} \times \mathcal{T} \rightarrow (\mathcal{V}, \mathcal{C}, \mathcal{T})$$

used to extract the subgraphs corresponding to time intervals. Figure 6.3 shows the subgraphs extraction process by using the projection operator from the initial dataset.

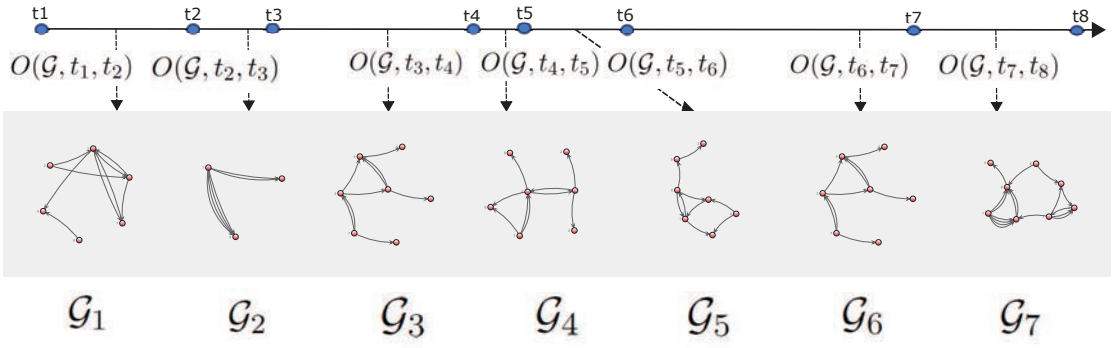


Figure 6.3: Subgraphs extraction from the original dataset: projection operator O is operated on each time interval to obtain subgraphs $\mathcal{G}_1, ...\mathcal{G}_7$.

The optimization problem to be solved is the one described in 4.4, the goal being to find best parameters θ^*, for a given parameterized anonymization method f, such as:

$$\theta^* = \arg\min_{\theta} \left(\frac{1}{T} \sum_{j=1}^{T} \left(\frac{1}{m} \sum_{\mathcal{G}_i} \Delta(\mathcal{G}_i, f_\theta(\mathcal{G}_i)) + \lambda \frac{1}{|A|} \sum_{a_k} R(\mathcal{G}_i, f_\theta(\mathcal{G}_i), a_k) \right) \right) \tag{6.1}$$

Optimization algorithms described in section 4.6 are then used to learn on the subgraphs forming the learning set, the best parameterized function to be used in the given context on the rest of the dataset.

6.3 Anonymization Method

We have used an anonymization method based on randomized techniques to perform data perturbation by adding noise into the original data. The operations performed on a given graph are: add a new edge, delete an edge, replace calling vertex by a random vertex and replace called vertex by a random vertex.
The anonymization algorithm is parameterized with parameter vector of size n:

$$\boldsymbol{\theta} = (\boldsymbol{\theta}^{\boldsymbol{Add}}, \boldsymbol{\theta}^{\boldsymbol{Delete}}, \boldsymbol{\theta}^{\boldsymbol{Replace\_Left}}, \boldsymbol{\theta}^{\boldsymbol{Replace\_Right}})$$

where each part of the vector $\boldsymbol{\theta}$ corresponds to one of the operations used for anonymization as in the following:

$$\boldsymbol{\theta}^{\boldsymbol{Add}} = (\theta_1^{add}, ..., \theta_p^{add})$$

$$\boldsymbol{\theta}^{\boldsymbol{Delete}} = (\theta_1^{delete}, ..., \theta_p^{delete})$$

$$\boldsymbol{\theta^{Replace\_Left}} = (\theta_1^{replaceLeft}, ..., \theta_p^{replaceLeft})$$

$$\boldsymbol{\theta^{Replace\_Right}} = (\theta_1^{replaceRight}, ..., \theta_p^{replaceRight})$$

We make the hypothesis that the noise to add on a pair of vertices is dependent of the degree corresponding to each vertex in the pair. Vertices are split into p groups according to their degrees ($n = 4 \times p$). To each operation, p parameters are associated, one for each group of degrees.
For each vertex v, given its degree, a parameter $\theta_i^{operation}$ will be associated for each type of possible operations (add, delete, replace left or replace right). Let D be the maximum degree of the original graph $\mathcal{G}$. The parameter $\theta_i^{operation}$ will be associated to all nodes of degree d such as:

$$\left|\frac{D}{p}\right|(i-1) < d \leq \left|\frac{D}{p}\right| i \tag{6.2}$$

except for the parameter $\theta_p^{operation}$ for which corresponding nodes have $d \leq D$.
We will denote the parameter $\theta_i^{operation}$ associated with the vertex v according to the degree of v with $\theta_{gr(v)}^{operation}$ where operation $\in \{add, delete, replaceLeft, replaceRight\}$ and gr(v) is the group of vertices corresponding to parameter $\theta_i^{operation}$.

The anonymization algorithm is described in Algorithm 5. First, $\mathcal{G}'$ is initialized with the values of $\mathcal{G}$ (line 1). We suppose that the graph $\mathcal{G}$ has already been naively anonymized.

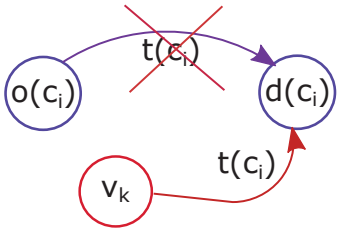


Figure 6.4: Replace left operation: origin vertex of the edge is replaced by a random vertex.

For the replace left operation, for each communication $c = (o(c), d(c), t(c))$, where $o(c)$ is associated to $\theta_{gr(o(c))}^{replaceLeft}$ and $d(c)$ is associated to $\theta_{gr(d(c))}^{replaceLeft}$, a boolean is generated with the probability $\theta_{gr(o(c))}^{replaceLeft} \times \theta_{gr(d(c))}^{replaceLeft}$ (line 3). If the generated boolean is true, then a replace left operation is performed in the communication $c = (o(c), d(c), t(c))$ (lines 4 and 5). The procedure for the replace left operation is illustrated in Figure 6.4. The origin vertex is replaced with a random vertex in the graph. The procedure is

Algorithm 5: Anonymization Method

Data: $\mathcal{G}$; θ_1, ...,θ_n
Result: $\mathcal{G}'$

1 $\mathcal{G}' \leftarrow \mathcal{G}$;
2 **foreach** $c_i = (o(c_i), d(c_i), t(c_i))$ *in* $\mathcal{G}'$ **do**
3 **if** *generatedBoolean*$\left(\theta^{replaceLeft}_{gr(o(c_i))} \times \theta^{replaceLeft}_{gr(d(c_i))}\right)$ **then**
4 randomly select vertex v_k from $\mathcal{G}'$;
5 $o(c_i) = v_k$;
6 **end**
7 **if** *generatedBoolean*$\left(\theta^{replaceRight}_{gr(o(c_i))} \times \theta^{replaceRight}_{gr(d(c_i))}\right)$ **then**
8 randomly select vertex v_k from $\mathcal{G}'$;
9 $d(c_i) = v_k$;
10 **end**
11 **if** *generatedBoolean*$\left(\theta^{delete}_{gr(o(c_i))} \times \theta^{delete}_{gr(d(c_i))}\right)$ **then**
12 delete c_i from $\mathcal{G}'$;
13 **end**
14 **end**
15 **foreach** *vertex* v_i *in* $\mathcal{G}'$ **do**
16 **foreach** *vertex* v_j *in* $\mathcal{G}'$ **do**
17 **if** *generatedBoolean*$\left(\theta^{add}_{gr(v_i)} \times \theta^{add}_{gr(v_j)}\right)$ **then**
18 generate random T_k between T_{min}, T_{max};
19 insert into $\mathcal{G}'$ c(v_i, v_j, T_k);
20 **end**
21 **end**
22 **end**

the same for the replace right operation. For each communication $c = (o(c), d(c), t(c))$, where $o(c)$ is associated to $\theta^{replaceRight}_{gr(o(c))}$ and $d(c)$ is associated to $\theta^{replaceRight}_{gr(d(c))}$, a boolean is generated with the probability $\theta^{replaceRight}_{gr(o(c))} \times \theta^{replaceRight}_{gr(d(c))}$ (line 7). If the generated boolean is true, then a replace right operation is performed in the communication $c = (o(c), d(c), t(c))$ (lines 8 and 9). The procedure for the replace right operation is illustrated in Figure 6.5. In this case, the destination vertex $d(c)$ is replaced with a random one.

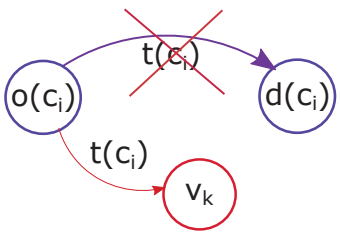


Figure 6.5: Replace right operation: destination vertex of the edge is replaced by a random vertex.

For the delete operation, for each communication $c = (o(c), d(c), t(c))$, where $o(c)$ is associated to $\theta^{delete}_{gr(o(c))}$ and $d(c)$ is associated to $\theta^{delete}_{gr(d(c))}$, a boolean is generated with the probability $\theta^{delete}_{gr(o(c))} \times \theta^{delete}_{gr(d(c))}$ (line 11). If the generated boolean is true, a delete operation is performed as shown in Figure 6.6. The communication c wil not exist anymore in the anonymized data.

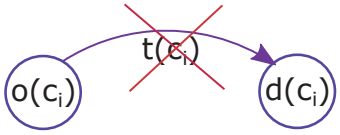


Figure 6.6: Delete operation: edge is deleted.

For the add operation, given each pair (v_i, v_j) of vertices ("for" loop, lines 15 and 16) , where v_i is associated to $\theta^{add}_{gr(v_i)}$ and v_j is associated to $\theta^{add}_{gr(v_j)}$, a boolean is generated with the probability $\theta^{add}_{gr(v_i)} \times \theta^{add}_{gr(v_j)}$ (line 17). T_{min} is the minimum timestamp of $\mathcal{G}$ and T_{max} is the maximum timestamp of $\mathcal{G}$. If the generated boolean is true, a new edge between v_i and v_j (lines 18 and 19) is added (Figure 6.7) with a timestamp T_k randomly generated between T_{min} and T_{max}.

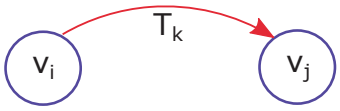


Figure 6.7: Add operation: new edge is added between vertices.

Note that when considering $n = 1$ with only add operations, the probability of adding an edge does not depend on the considered vertices, and thus the obtained method is a random anonymization method with a uniform parameter similar to the one used in Ying and Wu [2008].

6.4 Privacy Risks in Call Logs

Once the parameterized anonymization function defined, the blackbox used to evaluate the anonymized data's privacy has to be defined.
Note that the privacy risks defined hereafter have as main goal the methodology evaluation in the context of timestamped multiple graphs anonymization. Any other privacy risk evaluation method could be used, our approach being independent of the content of the privacy blackbox.
We next define two types of attacks: one proper to call logs that we first describe in this

paper and a second one already known in the literature (Sharad and Danezis [2013]). The first type of attack is strictly related to call logs and shows the sensitivity of this type of data when exploiting their temporal component. The second type of attack we have used for evaluation has been recently described in the literature in the case of the D4D call logs de-anonymization. Both of the algorithms described are based on the re-identification in the anonymized data of one or more "fingerprints" characterizing users, known by the adversary and not modified by the anonymization process.

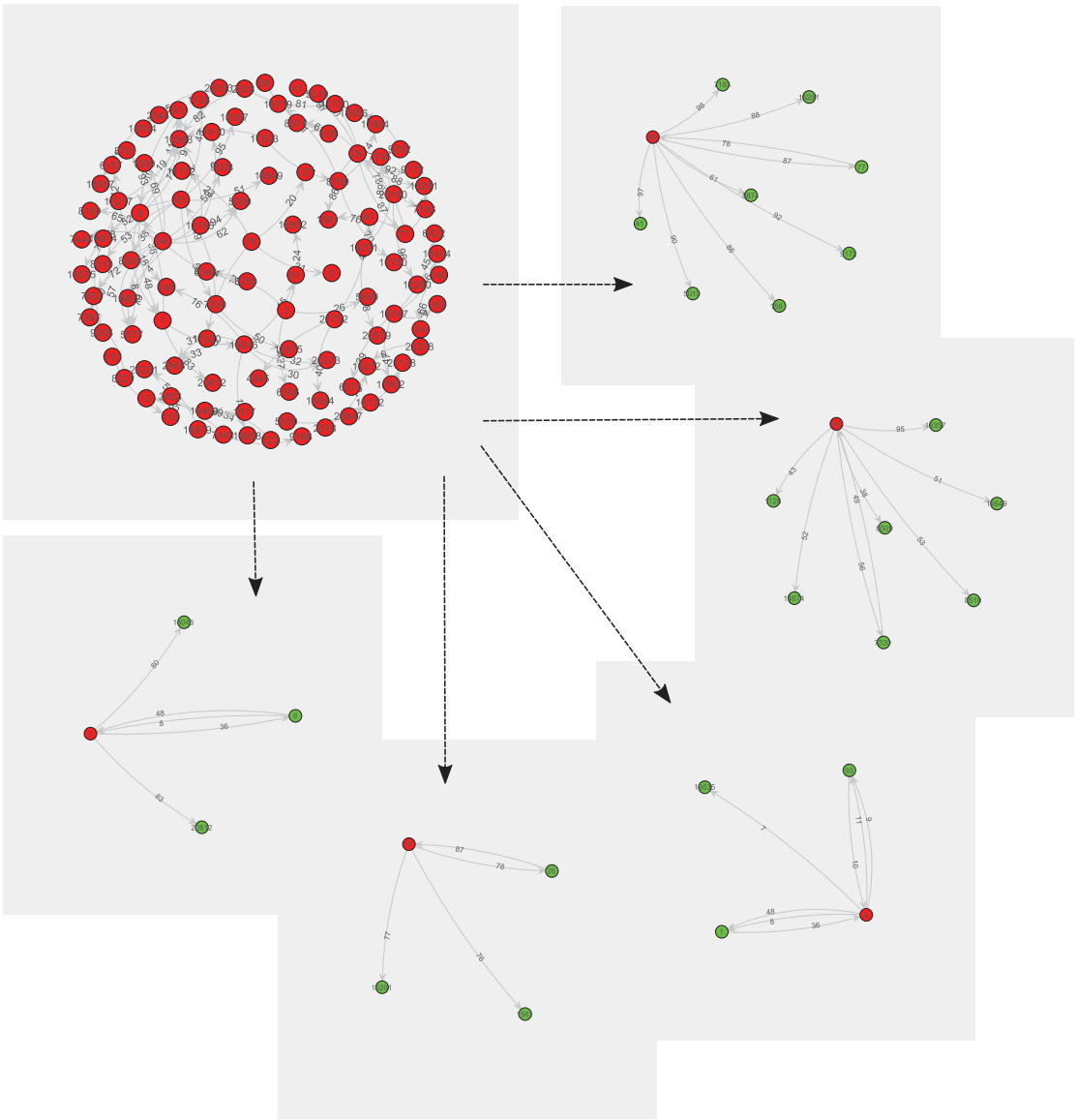


Figure 6.8: Fingerprints identified as potential attacks in a graph.

A "fingerprint" corresponds to a sequence extracted from the dataset capable of re-identifying a small amount of nodes in the released dataset. Figure 6.8 shows an example of extracted attacks from a given dataset. These fingerprints generation (and

related attacks identification) are based on different de-anonymization techniques described further in sections 6.4.1 and 6.4.2. To evaluate the global privacy risk we have

Algorithm 6: Privacy Score Evaluation Based on k-Anonymity

Input: FingerprintList for vertices in $\mathcal{G}$
Input: FingerprintList for vertices in $\mathcal{G}'$
Input: k
Input: Attacks - potential attacks in $\mathcal{G}$
Output: privacyRiskLevel for $\mathcal{G}$ when releasing $\mathcal{G}'$

```
1  init privacyRiskLevel=0;
2  init Hashtable privacyUsers;
3  build UsersFingerprint(G), UsersFingerprint(G') ;
4  foreach fingerprint s_i in UsersFingerprint(G') do
5      if Count(UsersFingerprint(G').getValue) < k then
6          foreach v_i in (UsersFingerprint(G').getValue) do
7              privacyRisk=0;
8              if privacyUsers.contains(v_i) then
9                  privacyRisk =
                   privacyUsers.get(v_i) + 1/(UsersFingerprint(G').getValue);
10             end
11             else
12                 privacyRisk = 1/(UsersFingerprint(G').getValue);
13             end
14             privacyUsers.put(v_i, privacyRisk) ;
15         end
16     end
17 end
18 foreach v_j in privacyUsers do
19     if Attacks.containsValue(v_j) then
20         at = Attacks.getKey(v_j);
21         if (UsersFingerprint(G).getValue(at).contains(v_j)) then
22             privacyRiskLevel = privacyRiskLevel + privacyUsers.get(v_j);
23         end
24     end
25 end
```

used the k-anonymity concept described in Sweeney [2002]. According to the privacy risks described below, our call graph is considered as being k-anonymous if the generated fingerprints for a vertex cannot be distinguished from at least $k-1$ other vertices in the anonymized graph.

Algorithm 6 describes how the global privacy evaluation is performed. We first identify in the original graph fingerprints associated with potential attacks because of their low occurrence frequency. "Attacks" is a hashtable used as input for the privacy evaluation algorithm. It contains as key the fingerprints identified as possible attacks and as values the corresponding users characterized by the fingerprint (Figure 6.9). The algorithm

takes as input the "FingerprintList" which is a hashtable containing as keys the users and as values the fingerprints characterizing the users in a given graph.

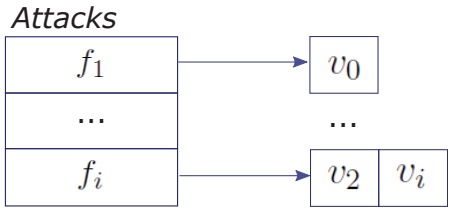


Figure 6.9: Hashtable containing the identified attacks (fingerprints) in $\mathcal{G}$ and the corresponding vertices.

Fingerprints characterizing the users are generated following algorithms further described in sections 6.4.1 or 6.4.2. Based on this input data, the algorithm builds "UsersFingerprint($\mathcal{G}$)" and "UsersFingerprint($\mathcal{G}'$)" (line 3), hashtables containing as key all the fingerprints generated and as values the list of the vertices characterized by the fingerprint.

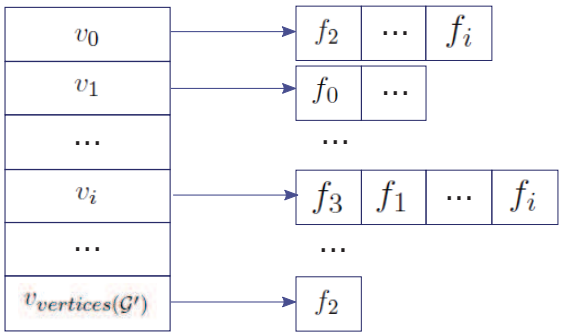


Figure 6.10: Hashtable containing the fingerprints characterizing each vertex in $\mathcal{G}'$.

Figure 6.11 shows an example of "UsersFingerprint($\mathcal{G}'$)" for $k = 4$. In red are represented the sequences for which the number of corresponding users is less than 4.

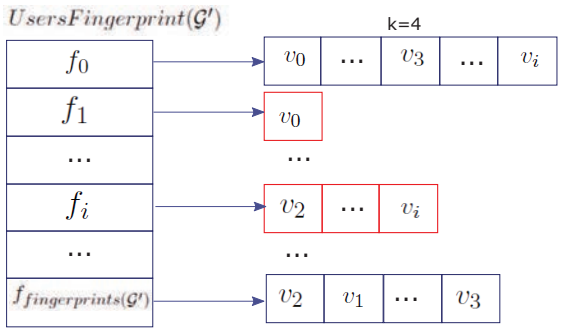


Figure 6.11: Hashtable containing as key the fingerprints and as value the list of the vertices characterized by the corresponding fingerprint in $\mathcal{G}'$.

Hashtable "privacyUsers" (Figure 6.12) is used to store for each sensitive user, its corresponding privacy level.

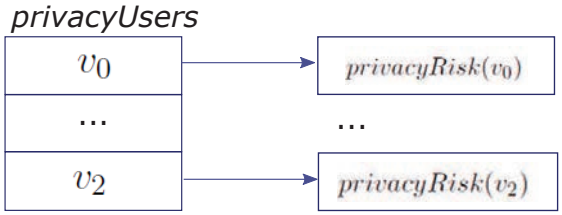


Figure 6.12: Hashtable storing for each vertex the corresponding privacy level. "privacyRisk" value is incremented at each time a privacy infringement is detected for the corresponding vertex.

It contains as key all the users in $\mathcal{G}'$ with a fingerprint present in the dataset characterizing less than k users. For each user the level of the privacy infringement is stored as value in this hashtable.
The algorithm goes over all the values of "UsersFingerprint($\mathcal{G}'$)" and for each fingerprint in the anonymized graph, if the number of corresponding users is less than k (line 5) we consider that a k-anonymity based privacy infringement occurred with a level equal to 1/(users with the same fingerprint). The privacy risk is then incremented for the concerned users (lines 9 and 12) and stored in the hashtable "privacyUsers".
We took into account for the global risk evaluation the privacy infringement corresponding only to the potential attacks. For each sensitive user in the anonymized data (loop "for", line 18), if the user was identified as a potential attack in the original data (line 19) with the same fingerprint (line 21), the global privacyRiskLevel of the dataset is incremented with the corresponding value from the hashtable "privacyUsers".
The final value for the privacy risk of an anonymization method, is obtained after several iterations by calculating the mean on the privacy risk obtained at each iteration.
When the anonymization method is limited to the identifiers replacement, the worse case for privacy is met. If a lot of noise is added, privacy risk decreases and becomes 0 at the price of losing the utility of data.
We next describe the fingerprints generation corresponding to each de-anonymization method.

6.4.1 Privacy Attack by Communication Sequence Generation (CSG)

The problem of call logs anonymization is relatively new, the chronologic order of the communications not being much exploited by the de-anonymization algorithms. We define a new type of attack based on the order of the communications occurred between a user and its callers or callees.

Since each edge is a communication/interaction occurred at a certain time, each subgraph of the communication log can be represented as a call sequence. Therefore, data can be decomposed in a multitude of call sequences. For each user we are able to generate her call sequences corresponding to a certain period of time. To generate all these call sequences we use the projection operator O as defined before.

When anonymizing communication data, a first approach used is to simply replace real identifiers by generated identifiers. However, the order in which a user called his correspondents remains the same. This call order acts like a fingerprint allowing user's de-identification in the anonymized data. We suppose the adversary has access to this type of information or that he embedded this information in the original dataset.

For each user in the dataset, we generate a call sequence as described in the Algorithm 7 of maximum length L. The call sequences of each user act like a fingerprint and are stored in the hashtable "FingerprintList"used as input for the Algorithm 6 represented in Figure 6.10.

Algorithm 7: Fingerprints Generation Based on Call Sequences

Input: $\mathcal{G}$
Input: Length L of the sequences
Output: FingerprintList

```
1  init FingerprintList;
2  foreach vertex v_i in V do
3      Select calls c_1 .. c_CallsVi connected with v_i;
4      init VerticesList;
5      insert (v_i, "0") in VerticesList ;
6      init String Sequence;
7      init index=1;
8      foreach call c_j ordered by timestamp ASC do
9          if Sequence.length < L then
10             get corresponding call vertex v_k;
11             if v_k is not in VerticesList then
12                 insert (v_k, index) in VerticesList ;
13                 index++;
14             end
15             correspNodeID=VerticesList.getValue(v_k);
16             if c_j is outgoing then
17                 Sequence = Sequence + "-0–" + correspNodeID + "-";
18             else
19                 Sequence = Sequence + "-" + correspNodeID + "–0-";
20             end
21         end
22     end
23     insert (v_i, Sequence) in FingerprintList;
24 end
```

To construct a call sequence of size less than L (line 9), we start by assigning to the

analyzed vertex the value 0 (line 5 of the algorithm). We store the identifier assigned to each new user in the hashtable "VerticesList". Then, we allocate to each new vertex an identifier corresponding to the order it appears in the call sequence. For this purpose, we initialize an index at 1 (line 7).

For each new communication of the analyzed vertex (loop "for", line 8), if we encounter the called or caller vertex for the first time (line 11), we assign it an identifier equal to the index value (line 12) and we increment the index (line 13).

The generated sequence ("Sequence") will consist in a concatenation of strings corresponding to the communications of the analyzed vertex in the order the communications have occurred.

Each string representing a communication contains the identifiers of the called person and of the caller involved in it in the format "-callerID-calledID".

If the analyzed communication is an outgoing call, then the generated fingerprint is concatenated with the origin "-0–" and the value of the corresponding node (line 17).

If the analyzed communication is an incoming call, then the generated fingerprint is concatenated with the origin node identifier and with "–0-" (line 19).

Figure 6.13 illustrates the sequence generation by a user Bob having called Alice twice, then having received a call from Jane and having recalled Alice.

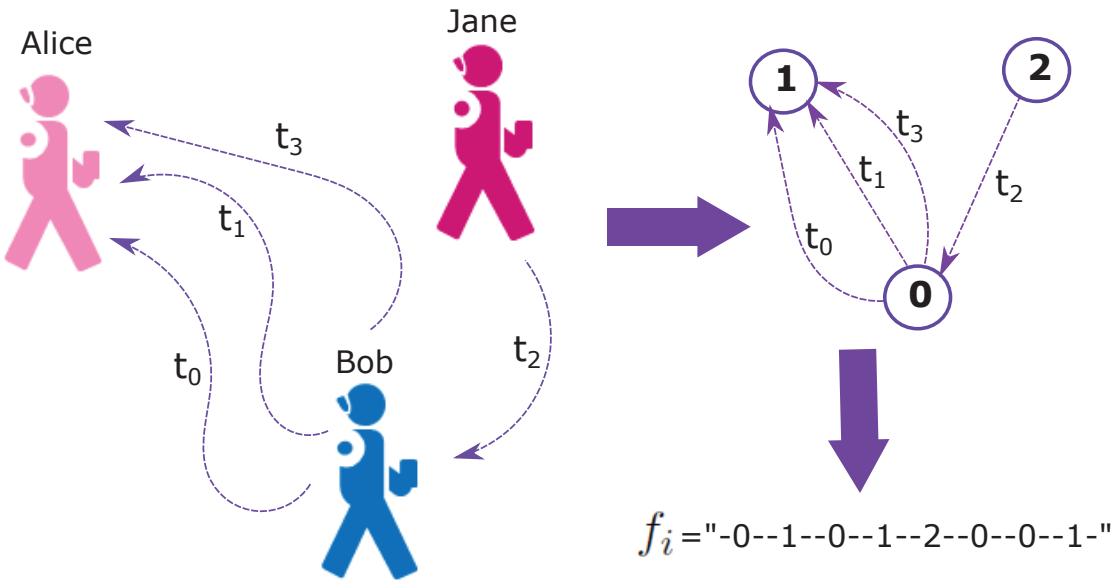


Figure 6.13: Call sequence/fingerprint generation for user Bob.

In this case, the generated fingerprint will be: f_i ="-0–1–0–1–2–0–0–1-".

An important number of call sequences existing in the original data appear only once. This could be an important indicator of data's vulnerability. It means that an adversary having the knowledge of a call sequence, when retrieving the anonymized data will

potentially re-identify the sequence in the anonymized data by generating the call sequences as described before.

To determine the risk for de-identification, we perform a selection among the generated sequences, according to their frequency in the dataset. We consider isolated sequences as potential external attacks.

The privacy evaluation is performed according to the capacity to de-identify the selected sequences in the anonymized dataset.

6.4.2 Privacy attack by Neighborhood Degree Distribution (NDD)

Algorithm 8: Fingerprint Generation Based on Neighbor's Degrees Distribution

Input: $\mathcal{G}$
Output: FingerprintList

```
1  init FingerprintList;
2  SequencesList.Entries=0;
3  foreach call c_j = (v_i, v_j, t_k) in G do
4      if First occurence of interaction (v_i, v_j) then
5          get Degree(v_i);
6          get Degree(v_j);
7          if FingerprintList.Contains(v_i) then
8              FingerprintListOfVertices = FingerprintList.Get(v_i);
9          end
10         else
11             init FingerprintListOfVertices;
12         end
13         FingerprintListOfVertices.put(Degree(v_j));
14         FingerprintList.put(v_i, FingerprintListOfVertices);
15         if FingerprintList.Contains(v_j) then
16             FingerprintListOfVertices = FingerprintList.Get(v_j);
17         end
18         else
19             init FingerprintListOfVertices;
20         end
21         FingerprintListOfVertices.put(Degree(v_i);
22         FingerprintList.put(v_j, FingerprintListOfVertices);
23     end
24 end
25 foreach user v_k in FingerprintList do
26     FingerprintListOfVertices = FingerprintList.get(v_k);
27     Sort in ASC (FingerprintListOfVertices);
       FingerprintList.put(v_k,FingerprintListOfVertices);
28 end
```

This algorithm has already been described in Sharad and Danezis [2013] and it has been used to study the de-anonymization of the D4D (Data for Development) dataset.

This de-anonymization technique uses the 1-hop nodes neighborhood degree distribution to generate a fingerprint for each node allowing de-identification. A fingerprint is generated for each vertex based on the degree distribution of the 1-hop neighbor nodes. As for the previous method, we suppose the adversary has access to this information from external sources or that he embedded this information in the original data.

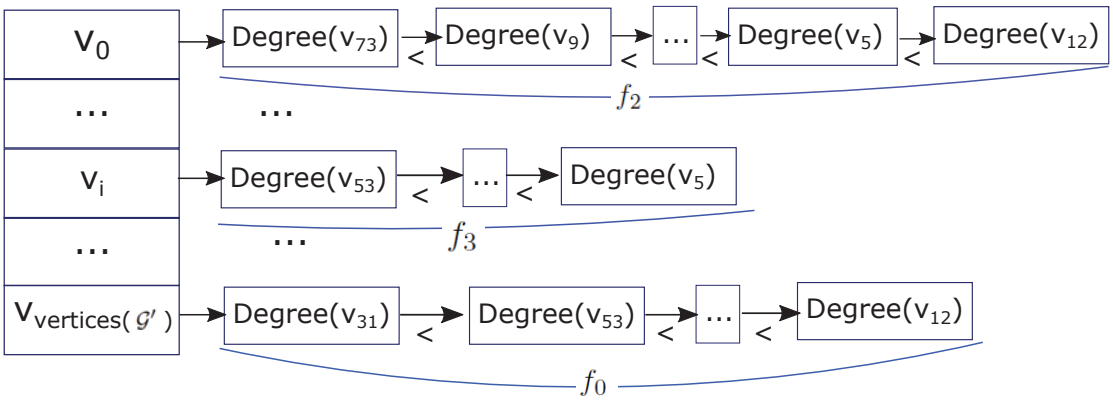


Figure 6.14: Fingerprint list using NDD.

Algorithm 8 describes how the fingerprints are generated for each vertex in the graph. For each call in the dataset (line 3), if the interaction between vertices occurred for the first time (line 4), the degree of the vertices involved in the communication are extracted (lines 5 and 6). If the origin vertex is already in the hashtable "FingerprintList" (line 7), the list of its neighbors degrees is retrieved updated (line 8). Otherwise, the list of its corresponding neighbors is created (line 11). The list of the corresponding neighbors is updated with the new neighbor discovered (line 13). The same operations are performed for the destination vertex. The neighbor list of the destination vertex is updated line 21. At the end of the algorithm, for each node (line 25), the list of its neighbors is generated and sorted in ascending order of the degrees as shown in Figure 6.14, acting as a fingerprint. Fingerprints obtained for all vertices in the graph $\mathcal{G}$ and $\mathcal{G}'$ will be used as an input to compute the global level of privacy in Algorithm 6.

6.5 Utility Loss Evaluation

Call detail records are used for a multitude of analysis from marketing to social studies. An important number of metrics can be employed when evaluating the utility of data. Lin and Wan [2009] describes a method for mobile customers clustering based on call detail records, the goal being the use of the results for marketing campaigns. Several measures are used for this clustering. Call diameter is one of the measures and is defined

as the number of different mobile customers called by or calling to one mobile customer in one month. It then indicates the activeness of a mobile user. The ratio of calling number and called number, the ratio of local calling duration and long distance duration are other measures used in Lin and Wan [2009] to exploit the utility of the call detail records.
Hereafter are some examples of analysis relevant for call detail records:

- Number and distribution of the calls emitted/received by one person or a certain category of people
- Distribution of calls emitted/received by the users in the time interval
- Average distance between two types of nodes
- Detection of frequent calls between two nodes
- Spam detection
- Number of clusters in the communication log
- Sequence degree
- Influence analysis

In our approach the utility is considered as a blackbox and complex data analysis could be implemented inside of it. However, for evaluation purpose, we consider three main families of utilities detailed further.

6.5.1 Changes Performed by the Anonymization Algorithm in the Graph (CHG)

The cost of the anonymization in terms of utility can be defined as being proportional with the number of changes performed in the graph by the anonymization algorithm. In Liu and Terzi [2008] this cost is defined as being equal to the number of edges added by the algorithm in $\mathcal{G}$ to obtain $\mathcal{G}'$.
This type of utility loss evaluation has previously been used on simple graphs. We consider that the loss in the utility of data in a multiple oriented graph is proportional to the operations modifying the graph for anonymizing it.

6.5.2 Query Based Measures: Call Distribution Distance (CDD)

Query based measures consist in measuring the utility as being proportional to the accuracy of the answer to a set of queries.
As described in the prior art, this evaluation method consists in listing a given set of queries and then compare the obtained answers before and after anonymization.
Some examples of query based measures applied on call detail records are given hereafter.

Example 1: Total number of call distribution Randomly pick a certain number of time intervals. Compare the total number of calls per user between $\mathcal{G}$ and $\mathcal{G}'$.

Example 2: Distribution of calls in time Compare the distribution of total incoming/outgoing calls in time between $\mathcal{G}$ and $\mathcal{G}'$.

Example 3: Distribution of total number of callers/callees In most of the given examples, the measure can be represented as a vector. We evaluate the utility loss as being the sum of the difference between each element of the obtained vectors. Randomly pick a certain number of time intervals. Compare the total number of callers/callees in each time interval between $\mathcal{G}$ and $\mathcal{G}'$.

We evaluate the utility loss corresponding to the query related to the calls distribution in data (Example 2). The query for which we need an accurate answer is the distribution of the total number of calls occurred on several periods of time in the dataset. We extract for each time interval the total number of calls having occurred in the time interval. We then evaluate the delta between the responses obtained by using $\mathcal{G}$ and $\mathcal{G}'$.

6.5.3 Graph Topological Properties: Vertices In/Out Degrees (DE)

The "In" and "Out" degrees present in a collaboration network is an important measure used for social networks analysis. Determining the vertex corresponding to an influent person or to an expert is mandatory for certain datasets. An expert could correspond to a person replying for example to an important number of messages. Therefore, the ratio between its "In" degrees and its "Out" degrees should be kept as close as possible to the original one in the anonymized data.
To evaluate the changes in the In/Out degrees between original and anonymized data, we evaluate the changes of the distribution of "in" and "out" degrees in the graph before and after anonymization. In order to do that, we compare vectors representing the

degrees distribution before and after anonymization.
For instance, for the example showed in figure 6.15, we evaluate the loss in the utility of data according to the degrees modifications by comparing vector $[2, 2, 1, 1, 0, 2, 0]$ with vector $[2, 2, 1, 1, 0, 1, 1]$.

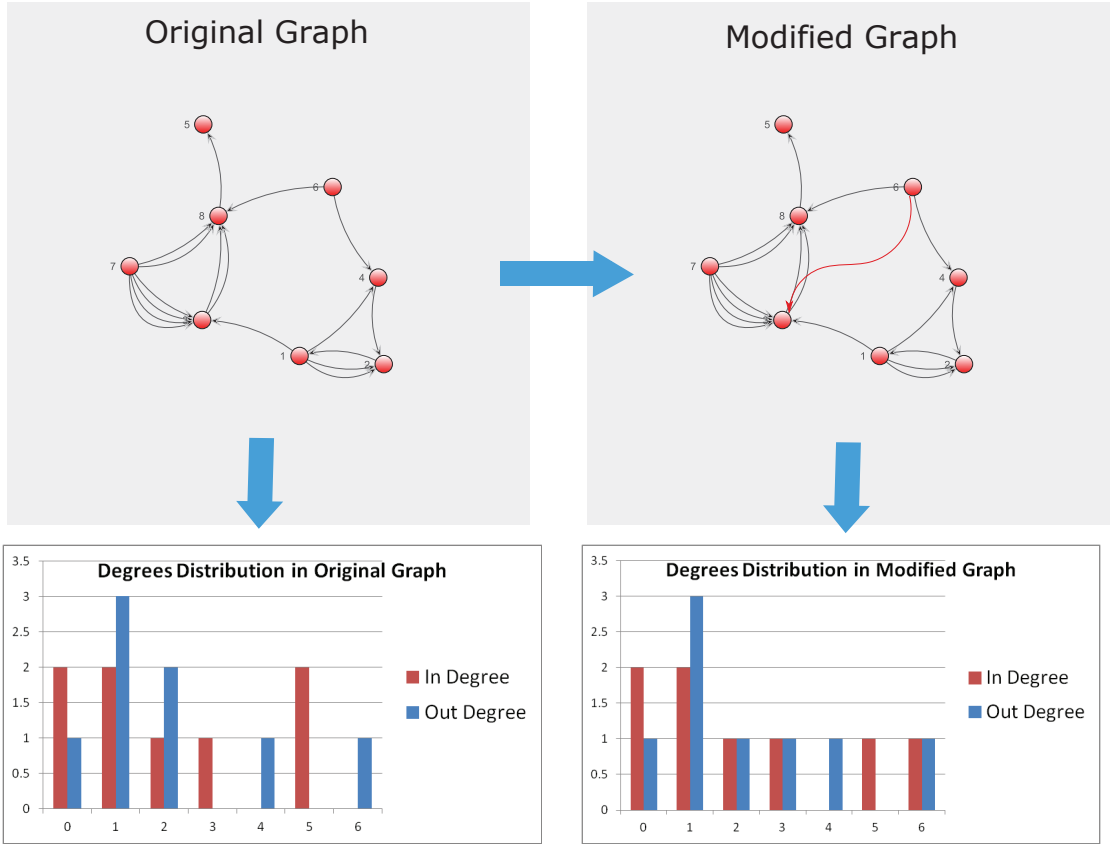


Figure 6.15: Degrees distribution evolution when modifying graph.

The utility losses obtained by the three methods are normalized against the worst value for each graph in order to obtain a value between 0 and 1, 1 corresponding to the highest utility loss obtained.

6.6 Experiments

6.6.1 Baseline: Random data perturbation

Graph data anonymization has been performed in the past mainly for simple graphs by using techniques based for example on k-anonymity.
As described in G29 [April 2014], in practice, when there is a need to deliver anonymized complex data, randomly adding noise is a commonly used technique. The common

mistakes pointed out by the European opinion (G29 [April 2014]) when using randomized techniques are:

- adding inconsistent noise - in this case added "out-of-scale" noise could be filtered out
- assuming noise addition is enough - not adding the right level of noise could be a risk for privacy like e.g., in the Netflix case

We compare our results with results obtained when perturbing data with random noise generated on a scale between 0 and the maximum value allowed used for our learning process (we have used a maximum of 0.1 for the add operation and of 0.5 for the rest of the operations).

6.6.2 Datasets

We have used for the evaluation two anonymized call logs datasets, one from the well known Enron dataset and the other one from Twitter. We have extracted chronologically by using the projection operator O, 100 subgraphs corresponding each of them to a period of time. We have used for learning the first 10 subgraphs extracted. The result has been evaluated on the following 90 subgraphs representing 90% of the dataset used in our evaluation.
The advantage of our approach is that we do not need access to the entire dataset to determine the anonymization strategy. Only the learning set is needed. The method can then be applied directly to newly created data.
Parameterized anonymization method described in Algorithm 5 is performed $T = 10$ times for each graph in the learning set (corresponding to the Monte-Carlo method applied). The obtained utility loss and privacy risk is the mean of the values corresponding to each iteration on the whole dataset. We next describe the two datasets used for experiments.

6.6.2.1 Enron dataset

The Enron corpus, was made public during the legal investigation concerning the Enron corporation and it contains a large set of email messages. Enron dataset has been described in Klimt and Yang [2004] and it contains 619,446 messages. In Enron dataset, two types of users are represented:

- "core users" for which the entire set of communications is represented in the dataset

- non "core users" only connected with core users. Only a part of their communications is present in the dataset

We use for our experiments a cleaned dataset retrieved from Enron Cleaned Dataset and containing 164,081 communications involving 28,062 users and 156 core users which occurred between January 2001 and March 2002. The communication dataset used is cleaned of invalid timestamps or broadcast messages and is represented in a format of type "unix_timestamp; sender; receiver". The training dataset is formed by projecting the first 10 subgraphs, each one of them containing 1640 interactions. The testing dataset constructed contains 90 graphs of 1640 interactions.

6.6.2.2 Twitter dataset

A second dataset crawled from the social platform Twitter has been used for evaluation. It contains around 1,530,000 anonymized interactions occurred on a period of several months between April 2009 and November 2009. The dataset is the same as the one used in Chapter 5 for experimentation of the methodology on simple graphs, except that we consider each interaction as a new oriented timestamped link.
The dataset is represented in the same format as the Enron dataset: "unix_timestamp; sender; receiver". We considered that a communication occurred between the sender and the receiver, if the sender sent a message containing a "@receiver" indication. Based on this, we have generated from the crawled data an anonymized database containing the communications extracted from the Twitter messages.
We have constructed the timestamped oriented multiple graphs used for experimentation by extracting, ordered by timestamp, the first 200,000 interactions and forming 100 graphs. Each extracted graph has 2000 interactions and the training dataset contains 10 graphs. The testing dataset contains the remaining 90 graphs.

6.6.3 Results

6.6.3.1 Learning with one parameter and one operation - $p = 1$

We have started our evaluation by performing an exhaustive search among a very large set of possible values for parameter θ when $p = 1$ and when only add operations are performed. In this case, the search is possible among all possible values and it allows us to compare the real minimum of the empirical loss and the one found by using optimization algorithms. It also allows us to compare the behavior of the training set to the behavior of the testing set. We have used privacy CSG (described in 6.4.1) and utility

CDD (described in 6.5.2). The parameter k was set at 5 and the length used for the CSG sequences generation was 5.

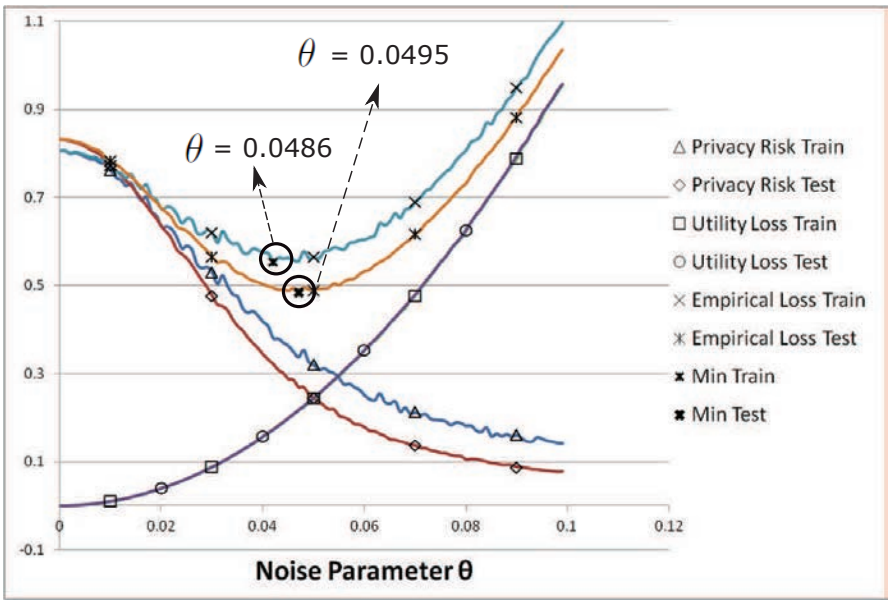


Figure 6.16: Exhaustive search for minimum empirical loss on training dataset and testing dataset: privacy risk is decreasing while utility loss is increasing. The behavior of the training set is similar to the behavior of the testing set.

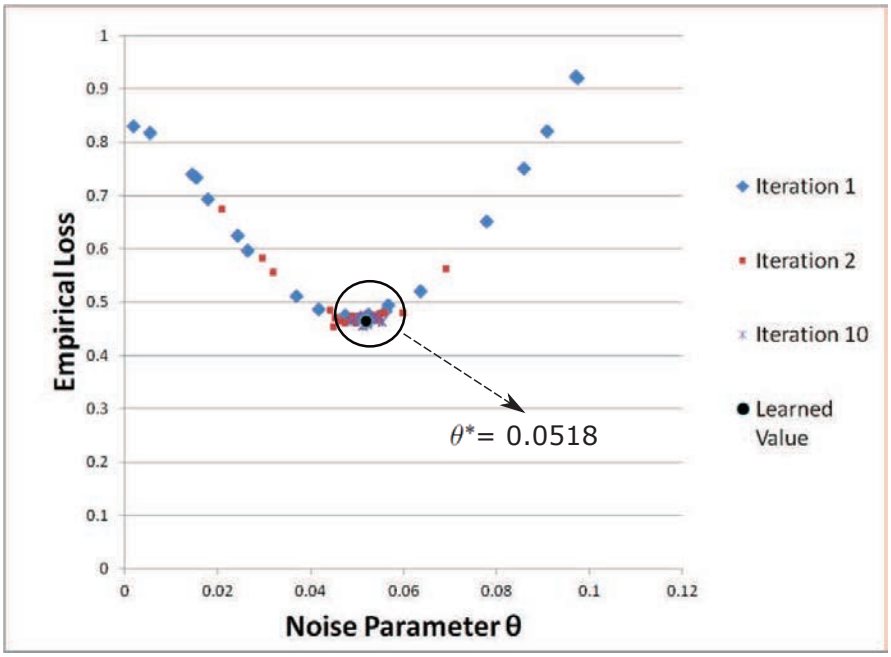


Figure 6.17: Learning with EDA on the training set: initial population is initialized to 20 possible values between 0 and 0.1. Parameter learned after 10 iterations is close to the one obtained with the exhaustive method.)

Figure 6.16 shows the behavior of the empirical loss according to the quantity of added noise. The privacy risk is decreasing while the quantity of noise increases. In the same time, as expected, the loss in the utility of data increases. The behavior is very close for the training set of graphs and for the testing set of graphs. For the utility variation the two curves are almost superimposed. The argument of the minimum found when searching among all possible parameters is $\theta = 0.0486$ for the training set and $\theta = 0.0495$ for the testing set.

Figure 6.17 and Figure 6.18 show the different steps when using optimization algorithms like EDA (described in 4.6.1) and respectively GA (described in 4.6.2) to find θ^* by sampling only on a limited number of points the empirical loss. Initial model for parameter θ has been randomly initialized to 20 values ($N = 20$) between 0 and 0.1 for EDA as well as for GA (Iteration 1).

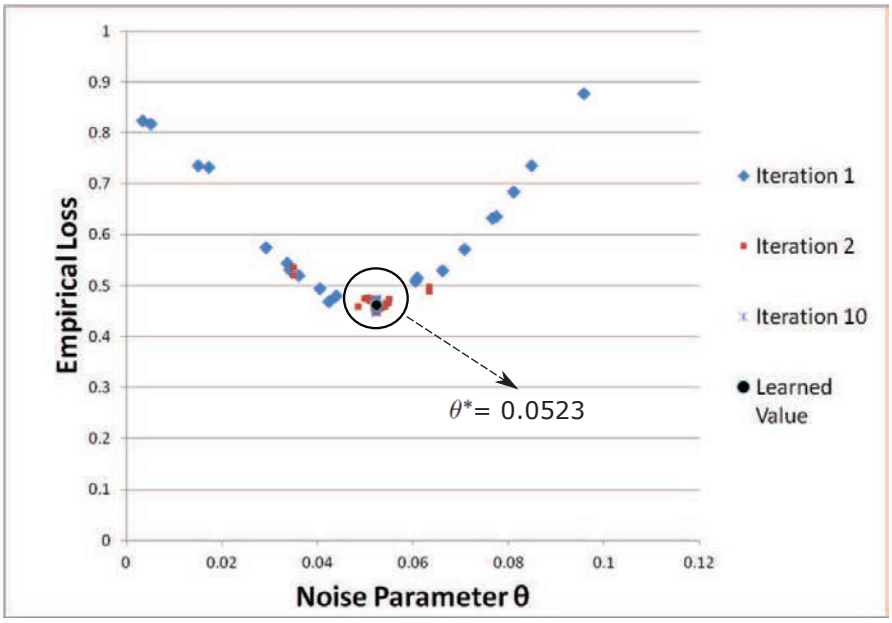


Figure 6.18: Learning with Genetic Algorithm on the training set: initial population is initialized to 20 possible values between 0 and 0.1.

The approximation of the parameter found after 10 iterations with EDA is $\theta^* = 0.0518$ and the one found with GA is $\theta^* = 0.0523$. For both of them, the corresponding empirical loss is very close to the one found when performing an exhaustive search. The parameter found with EDA is closer to the value of the exhaustive search.

When using only one parameter for anonymization, exhaustive search, i.e. in our case search among all parameters θ with a step of 0.0001 to find the one corresponding to the minimum of the empirical loss is still possible. However, when a combination

of multiple parameters is used, optimization techniques are mandatory. Next section presents experimental results when learning with multiple parameters.

6.6.3.2 Learning with multiple parameters

We have implemented the algorithm for a variable number p of parameters (1, 2, 5 or 10) and for the 4 operations described in 6.3: add, delete, replace left and replace right. In most of our experiments, after 20 iterations, the parameters are acceptably close to each other so that we can consider that the algorithm converged. The mean for the resulting parameters is giving the learned vector, as for example the following vector:

$$\boldsymbol{\theta = (0.009, 0.064, 0.117, 0.249, 0.415, 0.332, 0.632, 0.357)}$$

where each part of the vector $\boldsymbol{\theta}$ corresponds to one of the operations used for anonymization as in the following:

$$\boldsymbol{\theta^{Add}} = (0.009, 0.064)$$

$$\boldsymbol{\theta^{Delete}} = (0.117, 0.249)$$

$$\boldsymbol{\theta^{Replace\_Left}} = (0.415, 0.332)$$

$$\boldsymbol{\theta^{Replace\_Right}} = (0.632, 0.357)$$

The first two values 0.009, 0.064 correspond to the add new edge probability for each group of vertices degrees (θ_1^{add} and θ_2^{add}), 0.009 associated to the vertices with the lowest degrees and 0.064 associated with the vertices with the highest degrees. The next two values 0.117, 0.249 correspond to the delete edge probabilities for each group of vertices degrees (θ_1^{delete} and θ_2^{delete}) and so on.

Most of the learned parameters have this same format where the noise to be added on vertices with higher degrees is more important than the noise to be added on vertices with lower degrees for add and delete operations. This confirms an intuitive result as vertices with high degrees can be easily de-anonymized because of their complex structure. Table 6.1 illustrates the obtained results when learning with different sizes of n and with multiple utilities and privacies. The best model obtained for both datasets corresponds to the maximum number of parameters learned, in this case 40 parameters (10 parameters for each of the four operations).

A more granular fitting of the parameters according to the properties of the modified

Table 6.1: Learning with multiple utilities and privacies - testing set results. Best values for the empirical loss are obtained when learning with 40 parameters with EDA.

Dataset	Anonymization Method	Utility Loss				Privacy Risk			Emp. Loss
		CHG	CDD	DE	Global	CSG	NDD	Global	
Enron	Random method	0.178	0.177	0.330	0.228	0.199	0.012	0.105	0.334
Enron	p=1, n=4, EDA	0.122	0.036	0.264	0.141	0.065	0.000	0.033	0.173
Enron	p=2, n=8, EDA	0.092	0.007	0.165	0.088	0.079	0.001	0.040	0.128
Enron	p=5, n=20, EDA	0.064	0.010	0.116	0.063	0.110	0.003	0.056	0.119
Enron	**p=10, n=40, EDA**	**0.070**	**0.002**	**0.088**	**0.054**	**0.073**	**0.000**	**0.037**	**0.090**
Enron	p=1, n=4, GA	0.058	0.032	0.179	0.090	0.335	0.007	0.171	0.260
Enron	p=2, n=8, GA	0.074	0.044	0.217	0.111	0.276	0.004	0.140	0.252
Enron	p=5, n=20, GA	0.072	0.035	0.188	0.098	0.261	0.006	0.133	0.232
Enron	p=10, n=40, GA	0.042	0.019	0.102	0.054	0.257	0.005	0.131	0.185
Twitter	Random method	0.173	0.173	0.452	0.266	0.097	0.009	0.053	0.319
Twitter	p=1, n=4, EDA	0.036	0.032	0.378	0.149	0.126	0.000	0.063	0.212
Twitter	p=2, n=8, EDA	0.030	0.022	0.291	0.114	0.054	0.000	0.027	0.141
Twitter	p=5, n=20, EDA	0.014	0.008	0.116	0.046	0.120	0.001	0.061	0.107
Twitter	**p=10, n=40, EDA**	**0.001**	**0.000**	**0.007**	**0.003**	**0.131**	**0.001**	**0.066**	**0.069**
Twitter	p=1, n=4, GA	0.031	0.026	0.334	0.130	0.164	0.001	0.082	0.213
Twitter	p=2, n=8, GA	0.015	0.010	0.180	0.069	0.181	0.002	0.091	0.160
Twitter	p=5, n=20, GA	0.021	0.017	0.234	0.090	0.124	0.001	0.062	0.153
Twitter	p=10, n=40, GA	0.022	0.019	0.210	0.083	0.139	0.001	0.070	0.153

vertices results into a better minimization of the empirical loss.
In our anonymization function, we have made the hypothesis that the quantity of noise to be added for each type of operation is strictly related to the vertices degrees. This hypothesis is verified by the results as we obtain better values for the empirical loss when learning with $p = 10$ groups ($n = 40$ parameters) corresponding to the vertices degrees.

We compared the results obtained with the ones obtained by using a random data perturbation or GA optimization and model fitting the minimum empirical loss by using EDA outperforms random perturbation or GA obtained model.

6.6.3.3 Generalization

The described method aims at adapting to different utilities or types of attacks configured in the blackboxes. We have tested the algorithm by learning with only one utility (DE) and with both privacies (CSG and NDD). Table 6.2 shows the obtained results on the testing set.

The results show a global improvement of the value of the utility DE when compared to the results obtained when learning with all utilities and all privacies (in Table 6.1). The value of the utility DE is also improved when learning with multiple parameters. The best results are obtained when learning with EDA on the Enron dataset as well as on the Twitter dataset. The improvement compared with the random baseline or with

Table 6.2: Learning with one utility and all privacies: adaptability of the method.

Dataset	Anonymization Method	Utility Loss DE	Privacy Risk			Empirical Loss
			CSG	NDD	Global	
Enron	Random method	0.330	0.199	0.012	0.105	0.435
Enron	p=1, n=4, EDA	0.109	0.082	0.001	0.042	0.150
Enron	p=2, n=8, EDA	0.088	0.129	0.002	0.065	0.153
Enron	p=5, n=20, EDA	0.067	0.185	0.005	0.095	0.162
Enron	**p=10, n=40, EDA**	**0.062**	**0.180**	**0.004**	**0.092**	**0.154**
Enron	p=1, n=4, GA	0.074	0.272	0.007	0.140	0.214
Enron	p=2, n=8, GA	0.094	0.303	0.007	0.155	0.249
Enron	p=5, n=20, GA	0.082	0.402	0.018	0.210	0.292
Enron	p=10, n=40, GA	0.064	0.371	0.017	0.194	0.258
Twitter	Random method	0.452	0.097	0.009	0.053	0.505
Twitter	p=1, n=4, EDA	0.049	0.172	0.005	0.088	0.138
Twitter	p=2, n=8, EDA	0.090	0.130	0.008	0.069	0.159
Twitter	**p=5, n=20, EDA**	**0.037**	**0.213**	**0.012**	**0.113**	**0.150**
Twitter	p=10, n=40, EDA	0.082	0.255	0.010	0.132	0.214
Twitter	p=1, n=4, GA	0.043	0.442	0.046	0.244	0.287
Twitter	p=2, n=8, GA	0.121	0.271	0.003	0.137	0.258
Twitter	p=5, n=20, GA	0.137	0.244	0.004	0.124	0.261
Twitter	p=10, n=40, GA	0.103	0.272	0.010	0.141	0.243

GA baseline is important for the utility DE as well as for the global empirical loss.
This shows that the method adapts well to a new context and gives good results. In this simulation we have used a parameter $\boldsymbol{\theta}$ of dimension 40 (10 parameters for each operation). We have simultaneously used all the utilities and the privacies described for learning.

6.6.3.4 Learning Convergence

Figure 6.19 shows the quality and evolution of the result versus the number of calls sent to the external blackboxes according to the size (N) of the initial population the parameter $\boldsymbol{\theta}$ has been initialized to.
The best value for the empirical loss is obtained for the population $N = 200$. However, the number of calls to the blackboxes is very important in this case. A good model would correspond to a population $N = 30$. The minimum for the empirical loss in this case is close to the one obtained with $N = 200$ after less than 190 external calls.
This shows that the choice of the population size is important and could improve the efficiency of the algorithm.

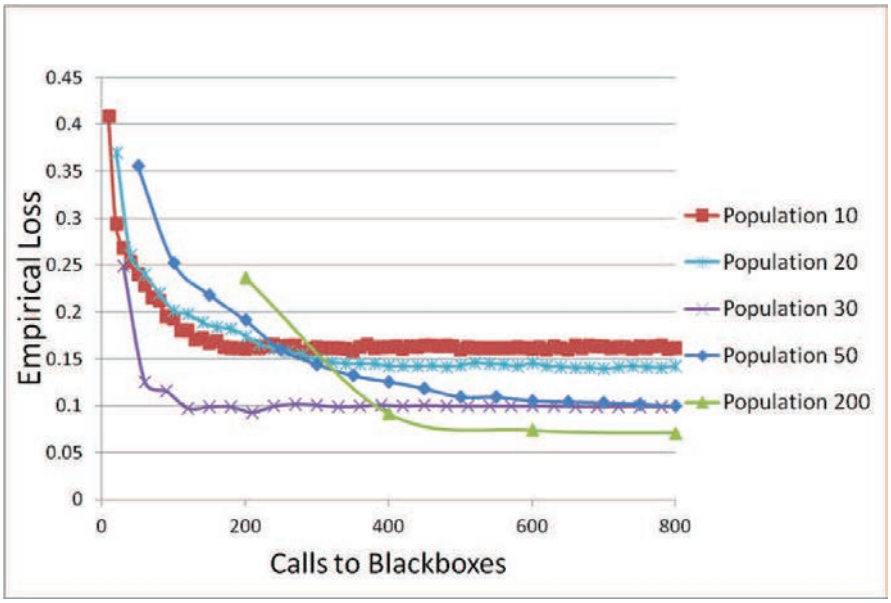


Figure 6.19: Algorithm efficiency versus external blackboxes calls: best value obtained for $N = 200$, best compromise obtained with $N = 30$.

6.6.3.5 Algorithms Complexity

This chapter introduces several algorithms, and among them the parameterized anonymization function for complex graphs described in Algorithm 5. In order to evaluate the privacy level, two algorithms are described: Algorithm 6 and Algorithm 7.

The anonymization algorithm consists in four operations. The first three operations (replace left, replace right and delete) are performed by iterating through all the existing edges (a total of U edges). The complexity of this operation is $O(U)$. The "add" operation is performed by iterating through all possible combinations of vertices and leads to a complexity $O(S^2)$. However, for each add iteration a new edge is added in real cases with a small probability therefore the worst case is never encountered in practice.

The Algorithm 6 is evaluating the privacy score based on the k-Anonymity concept for a set of fingerprints. It first iterates through all the fingerprints (maximum equal to the number of vertices S) and increments privacy risk for each user if necessary therefore it correspond to a complexity $O(S^2)$. The second step of the algorithm (lines 18 - 25) calculates the global privacy risk and in order to do that iterates through all the users S. The complexity of this algorithm corresponds then to $O(S^2)$.

The Algorithm 7 describes the generation of the fingerprints corresponding to each user. For each vertex (a total of S), the algorithm iterates through each call (a maximum of U) and generates the corresponding fingerprint(s). The complexity of this algorithm is then of $O(S \times U)$.

6.7 Conclusion

We have proposed in this chapter a generic method for call logs anonymization based on machine learning. This type of data can be represented as an oriented multiple timestamped graph. Most of the existing anonymization methods on graphs are related to simple graphs. The idea was to find a parameterized anonymization function corresponding to the best compromise between utility loss and privacy risk. The parameterized function is learned on a training set, therefore it can apply to new generated data.
We have implemented our algorithm in Java and tested it on two real datasets of call logs issued from Enron and Twitter. The parameterized function learned on the training set has been tested on the remaining dataset with successful results and outperforms baseline methods. The results improve according to the number of parameters used for the anonymization function. An interesting future direction would be to study this improvement when diversifying the types of parameters used for learning. As we deal with timestamped interactions, another direction is to learn an anonymization function based on time depending parameters. Results from this chapter are currently submitted for publication.

Chapter 7

Conclusion

Contents

7.1 Contributions

The main goal of my thesis was to study data anonymization when dealing with structures issued from communication interactions. The motivation of this thesis was the need to externalize data for analysis or other purposes or to publish it. When data is complex and when it can be modeled as a dynamic graph with multiple oriented links, external information can be easily combined with anonymized data allowing user's re-identification. This vulnerability has been showed in the past and leaded to a limitation of data publishing for research or other purposes. From the beginning of my thesis, the idea of finding an anonymization method easily adaptable to new contexts and to different attacks was a main challenge and motivated my work. The experimentation of the methods on data issued from telecommunication equipments (like e.g. CDRs) was another target of my research.

A part of my work has been dedicated to the analysis and comprehension of existing publications on data anonymization and particularly in graph data anonymization. Chapter 2 describes the state of the art for data issued from communication interactions anonymization.

The main contribution of Chapter 3 is related to data's vulnerability analysis when taking into account its temporal component. We showed that existing techniques are not robust to data decomposition in time intervals. We then proposed to apply existing methods on each part of data projected on a time interval. A patent have been filed with results from this chapter (Hacid and Maag [2014]).

One of the main contributions of my thesis is related to the methodology described in Chapter 4 which aims to model graph data anonymization as an optimization problem based on machine learning. This optimization problem aims at finding the best parameterized anonymization function adapted to a given context in order to find a compromise between utility loss and privacy protection. The analysis to be made on data and the families of privacy attacks are modeled as blackboxes. The anonymizer does not have access to the analysis methods but is able to obtain the output of the analysis measure. Our idea is to find a parameterized anonymization function corresponding to the best compromise between utility loss and privacy risk. We learn this parameterized function on a training set; therefore it can be applied to new generated data.

In Chapter 5 the methodology is applied with success on simple graph data and comparison with prior art shows the method adapts much better to new context. Methodology is applied on datasets retrieved from Twitter and Amazon with a successful result. The behavior of data in the testing set is similar to data in the training set. Results of this chapter have been published at the 28th IEEE International Conference on Advanced Information Networking and Applications (Maag et al. [2014]).

Chapter 6 is coming back to the original problem which is the anonymization of data issued from communication interactions. Parameterized anonymization functions are learned on oriented labeled temporal graphs and they give successful results when tested on the rest of data. In order to test the learning method in the context of temporal graphs, we defined new de-anonymization techniques based on the temporal component of the graph to be anonymized, mainly for implementing the blackboxes used in the learning phase and in the testing phase. The methodology applied on temporal graphs has been compared with baseline methods and showed its effectiveness on real datasets issued from Twitter and Enron. The results from this chapter are currently submitted for publication.

7.2 Perspectives

Data anonymization is a complex problem and it is very difficult to guarantee that when releasing it, individual's privacy will not be infringed. Today there is no method allowing a complete protection of individuals while providing useful datasets. Privacy risks still exist for correlated data even when using the promising differential privacy based guarantees. I hope the work performed during my thesis will motivate further research in order to try to find a universal method for data anonymization able to provide good privacy guarantees while preservig data's utility.

We have proposed an adaptable method for finding the best anonymization function fitting a certain context and minimizing utility loss while reducing privacy risks. The results improve according to the number of parameters used for the anonymization function. An interesting future direction would be to study this improvement while diversifying the types of parameters used for learning. This could lead to a significant improvement of the objective function.

Differential privacy guarantees have already been applied in the context of data release. Most of the differential privacy techniques are based on adding Laplacian noise to the dataset. In Comas [2013] a synergy between k-anonymity and ϵ-differential privacy has been described. In this paper, for data publication, utility has been improved when applying k-anonymity methods before achieving ϵ-differential privacy. It has been shown that for data publication, the k-anonymity family of models is powerful enough to achieve ϵ-differential privacy. Another promising direction for my work consists in trying to find a synergy between the parameterized functions learned with our methodology and ϵ-differential privacy based guarantees.

The results obtained in my thesis will be used in my future work as a researcher in Alcatel-Lucent Bell Labs. A certain number of applications are envisaged for the algorithms and methods proposed in this manuscript. My work leaded me to the definition of new attack methods for instantiating the blackbox containing the potential attacks. These instantiated attacks were based on a re-identification of a user in a given graph when holding a previous knowledge about the user's interactions. This work on pattern recognition in a given structure will be used in my future work.

Methodology allowing one to find the best adapted anonymization techniques matches perfectly with industrial projects needs dealing with privacy concerns about data containing users information. Moreover, the work related to oriented multiple timestamped graphs is close to the needs of information anonymization issued from telecommunication equipments or needs of data mining algorithms on this type of information.

Bibliography

Charu C Aggarwal, Yao Li, and Philip S Yu. On the hardness of graph anonymization. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1002–1007. IEEE, 2011.

Faraz Ahmed, Rong Jin, and Alex X Liu. A random matrix approach to differential privacy and structure preserved social network graph publishing. *arXiv preprint arXiv:1307.0475*, 2013.

Tristan Allard, Benjamin Nguyen, and Philippe Pucheral. Safe realization of the generalization privacy mechanism. In *Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on*, pages 16–23. IEEE, 2011.

Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM, 2007.

Michael Barbaro and Tom Zeller. A face is exposed for AOL searcher no. 4417749. *New York Times*, 9, 2008.

Alain Barrat and Martin Weigt. On the properties of small-world network models. *The European Physical Journal B-Condensed Matter and Complex Systems*, 13(3):547–560, 2000.

Roberto J Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228. IEEE, 2005.

C. Berge. *Graphes et hypergraphes*. Dunod Université. Dunod, 1973. ISBN 9782040097554.

Michele Berlingerio, Francesco Calabrese, Giusy Di Lorenzo, Rahul Nair, Fabio Pinelli, and Marco Luca Sbodio. Allaboard: a system for exploring urban mobility and optimizing public transport using cellphone data. In *Machine learning and knowledge discovery in databases*, pages 663–666. Springer, 2013.

Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Privacy in dynamic social networks. In *Proceedings of the 19th international conference on World wide web*, pages 1059–1060. ACM, 2010.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *Privacy, Security, and Trust in KDD*, pages 33–54. Springer, 2009.

Jordi Soria Comas. *Improving data utility in differential privacy and k-anonymity*. PhD thesis, Department of Computer Engineering and Mathematics, Taragona, 2013.

Graham Cormode. Individual privacy vs population privacy: Learning to attack anonymization. *arXiv preprint arXiv:1011.2511*, 2010.

Graham Cormode, Divesh Srivastava, Ting Yu, and Qing Zhang. Anonymizing bipartite graph data using safe groupings. *Proceedings of the VLDB Endowment*, 1(1):833–844, 2008.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

Roger Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science, Special Issue*, 15:131137, 1987.

Enron Cleaned Dataset. Enron cleaned dataset. http://sociograph.blogspot.fr/2011/04/communication-networks-part-1-enron-e.html. Accessed: 2013-11-25.

European Parliament. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, October 1995.

Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4): 14, 2010.

G29. Opinion 05/2014 on anonymization techniques. http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf, April 2014. Working party on the protection of individuals with regard to the processing of personal data.

David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989.

Romain Guigoures and Marc Boullé. Segmentation of towns using call detail records. In *NetMob Workshop at IEEE SocialCom*, 2011.

Romain Guigourès, Marc Boullé, and Fabrice Rossi. Étude des corrélations spatio-temporelles des appels mobiles en france. In *Extraction et Gestion des Connaissances*, pages 437–448, 2013.

Adrien Guille and Hakim Hacid. A predictive model for the temporal dynamics of information diffusion in online social networks. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 1145–1152. ACM, 2012.

Hakim Hacid and Laura Maag. Systems and methods for data anonymization, December 25 2014. US Patent 20,140,380,489.

Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, 2008.

JUNG Library. Java universal network/graph framework. http://jung.sourceforge.net/. Visited the 17 of October 2014.

Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204. ACM, 2011.

Bryan Klimt and Yiming Yang. Introducing the Enron Corpus. In *CEAS*, 2004.

Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph. *Data mining and knowledge discovery*, 11(3):243–271, 2005.

David Leoni. Non-interactive differential privacy: a survey. In *Proceedings of the First International Workshop on Open Data*, pages 40–52. ACM, 2012.

Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

Ninghui Li, Wahbeh H Qardaji, and Dong Su. Provably private data anonymization: Or, k-anonymity meets differential privacy. *CoRR, abs/1101.2604*, 49:55, 2011.

Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526. ACM, 2009.

Yidong Li and Hong Shen. Anonymizing graphs against weight-based attacks. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 491–498. IEEE, 2010.

Yidong Li and Hong Shen. Preventing identity disclosure in hypergraphs. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 659–665. IEEE, 2011a.

Yidong Li and Hong Shen. Anonymizing hypergraphs with community preservation. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*, pages 185–190. IEEE, 2011b.

Qining Lin and Yan Wan. Mobile customer clustering based on call detail records for marketing campaigns. In *Management and Service Science, 2009. MASS'09. International Conference on*, pages 1–4. IEEE, 2009.

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. ACM, 2008.

Lian Liu, Jie Wang, Jinze Liu, and Jun Zhang. Privacy preserving in social networks against sensitive edge disclosure. *Computer*, 2008.

Maria Laura Maag, Ludovic Denoyer, and Patrick Gallinari. Graph anonymization using machine learning. In *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, pages 1111–1118. IEEE, 2014.

Darakhshan J Mir, Sibren Isaacman, Ramón Cáceres, Margaret Martonosi, and Rebecca N Wright. Dp-where: Differentially private modeling of human mobility. In *Big Data, 2013 IEEE International Conference on*, pages 580–588. IEEE, 2013.

Kato Mivule and Claude Turner. An investigation of data privacy and utility preservation using knn classification as a gauge. *arXiv preprint arXiv:1309.3964*, 2013.

Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the Netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45 (2):167–256, 2003.

Kieron O'Hara. Transparent government, not transparent citizens: a report on privacy and transparency for the cabinet office. 2011.

Paul Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57:1701, 2010.

Orange. D4D challenge. http://www.d4d.orange.com. Accessed: 2014-06-11.

Wei Peng, Feng Li, Xukai Zou, and Jie Wu. Seed and grow: An attack against anonymized social networks. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 587–595. IEEE, 2012.

Nicola Perra and Santo Fortunato. Spectral centrality measures in complex networks. *Physical Review E*, 78(3):036107, 2008.

Julie K Petersen. *The Telecommunications Illustrated Dictionary*. Hoboken: CRC Press, 2002.

Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98. ACM, 2011.

Kumar Sharad and George Danezis. De-anonymizing D4D datasets. In *Workshop on Hot Topics in Privacy Enhancing Technologies, Bloomington, Indiana, USA*, 2013.

Kumar Sharad and George Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 47–58. ACM, 2014.

Reza Sherkat, Jing Li, and Nikos Mamoulis. Efficient time-stamped event sequence anonymization. *ACM Transactions on the Web (TWEB)*, 8(1):4, 2013.

Yi Song, Sadegh Nobari, Xuesong Lu, Panagiotis Karras, and Stéphane Bressan. On the privacy and utility of anonymized social networks. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, pages 246–253. ACM, 2011.

Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. Enhancing data utility in differential privacy via microaggregation-based k k-anonymity. *The VLDB JournalThe International Journal on Very Large Data Bases*, 23(5):771–794, 2014.

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

György Szarvas, Richárd Farkas, and Róbert Busa-Fekete. State-of-the-art anonymization of medical records using an iterative machine learning framework. *Journal of the American Medical Informatics Association*, 14(5):574–580, 2007.

Chih-Hua Tai, S Yu Philip, De-Nian Yang, and Ming-Syan Chen. Structural diversity for privacy in publishing social networks. In *SDM*, pages 35–46. SIAM, 2011a.

Chih-Hua Tai, Peng-Jui Tseng, Philip S Yu, and Ming-Syan Chen. Identities anonymization in dynamic social networks. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1224–1229. IEEE, 2011b.

Chih-Hua Tai, Philip S Yu, De-Nian Yang, and Ming-Syan Chen. Privacy-preserving social network publication against friendship attacks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1262–1270. ACM, 2011c.

Christine Task and Chris Clifton. A guide to differential privacy theory in social network analysis. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 411–417. IEEE Computer Society, 2012.

United States Congress. Video privacy protection act. http://www.law.cornell.edu/uscode/text/18/2710, 1988. Accessed: 2014-11-6.

Stef van den Elzen, DH Jorik Blaas, Danny Holten, Jan-Kees Buenen, Jarke J van Wijk, Robert Spousta, Anna Miao, Simone Sala, and Steve Chan. Exploration and analysis of massive mobile phone data: A layered visual analytics approach. In *Proceedings of the 3rd International Conference on the Analysis of Mobile Phone Datasets (NetMob'13). Boston, MA, USA*, 2013.

Stef Van den Elzen, DH Jorik Blaas, Danny Holten, Jan-Kees Buenen, Jarke J van Wijk, Robert Spousta, Anna Miao, Simone Sala, and Steve Chan. Exploration and analysis of massive mobile phone data: A layered visual analytics approach. In *Proceedings of the 3rd International Conference on the Analysis of Mobile Phone Datasets (NetMob'13). Boston, MA, USA*, 2013.

Sergey Vinogradov and Alexander Pastsyak. Evaluation of data anonymization tools. In *DBKDA*, pages 163–168, 2012.

Staal A Vinterbo. Privacy: a machine learning view. *Knowledge and Data Engineering, IEEE Transactions on*, 16(8):939–948, 2004.

Martin J Wainwright, Michael I Jordan, and John C Duchi. Privacy aware learning. In *Advances in Neural Information Processing Systems*, pages 1430–1438, 2012.

Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Advances in Knowledge Discovery and Data Mining*, pages 329–340. Springer, 2013.

Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

Wikipedia. Estimation of distribution algorithm — wikipedia, the free encyclopedia, 2013. URL http://en.wikipedia.org/w/index.php?title=Estimation_of_distribution_algorithm&oldid=550481224. [Online; accessed 12-January-2015].

Alden H Wright and Sandeep Pulavarty. On the convergence of an estimation of distribution algorithm based on linkage discovery and factorization. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 695–702. ACM, 2005.

Xintao Wu, Xiaowei Ying, Kun Liu, and Lei Chen. A survey of privacy-preservation of graphs and social networks. In *Managing and Mining Graph Data*, pages 421–453. 2010.

Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, pages 739–750, 2008.

Xiaowei Ying and Xintao Wu. Graph generation with prescribed feature constraints. In *Proceedings of the 9th SIAM Conference on Data Mining*, pages 966–977, 2009.

Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: differentially private model fitting using genetic algorithms. In *SIGMOD Conference*, pages 665–676, 2013.

Lijie Zhang and Weining Zhang. Edge anonymity in social network graphs. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, CSE '09, pages 1–8. IEEE Computer Society, 2009.

Qingfu Zhang and H. Muhlenbein. On the convergence of a class of estimation of distribution algorithms. *Evolutionary Computation, IEEE Transactions on*, 8(2):127–136, April 2004. ISSN 1089-778X.

Elena Zheleva. *Prediction, Evolution and Privacy in Social and Affiliation Networks*. PhD thesis, University of Maryland, College Park, 2011.

Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, pages 153–171, 2007.

Elena Zheleva and Lise Getoor. To join or not to join : the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 531–540. ACM, 2009.

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

Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, 28(1):47–77, 2011.

Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism : A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1): 946–957, 2009.