Differential and Blowfish Privacy applied on graph, transactional and image datasets

Élie Alchicha

Differential and Blowfish Privacy applied on graph, transactional and image datasets

by

Elie Chicha

December, 2021

A thesis submitted to the
Graduate School
of the
University of Pau and Pays de l'Adour (UPPA)
for the degree of
Doctor of Philosophy

Committee in charge:

Pr. Richard CHBEIR, Supervisor
Dr. Bechara AL BOUNA, Co-Supervisor
Pr. Lionel BRUNIE, Reviewer
Pr. Alban GABILLON, Reviewer
Pr. Djamal BENSLIMANE, Examiner
Pr. Allel HADJALI, Examiner
Dr. Mohamed NASSAR, Examiner

Abstract

Digital data is playing crucial role in our daily life in communicating, saving information, expressing our thoughts and opinions and capturing our precious moments as digital pictures and videos. Digital data has enormous benefits in all the aspects of modern life but forms also a threat to our privacy. In this thesis, we consider three types of online digital data generated by users of social media and e-commerce customers: graphs, transactional, and images. The graphs are records of the interactions between users that help the companies understand who are the influential users in their surroundings. The photos posted on social networks are an important source of data that need efforts to extract. The transactional datasets represent the operations that occurred on e-commerce services.

We rely on a privacy-preserving technique called Differential Privacy (DP) and its generalization Blowfish Privacy (BP) to propose several solutions for the data owners to benefit from their datasets without the risk of privacy breach that could lead to legal issues. These techniques are based on the idea of recovering the existence or non-existence of any element in the dataset (tuple, row, edge, node, image, vector, ...) by adding respectively small noise on the output to provide a good balance between privacy and utility.

In the first use case, we focus on the graphs by proposing three different mechanisms to protect the users' personal data before analyzing the datasets. For the first mechanism, we present a scenario to protect the connections between users (the edges in the graph) with a new approach where the users have different privileges: the VIP users need a higher level of privacy than standard users. The scenario for the second mechanism is centered on protecting a group of people (subgraphs) instead of nodes or edges in a more advanced type of graphs called dynamic graphs where the nodes and the edges might change in each time interval. In the third scenario, we keep focusing on dynamic graphs, but this time the adversaries are more aggressive than the past two scenarios as they are planting fake accounts in the dynamic graphs to connect to honest users and try to reveal their representative nodes in the graph.

In the second use case, we contribute in the domain of transactional data by presenting an existed mechanism called Safe Grouping. It relies on grouping the tuples in such a way that hides the correlations between them that the adversary could use to breach the privacy of the users. On the other side, these correlations are important for the data owners in analyzing the data to understand who might be interested in similar products, goods or services. For this reason, we propose a new mechanism that exposes these correlations in such datasets, and we

prove that the level of privacy is similar to the level provided by Safe Grouping.

The third use-case concerns the images posted by users on social networks. We propose a privacy-preserving mechanism that allows the data owners to classify the elements in the photos without revealing sensitive information. We present a scenario of extracting the sentiments on the faces with forbidding the adversaries from recognizing the identity of the persons.

For each use-case, we present the results of the experiments that prove that our algorithms can provide a good balance between privacy and utility and that they outperform existing solutions at least in one of these two concepts.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Pr. Richard Chbeir and co-supervisor Dr. Bechara Al Bouna for the continuous support of my Ph.D study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Beside my supervisors, I would like to acknowledge and give my warmest thanks to Dr. Mohamed Nassar, his guidance helped me in all the time of research. I would also like to thank my committee members for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

From the bottom of my heart, I would like to give special thanks to my parents Caroline and Nassif, my family as a whole, and my friends for their continuous support and understanding when undertaking my research and writing my project. Your prayer for me was what sustained me this far.

I would also like to acknowledge the National Council for Scientific Research of Lebanon (CNRS-L) and Univ. Pau & Pays Adour, UPPA - E2S, LIUPPA for granting a doctoral fellowship to Elie Chicha.

Finally, I would like to thank God, for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree. I will keep on trusting you for my future.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

"Data is the new oil" is a well-known quote used to express the transition from the industrial era to the digital age. In the formal, oil was the most critical resource to build wealth. While in the latter, data is a crucial element to develop and expand any successful business in almost every sector.

Data and oil also have many similar characteristics. The first one is that we cannot capitalize any of them in their raw form. Before being consumed by the final customer, the oil needs to be refined into gasoline, diesel, kerosene, etc. Data also cannot form a valuable asset by itself. Looking into millions or billions of tuples of a dataset on a screen in front of you will not help you expand your customer base, increase your profits or make the right decisions in your business management. You need to analyze these tuples, to search for helpful information leading to the right decision-making choices.

Amazon is one of the Tech giants with a Market Cap that exceeds the $1.6 Trillion. It was not about luck but about analyzing the enormous size of data produced by their customers in the best possible way to recommend to each customer what makes them buy more and more from that platform. Jeff Bezos, the founder of Amazon, used the term "customer obsession" to express the company's most important priority "figure out what they want, what's important to them." Many articles and documentaries could be fined that tracked their rise through the prism of being a data collector. One of the former executives said, "They happen to sell products, but they are a data company." [79].

The countries producing crude oil might have the possibility of refining, but this is not the case in many countries, as refining needs significant capital investment. The same could be said about data analysis. To benefit for the maximum from your raw data, sometimes you need a team of experts in machine learning, deep learning, or other techniques in Artificial Intelligence, as well as expensive software solutions. In this case, the data owner prefers to rely on a third party to complete the complex analysis.

Simultaneously, precisely as the governments try as hard as possible to maintain the security and the stability of the main arteries for oil transport, the data owners must protect the individuals from privacy breaches if they are selling, sharing, or publishing data.

Some may ask if data analysis results provided by the data scientist inside the company or from a third party are the best possible output or better results could be reached. Netflix had a supposedly good idea of publishing data to the public and rewarding a $1 million prize to the contestant who develops the best movie recommendations algorithm. The only issue with that idea is that they think only possible contestants will be interested in the published data, but some privacy researchers were also excited about this opportunity. The data contained 100 million tuples of users with their movie ratings, and the identifier attributes were removed. The authors in [121] aimed to prove that eliminating identifier attributes in that dataset is not enough to protect the privacy of individuals. They exploit some public datasets from the IMDb website and voter databases to link the tuples to specified individuals breaching their movies' ratings. This is considered a privacy breach, especially the rating of controversial ones related to religious, political, or sexual orientation issues.

An important revenue for data owners is creating a user-centric business model to sell data to ad agencies, social studies organizations, or pharmaceutical companies if the data is related to diseases. Thus, the data owner might share the data with a third party or publish it to the whole public. Either way, the privacy threats on the individuals represented in the data are the same, then in this thesis, we will use the terms shared, released, and published interchangeably.

Therefore, questions are raising about the threats of privacy breaches. How can you be sure that the third party, buyer, or public won't exploit your data to gain unauthorized knowledge about the individuals represented in the analyzed data?

One big difference between oil and data is that the oil era, sooner or later, will end. In contrast, the era of data as a valuable asset is here to stay with us for a long time. And, as long as this asset is very profitable, more issues about data privacy will appear.

1.1 Digital data in international conflicts

We have explained the importance of data economic and business-wise, but the data is even more vital than that. In war, politics, and diplomatic relations, knowing more about your enemies, opponents, or even allies makes your position more robust and your decision-making easier.

From here, we can find a new comparison between oil and data. Both of them could lead to dangerous international conflicts, especially between the superpowers, because those who obtain more of these two have more chances to surpass their rivals.

Edward Snowden, a well-known whistleblower who was behind the biggest intelligence leak in the American National Security Agency's history that exposed the agency activities in spying on American people [62], has revealed another document in 2015 proving that China has hacked 50 Terabytes of top-secret data about some of the most advanced US military equipment [61, 44]: the only two stealth and fifth-generation F-35 joint strike and F-22 Raptor, the only stealth strategic bomber in the world B-2, in addition to space-based lasers, missile navigation and tracking systems, as well as nuclear submarine/anti-air missile designs.

This example show the role that data could play in the military and weaponry field. The political field is also a hugely important domain that needs information about other parties so you can make your decisions. NSO Group, an Israeli cybersecurity company, has developed spyware called Pegasus [114, 118] dedicated for the governments to spy on terrorist suspects' phones as the company claimed. But, as anyone with little political knowledge would imagine, dozens of governments buy this $50 million product to control and retrieve the data on the phones of political and human rights activists, journalists, lawyers, activists, foreign leaders, and of course, their political opponents. The reports say that the leaked list of victims contains 50,000 phone numbers [32], including several presidents, prime ministers, and ministers. This scandal clarifies how politicians and governments are thirsty to get personal data about other domestic and foreign politicians.

In many other cases, the victims are not specific individuals but a large portion of the public. Facebook is the largest online social media service in the world. In 2006, user privacy

concerns started emerging [48], and since that, every one to two years, a new privacy issue about this company appears. The most massive data breach is the one linking Facebook and a political data-analytics firm named Cambridge Analytica (CA). CA improperly obtained the personal data of 87 million Facebook users without their consent [112]. The data-analytics firm, then, used the data to help two campaigns target election ads using voter data in two major political events in that period, the US presidential election [84] and Brexit referendum [139, 145] (a referendum to ask the electorate whether the country should remain a member of, or leave, the European Union). The results were that the campaigns hiring CA won both the election and the referendum.

The most apparent evidence these days about the roles that data plays in international affairs and national security is the conflict between the USA and China about the fifth-generation technology standard for broadband cellular networks or what is known as the 5G war [67]. The US has warned its European allies who used communications technology provided by Huawei in their networks that they put intelligence relationships at risk. Washington has started a pressure campaign to encourage many countries to stop relying on the Chinese to build their 5G networks. The two most effective efforst were:

- **Clean Network [8]:** to "protect citizens' privacy and companies' sensitive information on 5G mobile networks and secures data across the full range of telecommunications and technology".

- **Criteria for Security and Trust in Telecommunications Networks and Services [92]:** as a tool to determine trustworthiness and security of telecommunications equipment suppliers for governments and network owners or operators.

These campaigns succeeded in some countries [135] like the UK and Australia, where the governments blacklisted the chinese companies from the national 5G networks for security reasons.

We listed these examples to prove the huge influence digital data could play in such a crucial domain as the relations between superpowers not to claim that we are trying to solve these complicated issues. In this thesis, we restrict our scenario on data generated by users on social media and e-commerce websites, how to benefit from this data and protect the privacy of the users at the same time.

1.2 Scenario

To talk more about the privacy issues in the data domain, we will present a scenario about a social network company that adopts the Ad-based business model that needs user-centric designs to grow. All the services related to the social network are free, and, as the American sculptor Richard Serra once said, "if something is free, you're the product". Actually, the real product is the data generated by the users, and the customers are the parties willing to buy the results of the analysis, the queries, or the mechanisms applied to the data.

When using the data in any process, the company faces two main problems:

- **User privacy:** As mentioned previously, sharing the data with a third party to perform the analysis might pose a real threat to the privacy of the users. Thus, our role in this company is to provide privacy-preserving mechanisms that could ensure a good balance between the privacy of the users and the utility of shared data.
- **Heterogeneity of the generated data:** The data generated by social media users is very diverse in its types and forms like text, image, video, location, graphs, microdata, transactional data, etc. In this thesis, we focus on three types of them:
  - Graphs: The main type of data that a social media firm could be interested to exploit is the graphs generated by numerous actions performed by the users: being friends, following, messaging, tagging, blocking, muting,... Furthermore, the graphs constructed from these actions could have different types. The static graph presents the relations between several users in a single time interval. In contrast, a dynamic graph represents these relations in multiple intervals by modifying its edges and adding or removing some of its nodes to describe the relationships in every interval.

    The solution could be offline, where we already have all the instances of the dynamic graph. The online solutions are applied when the company demands that every instance be anonymized approximately in real-time. We cannot wait for all other instances before publishing the current one.

    The attacks also can differ between static and dynamic attacks. In the formal, the adversaries try to extract personal private data from published datasets relying on their auxiliary data. Active attacks occur when the adversaries plant fake accounts and attack the victim till they find their nodes in the published graphs.
  - Images: Other type of data generated by the users is the images. The image could contain a lot of information but extracting data from this type of post is not as easy as extracting it from a text or the relationships between the users. The gallery images of a user could reveal everything about them, starting from their preferred clothes, shoes, watches, sunglasses, and jewelry brands to their favorites restaurants and type of foods. Even more, the political parties, religious institutions, athletics clubs, and non-governmental organizations will be very interested in delivering their messages to people posting images on our social network about these issues.

    These are just an example of what we can uncover from the gallery images of a user. But the problem with the photos is that they don't contain a simple type of data that is easy to detect and analyze. For example, a post like ¨I miss sushi!¨ or a user following 14 fast-food restaurants requires simple algorithms to retrieve valuable data from the post or the user's connections. On the other hand, an image of a family having dinner in a restaurant contains many profitable data: What are they wearing, eating, and drinking, in addition to the type of restaurant that they might prefer and the cost that they are willing to pay to eat outside.

Revealing these pieces of information requires a sophisticated and expensive algorithm. Thus, our company might prefer to send these data to a third party to perform this analysis on the images at an acceptable price. Here comes our mission to apply a privacy-preserving algorithm on the images resulting in a trade-off between privacy where the third party could not identify the faces in the pictures and the utility where valuable data are preserved.

– Transactional data: The company could adopt a new strategy to benefit from all the data. Instead of just selling them to interested parties, the company decided to open its own e-commerce business. This platform targets the users by advertising their favorite products based on their social network data. Selling the products also produces what is called transactional data, which by analyzing it, could help us more and more in recommending the right products to our users.

  Tuples form a transactional dataset; each represents one consumer buying process, containing the consumer id, a number of their quasi-identifiers attributes, the product id, the price, and at least one attribute considered sensitive about the process or the customer. Analyzing these tuples will realize the relation between the attributes that characterize the user, from one side, and the product and its price from the other side. This data mining can help us anticipate that users with similar attributes will be interested in similar products with a specific price range.

  For the same reasons as the past two analyses we have discussed, this analysis needs to be done safely without threatening the consumers' data.

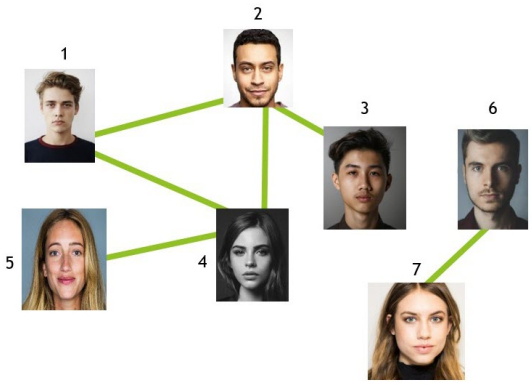
In this thesis, instead of applying a different type of privacy definitions on each of these types of data, we will propose to adopt one privacy definition called Differential Privacy and one of its extensions called Blowfish Privacy on the heterogeneous data.

1.3 Graph Anonymization

1.3.1 Naive anonymization

A communication graph is a graph where vertices represent individuals, and an edge between two individuals exists if communication has happened between the individuals corresponding to the vertices. Social networks and call detail records can be modeled as communication graphs. One way to anonymize the data of a communication graph is to remove the identifiers at the vertices. The goal of an adversary is therefore, to discover the individual corresponding to a node in the graph.

A communication graph $G_c(V_c, E_c)$ can be represented as a database D of size $n = |V_c|$ where each tuple of D corresponds to an individual id. The tuple dimension is $m = |V_c|$ as well. The ith attribute of a tuple t is 1 if $t.\_id$ has communicated with the individual corresponding to node i, and 0 otherwise. A row in D represents the ego network of the corresponding vertex.



id	tuple
Bob (1)	(0, 0, 1, 1, 0, 0, 0)
Alice (2)	(0, 0, 0, 1, 0, 0, 0)
Adam (3)	(1, 0, 0, 1, 1, 0, 0)
Carol (4)	(1, 1, 1, 0, 0, 0, 0)
Roan (5)	(0, 0, 0, 1, 0, 0, 0)
Karl (6)	(0, 0, 0 , 0, 0, 0, 1)
Elyse (7)	(0, 0, 0, 0, 0, 1, 0)

Figure 1: Original graph of social network users and their connections

An example of a communication graph and its corresponding database is shown in Figure 1. Note that we consider a binary communication event (0 or 1). Other models might be explored in future work, for instance, annotating the edges with call frequency, average duration, call time, or other meta-data.

One way to anonymize the data of a communication graph is to remove the identifiers at the vertices. Therefore, the goal of an adversary is to discover the individual corresponding to a node in the graph. Overlapping this naive anonymization is quite simple, using some background knowledge about the community.

Let's say that the graph in Figure 1 represents a small community and all the communications that took place between the individuals inside this community. To protect the individuals' identities, we have removed the identifier attributes from each vertex like the account id, the name, the phone number, and the email address and replaced them with hashed ids. At the same time, we have kept sensitive attributes necessary for the study or the advertisers.

The challenge is to keep the real identity of each user in the graph hidden. Otherwise, an adversary can relate the actual user with the sensitive private data of each node and the communications placed between this user and the other users, which can be considered a serious privacy breach.

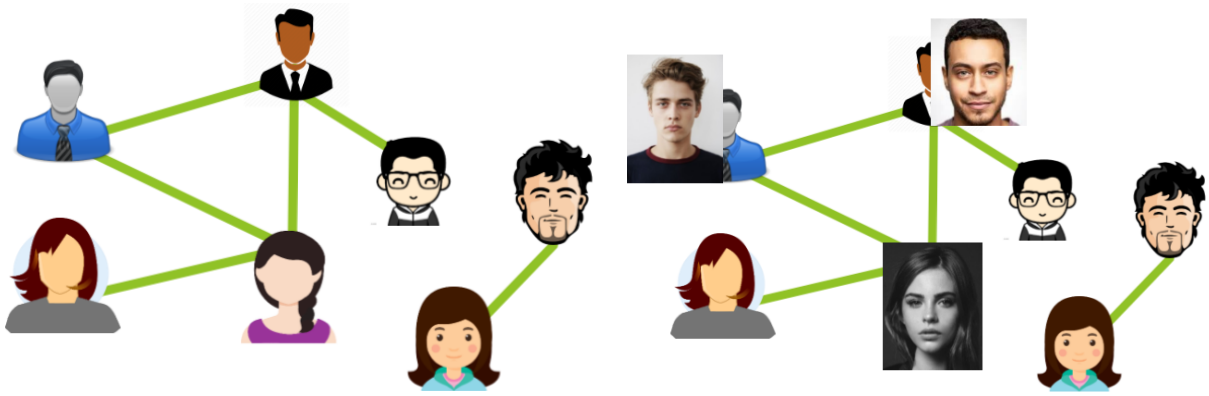
Suppose an adversary has some public or auxiliary knowledge about our community. For example, Bob, Adam, and Carol communicate with each other every day. As we can see in Figure 2, this adversary can find the three nodes communicating with each other, then locating Carol in the graph with 100% accuracy and John and Bob with 50% precision, that's called background attack.

1.3.2 Aspects of the graph anonymization

After explaining the failure of the naive anonymization, we will present our own privacy-preserving mechanisms, but before that, we will list some characteristics of the graphs and their anonymization.

The graph datasets produced by the relationships of the users have two types:

- **Static:** the published version of the graph is released once, and that's it.



(a) Naive anonymization of the graph (b) Exposing the id of 3 nodes.

Figure 2: Naive anonymization failed to protect the users facing a background attack.

- **Dynamic:** The published version is updated at the end of each time interval to represent the connections in this interval.

The attacks on the graphs could also be divided into two types:

- **Passive:** the adversary waits for the publication of the anonymized graph, then uses their auxiliary knowledge and some data science techniques to obtain the most achievable knowledge about the individuals in the graph and their relations.
- **Active:** the adversary doesn't wait for the publication. But instead, they take their first step even before the data owner collects the data from the community to create the graph. They form one or many fake accounts with specific characters connected to each other and, most importantly, to the possible attack victims. When the graph is published, these fake accounts are represented by nodes called Sybil nodes which, connected to each other, form one or many Sybil subgraphs. The adversary tries to recognize these subgraphs based on their characters, leading to easy identification of the victim nodes.

passive and active.

Confronting these attacks also takes two possible approaches. Let's say we are publishing the call records in a community for two years daily, starting from the first day of 2021.

- **Offline:** this approach means that we already have all the set graphs before releasing the first anonymized graph. In this approach, we cannot release the first graph before getting the last graph of the last day of 2022. It means we cannot benefit from these data before two whole years, which may decrease the value of the published graph. Also, we could not apply the offline solution on an indefinite stream of graphs.
- **Online:** the previous two cases show the importance of having an online solution where it's possible to publish every morning the private graph of the past day without waiting for any future graph.

The last characteristic that we will mention about anonymization algorithms is centered around the nature of their results. Two approaches could be listed:

- **Non-interactive:** The output of some of these algorithms is an anonymized version of the dataset. In this case, the data owner released the result and could not control any query or analysis done on the published output; for this reason, it's called the non-interactive approach.

- **Interactive:** The interactive approach doesn't release a dataset but a query or a mechanism result. The data owner receives the query, applies it to the dataset, performs the anonymization process on the result, verifies that the anonymized output doesn't breach the privacy of any individual, and finally, shares or releases the modified result.

These aspects of the nature of the graphs, the attacks and the anonymization processes will be discussed in the Chapter 3, and we will propose three anonymization mechanisms covering these aspects.

1.3.3 VIP nodes in the graph

We can notice an essential characteristic of social network users that is not well studied in the literature. It's the fact that the accounts have a different level of importance, and there is no need to provide the same level of privacy for all of them. Thus, we proposed to divide the nodes into two groups: VIP and Standard. The nodes could be categorized based on how many problems a privacy breach of their data might cause. These problems could be in the form of lawsuits by the users, fines by the data privacy commission of a country, or a public relations crisis.

The concept of the most important nodes in the graph is well known and widely discussed in the literature [85, 63, 164], then we are not assuming that we are presenting a new concept. But, the nodes are categorized as most important based on some graph centralities as degrees, betweenness, closeness, farness, eigenvector. Usually, a VIP account will be categorized as important based on one of these characteristics, so there is no need for the data owner to intervene and label an account as VIP to get higher protection, but it's not the case in many scenarios. For example, many influencers or celebrities could have two accounts, one public and one private, for friends and family. Both accounts should have the same level of protection because breaching the privacy of the personal account could reveal the same personal data as the public one. Other type of accounts that could be labeled as VIP are those dedicated to help and get in touch with victims of domestic violence or any kind of human slavery, the people suffering from suicidal thoughts, depression, or bullying. This type of accounts might not be important on the scale of the mentioned centralities, but breaching and sharing the connections of these accounts may lead to tragic events for these vulnerable individuals and a vast legal and public relations crisis. For this reason, we urge our company to add a new type of VIP node based on the nature of the account instead of just the automatically computed centralities.

This concept of the most important nodes is often used to measure the utility of a privacy-preserving algorithm by comparing the original and the anonymized graph based on these

nodes. But in our work, we suggest benefiting from the concept to decide how much protection each node should have.

Before representing our privacy-preserving mechanism [5] for these types of nodes, we will introduce the Blowfish Privacy and how to apply it on graphs generally. Then, we will benefit from this privacy definition to execute our technique of higher privacy for privileged nodes.

After explaining Blowfish Privacy, we present our interactive service applied on static graphs that provide a high level of protection for the VIP nodes and a high level of utility for the Standard nodes. In this way, we aim to reach a good trade-off between the privacy of VIP nodes and the utility of Standard nodes. As an interactive service, the mechanism receives queries to be executed on the graph. The original result is summed with noise computed based on the Blowfish Privacy process to get the noisy result shared with the inquirer.

1.3.4 Protecting subgraphs in dynamic graphs

Usually, the privacy-preserving techniques are dedicated to protecting the nodes themselves or their edges. The majority of the mechanisms protecting the nodes provide a high level of privacy. For this reason, they suffer from a low level of utility in their results. On another side, the mechanisms giving edge privacy have great utility but are vulnerable against numerous types of attacks that we will detail later.

Therefore, our two following mechanisms are proposed to deal with two scenarios where we prove the edge privacy vulnerability and the node privacy's low utility. Then we propose our new solutions to deal with these scenarios.

Our second proposed anonymization mechanism [3] for graphs is applied to dynamic graphs and adopts the non-interactive approach. But, instead of being dedicated to nodes or edges, this mechanism protects particular subgraphs.

Applying the DP mechanism on graph datasets is done by adding noise to the edges, as in [22, 68, 69, 77, 131, 142, 156] or to the nodes [35, 45, 78] to prevent identity and link disclosure while keeping the dataset suitable for analysis. However, Kifer et al. [81] have proved that an adversary with background knowledge can still disclose sensitive information from the data that induces correlations across tuples even if DP is applied.

The limitation of DP can appear in a user-centric design that relies on sharing or publishing dynamic graphs, since anonymizing each release independently will not be sufficient, because an attacker can still infer information about the individuals by combining the anonymized released graphs together. Thus, our goal in this part is to show how a DP mechanism can be extended to address the dynamic graph anonymization.

Figure 3: Graphs at disjoint time frame

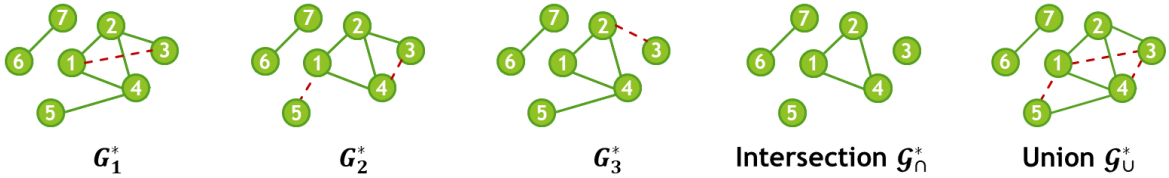


Figure 4: Graphs anonymized by an edge-DP mechanism.

To illustrate this scenario in an example, let us consider recording the communications inside a university community for some statistics and social studies. Let us say that the graphs in Figure 3 are the records of three days: G_1 for the first day, G_2 for the second, and G_3 for the third. However, by sharing these graphs as they are, a real threat of privacy could arise even if no identifiers are associated with the vertices. For example, if the shared graphs fall into the hands of an adversary possessing background knowledge, such as Doctor Bob and his two students Alice and John, call each other every day. By intersecting the three graphs, G_1, G_2, and G_3 as in Figure 3 and by projecting the Background Knowledge to this intersection, it becomes easy to know that vertices 1, 2, and 4 represent Doctor Bob and his two students.

Vertex	Age	Gender	Marital Status	Political View
1	30-40	Male	Divorced	Democratic
2	10-20	Male	Single	Republican
4	20-30	Female	Married	Republican

Table 1: Sensitive data associated to the nodes.

If the sensitive data associated with these nodes is as mentioned in Table 1, thus, it is effortless to guess that the vertices 1, 2, and 4 represent Bob, John, and Alice, respectively. In this way, the adversary can discover the three individuals' marital status and political views, which presents a serious privacy breach. The situation does not improve in Figure 4 despite applying a DP algorithm to add edges in each graph. These noisy edges cannot change the fact that just one triangle appears in the Intersection graph. Suppose we even use a privacy-preserving technique to protect the users' identity in the three graphs, the adversary still can connect the individuals with nodes 1, 2, and 4.

We intend to solve a paradox since we need the data to still carry out valid information about the population represented by the graphs without exposing individuals' privacy. We can define the utility of the released data as one or more statistical measures that the data user can compute with a certain degree of confidence. We propose here a new anonymization technique for a dataset composed of sequential dependent released graphs based on a relaxed version of Blowfish Privacy [71] under the non-interactive approach. Our goal is to propose a mechanism that protects the subgraphs that have a specific characteristic (like high occurrences in the released graphs, for example) by manipulating their existence or absence in each graph.

1.3.5 Node-Detention Differential Privacy

The third anonymization mechanism for graphs is specialized in confronting active attacks on dynamic graphs [4]. It outputs an anonymized dynamic graph under the online approach. A

fundamental challenge of an online solution is the active attack on sequentially released graphs. The adversary plants several Sybil nodes in the graphs, tries to identify them in the published graphs, then exploits them to identify other nodes called victim nodes. The massive advantage for the adversary in an online solution is that the data owner could not predict the new nodes, honest and Sybil, that the adversaries and the genuine users will add to the graphs in the future.

We already know all the nodes in the dataset in an offline solution before releasing the first graph. Then we can fake the existence of that node in any of the graphs even if its actual appearance is in a future graph. While in the online solution, we can only know the nodes of the current and previous graphs. Let's say that in the first graph, we had 1000 nodes; none of them is Sybil. In the second graph, 10 nodes were added, including two Sybil nodes. The adversary has to recognize the two Sybil nodes from the ten new nodes (having ids that don't appear in the first graph) instead of 1010 nodes. To make it easier to identify the Sybil ones, the adversary gives them unique characteristics, like a very high degree, by connecting via his fake account to many people in that community or a lesser degree, or any other features that are not very common in that graph. After creating one or many Sybil subgraphs using fake accounts and retrieving them, the adversary uses them to connect to some individuals and acknowledge their representative nodes in the graphs.

Let's say the adversary has successfully identified four individuals in the graph. Thus, he has the potential to reveal the relationship between these four. Who is calling who, how often? And several other private information about the relation between these victims. The data owners must protect these individuals before sharing or publishing their data. This paper proposes an online solution under Differential Privacy to safeguard individuals from active re-identification attacks. We prove that our solution provides a high level of protection against these attacks simultaneously with high utility of the published data.

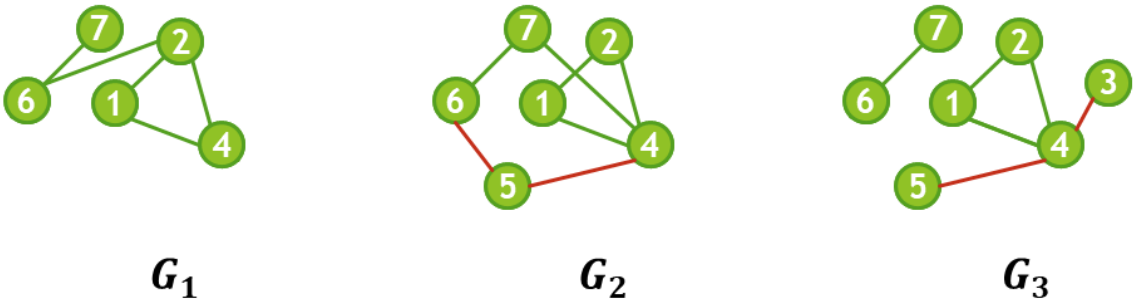


Figure 5: Active Attack on sequentially released graphs.

We can see in Figure 5 a simple example of an active attack happening during three days where a graph presenting the connections between a number of individuals is published each day after removing the identifying attributes . An adversary injects a Sybil node (Node 5) in G_2 and Node 3 in G_3. They represent the only new individuals in these two days; then, the adversary had no issue locating them in the published graphs. The adversary is targeting a couple, Alice and Bob. On Day 2, they get connected with both of them using Node 5. Now, it's known that nodes 4 and 6 represent the couple in the graph. On Day 3, Bob is targeted by both

Sybil nodes. The published graph shows that Node 4 is related to Bob, implying that Node 6 represents Alice. The adversary has found out that the couple didn't call each other for three days which might be a severe breach of their privacy.

1.3.6 Challenges and contributions of graph anonymization

In this way, we have covered by our three mechanisms the different characteristics of the anonymization graphs. We have proposed three different new solutions to help our company benefit from the generated graphs by a high utility of the graphs and the minimum possible graph. We can recap the challenges in this use case as follow:

- In a real-world scenario, the nodes of social networks have different levels of importance and need distinct levels of privacy. In contrast, Differential Privacy mechanisms give all the nodes the same privacy scale.
- Edge-Differential Privacy is unable to protect subgraphs in dynamic graphs.
- Edge-Differential Privacy is unable to protect the nodes' identity against active attacks on dynamic graphs.
- Node-Differential Privacy can resolve the past two problems but has a shallow utility.

Based on these challenges, we can list our contributions in this use-case by dividing them into three parts:

- First part:
  - Introducing the Blowfish Privacy technique into graph dataset anonymization.
  - Proposing the concept of two node privileges in the same graph.
  - Presenting the mathematical way of obtaining the Blowfish noise for several graph queries.
- Second part:
  - Presenting a passive attack on dynamic graphs that targets the subgraphs instead of the nodes or the edges.
  - Proving that the existing DP algorithms could not defy this type of attack.
  - Proposing a flipping mechanism to confront this attack on subgraphs and proving it complies to the BP requirements.
- Third part:
  - Suggesting an active attack on dynamic graphs targeting the nodes of the graphs, especially the new nodes of each instance.
  - Proving the high vulnerability of edge-DP in this scenario and the low utility of the node-DP results.

- Proposing a new DP technique called Node-Detention Differential Privacy (NNDP) to deal with active attacks on dynamic graphs.
- Proving the reliability of our new technique in this type of scenario.

1.4 Exposing Safe Correlations in Transactional Datasets

After expanding our company's business model by opening an online retail business, a large transactional dataset is being formed and growing with each sale operation. Analyzing this data will help us better understand our consumers and what they tend to buy based on their gender, age, nationality, level of education, history of purchases, and much other information, especially when linking the data generated from their social media profiles with their purchases.

But the same kind of analysis is not limited to transactional datasets but many other types of microdata. For example, if we expand our business to a video streaming service, watching, rating, liking, or adding a movie or a series to the watchlist will generate a row in the dataset that we should study. Other examples generating the same type of data could be online Taxi service, car rental, ticket or hotel booking, online marketplace for lodging, etc.

For the same reason why removing the identifiers from a node in the graphs is not enough, also removing them from the row of transactional data is not enough before sharing with a third party. Therefore, data providers should anonymize not only the identifying values but also the associations that link individuals to their sensitive values as an adversary may be able to combine their background knowledge [143, 151] to information in the released dataset to breach privacy. Hiding the associations between individuals and their sensitive values requires particular attention and cannot be done fairly straightforwardly.

Several anonymization techniques have been proposed to hide these sensitive associations. Some are differentially private [53] while others rely on generalization [143, 151, 108, 95, 15, 66] and/or bucketization [162, 40, 21, 99], separating what is sensitive from non-sensitive. (Generalization techniques can also satisfy differential privacy [97, 148, 46].)

These techniques have been shown to be effective and useful, but complications do arise when anonymizing transactional datasets [9, 10]. The above methods largely ignore these problems, which assume each record corresponds to a single individual. Datasets in which several tuples relate to the same individual may expose significant correlations between identifying and sensitive values. An adversary can use their knowledge of such correlations [80, 159] to breach privacy.

A particularly obvious example is datasets including location as part of the transaction. In these datasets, correlations provide foreground knowledge and could be used by the adversary to breach individuals' privacy. To better illustrate the problem, consider the car rental scenario given in Figure 6a where any vehicle can occasionally be rented from a location and returned to another.

In this example, only[1] the associations between *User ID* and *Location* in the Rental table

[1] We assume *User ID* and *Vin Number* are independent and identically distributed (i.i.d.).

are sensitive and should therefore be anonymized. One way of anonymizing the dataset is to use Anatomy [162], a bucketization technique [99, 162, 40, 93] that preserves privacy by dividing the dataset into sensitive and non-sensitive tables and keeping the values intact without any alteration to maximize the utility. An anatomized version of table Rental with attributes separated into *Rental_QIT* and *Rental_SNT* is shown in Figure 6b. Tuples in these tables are divided into groups in such a way that satisfies the privacy constraint requirements.

Anatomy uses ℓ-diversity as its underlying privacy constraint and thus protects the dataset against attribute disclosure. The anonymized tables in Figure 6b satisfy the 3-diversity privacy constraints[110]: each record must have at least three potential sensitive values.

However, given the transactional nature of the dataset, we recognize two types of correlations [9]:

- **Inter-group dependencies:** occur when an adversary knows certain facts about the individual (e.g., Roan_U1 frequently rents a car from a specific location).

- **Intra-group dependencies:** correlations between values in the same QI-group, which occur when there are multiple transactions for a single individual within a group; (e.g., if all transactions in a group were for the same individual, which results in an inherent violation of ℓ-diversity.) By considering this separately for transactional data, rather than simply looking at all tuples for an individual as a single "data instance", we gain some flexibility.

Dealing with correlations is a severe hurdle; losing them reduces the outsourced data utility while keeping them poses a threat to privacy. The authors in [9, 10] propose the safe grouping technique to ensure that each individual's tuples are grouped in one and only one QI-group that is at the same time ℓ-diverse, respects a minimum diversity for identifying attribute values, and all individuals in the same QI-group have an equal number of tuples. The approach is based on knowing (or learning) the correlations and forming buckets with a common antecedent to the correlation. This protects against inter-group dependencies. Identifiers are then suppressed where necessary (in an outsourcing model, this corresponds to encrypting just the portion of the tuple in the identifier table) to ensure the privacy constraint is met, including protection against intra-group correlation.

Figure 6c shows two QI-groups that respect safe grouping for a number of individuals $k = 2$ where we assume that there are no other QI-groups containing users $U1$ and $U4$. Figure 6c also shows that four identifying values in the QI-group are anonymized[2] to guarantee that individuals $U2$, $U3$, and $U5$ have an equal number of tuples.

For the sake of privacy, safe grouping inhibits the ability to learn correlations from the dataset and, thus, decreasing the utility of the dataset for aggregate analysis and frequent pattern mining.

[2]Some data is anonymized/suppressed in order to meet the constraint; this is in keeping with privacy models that uses partial suppression by replacing individual's values with a * to preserve privacy as in [151, 108] or encryption as in the model in [123] where some data is left encrypted, and only "safe" data is revealed.

User ID	Vin Number	Location
Roan_U1	0061d4a8248*	20.09;45.11
Bob_U5	05d7f4f9496*	19.10;38.13
Roan_U1	0036153c476*	20.09;45.11
Roan_U1	05d7f4f9496*	20.09;45.11
Lisa_U4	0e352814d34*	19.60;35.40
Lisa_U4	000cf44c9b3*	19.47;43.71
Lisa_U4	000cf44c9b3*	19.60;35.40
Elyse_U2	0038da44c64*	19.72;33.96
Bob_U5	0036153c476*	19.29;36.15
Carl_U3	0038da44c64*	19.72;33.96
Carl_U3	0061d4a8248*	19.57;44.70
Carl_U3	0e352814d34*	19.57;44.70
Bob_U5	0038da44c64*	19.72;33.96

(a) Original Rental table

User ID	Vin Number	GID
Roan_U1	0061d4a8248*	1
Elyse_U2	0038da44c64*	1
Bob_U5	0036153c476*	1
Carl_U3	0e352814d34*	1
Roan_U1	0036153c476*	2
Lisa_U4	0e352814d34*	2
Carl_U3	0038da44c64*	2
Roan_U1	05d7f4f9496*	3
Lisa_U4	0e352814d34*	3
Lisa_U4	000cf44c9b3*	3
Bob_U5	05d7f4f9496*	4
Bob_U5	0038da44c64*	4
Carl_U3	0061d4a8248*	4

GID	Location
1	20.09;45.11
1	19.72;33.96
1	19.29;36.15
1	19.57;44.70
2	20.09;45.11
2	19.60;35.40
2	19.72;33.96
3	20.09;45.11
3	19.60;35.40
3	19.47;43.71
4	19.10;38.13
4	19.72;33.96
4	19.57;44.70

(b) Anonymization using *Anatomy*; Tables *Rental_QIT* and *Rental_SNT*

User ID	Vin Number	GID
Lisa_U4	0e352814d34*	1
Lisa_U4	000cf44c9b3*	1
Lisa_U4	000cf44c9b3*	1
Roan_U1	0061d4a8248*	1
Roan_U1	0036153c476*	1
Roan_U1	05d7f4f9496*	1
Bob_U5	05d7f4f9496*	2
*	0036153c476*	2
*	0038da44c64*	2
*	0038da44c64*	2
*	0061d4a8248*	2
Carl_U3	0e352814d34*	2
Elyse_U2	0038da44c64*	2

GID	Location
1	19.60;35.40
1	19.47;43.71
1	19.60;35.40
1	20.09;45.11
1	20.09;45.11
1	20.09;45.11
2	19.10;38.13
2	19.29;36.15
2	19.72;33.96
2	19.72;33.96
2	19.57;44.70
2	19.57;44.70
2	19.72;33.96

(c) Anonymization using *Safe Grouping*

Figure 6: Table Rental anonymized

In this use case, we ensure that a dataset can be anonymized at the same level of privacy, as achieved by safe grouping, but with better utility. We show that a suitable trade-off can be made for which the privacy constraint is met without losing/suppressing *safe* correlations. More specifically, we demonstrate that the grouping can be achieved in a way that allows identifying and sensitive values to correlate across several QI-groups, exposing their correlations and still be considered safe [2].

Again, it is not a random grouping of tuples, but instead, we ensure, while anonymizing the dataset, if k distinct identifying values are grouped together, these same identifying values remain grouped together across the anonymized dataset. In other words, we preserve a safe grouping (i.e., given two distinct ℓ-diverse QI-groups, their intersection must yield either k identifying values or none) with better utility, since the correlations between identifying and sensitive values are exposed. To achieve this, we follow a divide-and-conquer strategy, dissecting large QI-groups into smaller ones by finding at least k identifying values that correlate in the original dataset and spreading them across several QI-groups. These groups are merged when necessary to preserve ℓ-diversity.

1.4.1 Challenges and contributions of transactional data anonymization

This use case aims to propose a solution for data owners interested in publishing or sharing their anonymized transactional datasets. But, at the same time, they are concerned about the correlations in their dataset that could be used to breach the individuals' privacy, and still, they don't want to lose the utility that these correlations present. The challenges of this use case are:

- Adversary could benefit from correlated data in transactional dataset to expose the identities of individuals.

- Hiding the correlations restrains the ability of extracting useful information from the dataset.

Based on this contradiction, we can summarize our contributions as follows:

- An in-depth study of safe grouping showing its strength and weaknesses against two types of adversaries: the first knows about the correlations, and the second can learn the correlations from the anonymized dataset.
- A correlation anonymization privacy constraint to ensure that only safe correlations are exposed in an anonymized dataset.
- An elastic safe grouping algorithm to achieve correlation anonymization using a divide-and-conquer strategy keeping correlated identifying values together across QI-groups.

1.5 Differential Private Image Classification

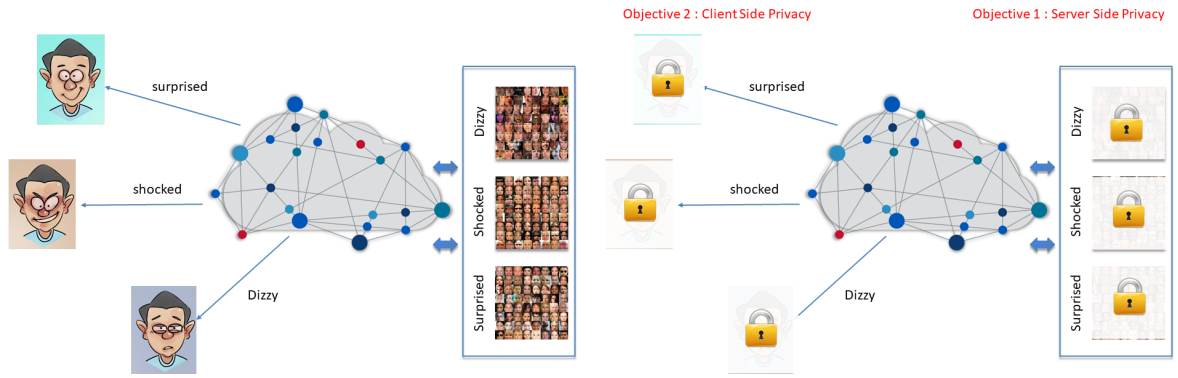
Many companies like Microsoft and Google provide free cloud storage services for individual users for limited data size and a paid service for a larger size. Other companies like Amazon are providing unlimited full-resolution photo storage for all prime customers. Posting images on social media is also a type of free storage where the user could post hundreds or thousands of photos for free. The company profits from these posts to know more about the users and their surroundings. Analyzing the images by a third party or sharing or even publishing the results of the analysis of images posted on public accounts would not form a real threat of private data. But, the problem appears when these photos are published on private accounts, stored on personal accounts on storage services, or when the images contain private information about individuals without their consent, like the example of the spectators' sentiments. The company must ensure that this process could not reveal personal data.

There are many scenarios of why images are stored on the cloud and treated. But all of them lead to the same research question: how do we benefit from the data without harming the privacy of individuals in the dataset?

Many cloud services that provide private image classification rely on encryption to ensure the security and privacy of the data. However, encryption does not help preserve privacy in many cases, like in an adversarial setting. For instance, while using partially homomorphic encryption to outsource K-Nearest Neighbors [161] classification, the authors show that distance learning attacks are possible [94]. These approaches are time-consuming as well due to the performance of encryption and decryption algorithms, and they are vulnerable to the theft of encryption/decryption keys [147].

In this use case, we propose a cloud-based DP image classification approach [1] that protects the privacy of individuals in a dataset of images. In fact, in our approach, we assume that the cloud computing service is semi-trusted. Thus, we should not be able to identify individuals in the dataset (i.e., images collected from the data owners) and individuals in the requests of inquirer (i.e., a typical user of the cloud service). Then, as we can see in Figure 7, our mission is to transform the cloud image classification into a protected process by ensuring the privacy of images on both sides, the pictures of the training data and those of the queries.

We use a Differential Private mechanism to convert images into noisy vectors, and at the same time, preserve their utility in a way that they remain useful for analysis. We study the



(a) Unprotected process (b) Protected process

Figure 7: Cloud face classification

trade-off between the accuracy and the privacy regarding the global privacy budget, which is the total allowed leakage as determined by the number of answered queries and the accuracy of the answers. In fact, we run several classification algorithms such as Support Vector Machines, Kernel Density Estimation (KDE), and K-NN to evaluate the privacy vs. accuracy trade-off. But first, we should define privacy and utility in our case. From the privacy side, we should provide a high rate of correct classifications of emotions on the faces. From the utility side, we present a scenario where an adversary was able to get, in a way or another, a number of these noisy vectors and has the needed technique to rebuild the images from the vectors, the adversary should not be able to recognize the individuals in the images.

1.5.1 Challenges and contributions of image classification anonymization

The challenges of this use case are:

- It's not safe to store vector converted from images on cloud. If an adversary managed to get the vector, they might be able to reconstruct the image and extract private information from it.
- Encryption approaches are time-consuming due to the performance of encryption and decryption algorithms, and they are vulnerable to the theft of encryption/decryption keys.

Therefore, our contributions in this use case is:

- We propose, implement and evaluate a private image classification framework based on a DP version of the PCA technique.
- We prove that reconstructing the images from the noisy version will not identify the individuals.

In this way, our company can benefit from the images posted on its social networks, stored in its free or low-cost storage services, or even provided safe image classification methods to other companies or organizations.

CHAPTER 2

Background: Differential and Blowfish Privacy

Before discussing our mechanisms, we introduce some basic notions of Differential Privacy (DP), some of their mechanisms, and what are the drawbacks of DP that motivated He et al. [71] to propose Blowfish Privacy (BP) as a generalization of DP and how BP solve these drawbacks.

2.1 Differential privacy

In many domains, getting statistical data about a dataset is necessary. However, this data can turn into a real threat to the privacy of any individual, participating or not, in this dataset. This breach of privacy is due to two factors; inferring auxiliary data about an individual and extrapolating statistical data about a community.

Dwork et al. [49] have found a solution by adding noise to the result in a way that preserves the utility of the result and protects the individuals' privacy. This technique is called Differential Privacy; it was proposed in 2006 to protect the individuals' rows when releasing statistical data about a dataset. The basic idea is to add organized random noise to the released data to protect all the individuals and ensure that the released data still have utility. Then it's a trade-off between privacy and utility. Dwork et al. [50] define DP as a privacy mechanism for a curator holding data of individuals in a database D, where each row represents the data of a single individual, to provide a sanitized database that allows statistical analysis and simultaneously protecting the individual rows. Formally speaking, a randomized mechanism M with domain $\mathbb{N}^{|\chi|}$ is (ϵ, δ)-Differential Private if for all $S \subset \text{Range}(\mathsf{M})$ and for all $D, D' \in \mathbb{N}^{|\chi|}$ such that $\|D - D'\|_1 \leq 1$:

$$Pr[\mathsf{M}(D) \in S] \leq e^{\epsilon}.Pr[\mathsf{M}(D') \in S] + \delta \tag{1}$$

where D and D' are two databases of records from a universe χ, which differ by just one row at most and e^{ϵ} is always greater or equal to 1 ($\epsilon \geq 0$). DP promises that the probability of harm for an individual to participate in a survey, for example, is not more significant than the probability of harm if she does not. By choosing $\delta = 0$, we ensure that, for every run, the randomized mechanism (M) returns the same results for D and D' with roughly the same probability. In other words, the probability that the output of $\mathsf{M}(D)$ is in the range S and the probability that $\mathsf{M}(D')$ is in the range S too are very close. This closeness is managed by a privacy factor ϵ.

This factor could have two types of jobs. First, when proposing a new privacy-preserving algorithm, to prove that it respects DP requirements, the authors could rely on the two probabilities mentioned in the formal definition by dividing them and trying to demonstrate the existence of an upper bound e^{ϵ}. If the upper bound is proved, then ϵ is defined, and the algorithm is DP.

The second job is when the algorithm is based on one of the basic DP algorithms; thus, it's already proved to be differentially private. In this case, this factor is a tool in the data owner's hand to decide the good balance of utility and privacy for its data. Many studies were done [73, 120, 82] to help the data owners chose the right ϵ.

(a) Interactive: Multiple Queries, adaptively chosen (b) Non-Interactive: Data are sanitized and released

Figure 8: The two approaches to release the DP and BP outputs: interactive and non-interactive.

Another parameter is required to generate the DP noisy result called Global Sensitivity ΔF or GS. The idea of DP is to add enough noise to cover the removal of any row in the dataset. $\Delta F = \max_{D' \in \mathcal{N}(D)} \mid M(D) - M(D') \mid$ where $\mathcal{N}(D)$ is the set of neighboring datasets (differ bu just one row) of D. The method that we use practically to calculate ΔF is:

1. Executing the query on the database to get the result r.

2. Executing the same query on this database but after excluding one row i to get the result r_i.

3. Calculating the absolute difference between r and r_i.

4. Considering the largest absolute difference as the sensitivity of the query.

The problems appear when a tiny minority or the rows have much higher sensitivity than the other. In this case, the noise will be much greater than needed for most of the tuples, affecting the utility of the result. For this reason, Local sensitivity [52] and Smooth sensitivity [126] were suggested to provide better utility. But, in a worst-case scenario, a mechanism relying on one of these two approaches will not be able to protect the tuples with the highest sensitivities. Therefore, this type of mechanism is not considered as DP.

DP and BP, like many other privacy-preserving algorithms, are divided into two approaches: interactive and non-interactive. Interactive approach [54, 51] is based on a query that can be used just once, can serve only one inquirer, and only for one task. Any mechanism that is based on this approach returns a noisy result to the user, as in Figure 8a. Each query has a budget $\epsilon = \sum_i \epsilon_i$ given by the database owner. Each execution i makes the budget loses ϵ_i of its value. The query cannot be executed anymore when the ϵ budget is less than ϵ_i. Hence, this approach has many restrictions on privacy preservation, especially if the budget is small where the number of queries could be insufficient. Besides, the data owner should validate the query. The non-interactive approach [54, 89], as in Figure 8b, returns a noisy synopsis data set. The inquirer can send queries to this synopsis to get noisy statistical data. This approach has no limits nor restrictions to the number and the sender of the queries.

2.1.1 Laplace mechanism

As we have explained, DP was proposed at first as an approach to add restrained noise on statistical queries. Laplace mechanism is the first and one of the most commonly used to implement DP. It relies on Laplace distribution to generate a controlled random noise. The Laplace distribution is:

$$Pr(x \mid b) = \frac{1}{2b} \times e^{-|x|/b}$$

where $b = \Delta F/\epsilon$.

To prove that the Laplace distribution could be applied as a DP mechanism, we will prove that it respects the formal definition (inequation) of DP. First, we should clarify that Laplace will be proved to be a mechanism of the strict version of DP where $\delta = 0$. Suppose that a query f is applied on a dataset D. M is an adding noise algorithm based on Laplace distribution where its output $M(D) = f(D) + x$ where $f(D)$ is the original output of the query and x is the noise generated by M. For a neighboring dataset D', with the same logic we get, $M(D') = f(D') + x'$. By dividing the two distributions, we seek to prove that e^{ϵ} is an upper bound of this division which complies to the definition of DP. Thus:

$$\begin{aligned}
\frac{Pr[M(D) = r]}{Pr[M(D') = r]} &= \frac{e^{-|x|/b}}{e^{-|x'|/b}} \\
&= \frac{e^{-|f(D)-r|/b}}{e^{-|f(D')-r|/b}} \\
&= e^{(|f(D')-r|-|f(D)-r|)/b} \\
&\leq e^{(|f(D')-r-f(D)+r|)/b}, \text{(following the triangle inequality)} \\
&= e^{(|f(D')-f(D)|)/b} \\
&\leq e^{(\Delta F)/b}, (\Delta F \text{ is the upper bound of } \mid f(D') - f(D) \mid) \\
&= e^{\epsilon}, (b = \Delta F/\epsilon)
\end{aligned}$$

$P(x \mid b)$ is a randomly generated number between 0 and $1/2b$. After getting the random value of the probability, we can compute the noise x as: $x = \pm \times ln(2b \times P(x \mid b))$. In this way, the Laplace mechanism generates a controlled random noise based on ϵ and ΔF.

Laplace distribution was first chosen because of its potential to generate a low noise with high probability. As we can see in Figure 9, to get a noise of ± 1, for example, the value of Laplace probability should be approximately 1.84, there is a 63.2% chance that this value Pr is $1.84 \leq Pr \leq 0.5$. implying a noise $x \leq \mid 1 \mid$.

For the Normal distribution, to have a noise $x < \mid 1 \mid$, the probability value should be $0.242 \leq Pr \leq 0.4$, the chance of having the value in this range is just 39.5%. This shows why Laplace distribution was the choice number one to apply DP on numerical datasets or statistical queries. It's because it could provide enough privacy based on the two mentioned parameters and a high level of utility by adding relatively low noise.

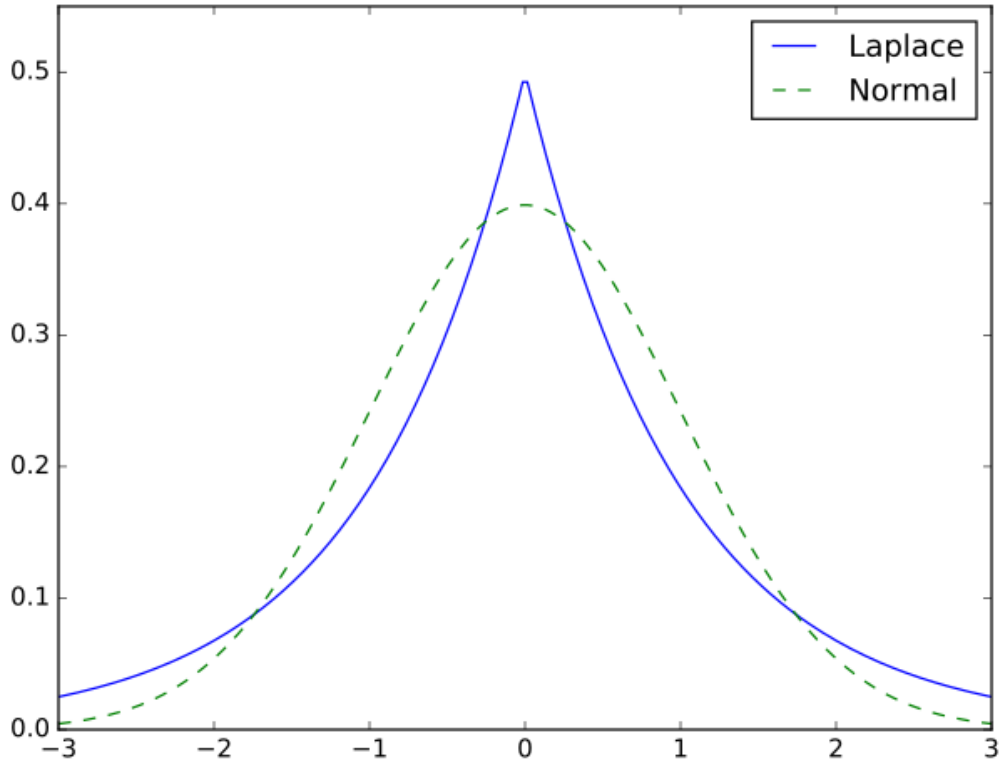


Figure 9: Comparing Normal and Laplace distributions

2.1.2 Exponential Mechanism

Laplace and Gaussian mechanisms were the first to be proposed for Differential Privacy. But both of them are applied to numerical data. The exponential mechanism was presented as a solution to use Differential Privacy on non-numerical data. Instead of adding noise on the numerical data, McSherry and Talwar [116] suggest applying Differential Privacy on non-numeric valued queries by adding noise on the possibility of choosing each category to be the result of the query or to be in the output of the mechanism.

Each category is paired with a quality score $q(D, r)$ representing how good this category could be as an output r for the query of the mechanism applied on the dataset D. The global sensitivity is the maximum change in the scoring function between two datasets that differ by just one row.

$$GS(q) = \Delta = max \mid q(D, r) - q(D', r) \mid$$

where D and D' differ at most by one row.

The probability of choosing r as the output of the mechanism is

$$P(exp(D, R, q, \epsilon) = r) = \frac{exp\left(\frac{\epsilon q(D,r)}{2\Delta}\right)}{\sum_{r' \in R} exp\left(\frac{\epsilon q(D,r')}{2\Delta}\right)}$$

where R is the set of all possible outputs and ϵ is the privacy parameter.

This probability returns the possibility for each value to be the output of the query. For a value r, we first compute an exponential number formed by ϵ multiplied by the quality of r and divided by the double global sensitivity. This number is used as the dividend in the probability. Then we compute similar numbers, each of them related to one of the possible outputs. All these numbers are summed, and the sum is used as the divisor in the probability. Therefore, the same divisor in the probability is used for two possible outputs, r_1 and r_2; nevertheless, the dividend varies from the exponential number related to the quality of r_1 to the one related to the quality of r_2.

In this way, the probability result represents the possibility of a value r to be the output based on its quality compared to all the qualities aggregated.

The DP formula is based on a privacy-preserving mechanism, two neighboring datasets differing by one row and a privacy parameter ϵ.

It's the same for the DP on graphs, but two neighboring datasets are two graphs that differ by one element. When this element is an edge, we call the approach edge-DP, and when it's a node, we call it node-DP [69].

Edge-DP ensures that the noise added to the graph is enough to protect an edge's existence or absence in the original graph. In comparison, node-DP ensures that the noise covers a node's existence or absence with its whole ego network.

To compute the global sensitivity GS, we must compute the maximum number of edges affected by adding or removing an element from the graph. For edge-DP, we have just one scenario; it's always two neighboring graphs that differ by any edge. Then $GS_{edge-DP} = 1$. For node-DP, the worst-case scenario is when the concerned node is related to all other nodes. In this case, the number of edges affected is equal to the number of all other nodes: $GS_{node-DP} =\mid V \mid -1$, which could be equal to thousands or even millions.

Higher GS leads to a higher injected noise to cover the worst-case scenario implying more privacy but less utility. Then node-DP surpasses edge-Dp on the privacy side but has severe drawbacks on the utility side. This paper proposes an approach to get the same privacy level of node-DP and, at the same time, much better utility.

2.1.3 Differential Privacy on Graphs

The DP formula is based on a privacy-preserving mechanism, two neighboring datasets differing by one row and a privacy parameter ϵ.

It's the same for the DP on graphs, but two neighboring datasets are two graphs that differ by one element. When this element is an edge, we call the approach edge-DP, and when it's a node, we call it node-DP [69].

Edge-DP ensures that the noise added to the graph is enough to protect an edge's existence or absence in the original graph. In comparison, node-DP ensures that the noise covers a node's existence or absence with its whole ego network.

To compute the global sensitivity GS, we must compute the maximum number of edges affected by adding or removing an element from the graph. For edge-DP, we have just one

scenario; it's always two neighboring graphs that differ by any edge. Then $GS_{edge-DP} = 1$. For node-DP, the worst-case scenario is when the concerned node is related to all other nodes. In this case, the number of edges affected is equal to the number of all other nodes: $GS_{node-DP} =\mid V \mid -1$, which could be equal to thousands or even millions.

Higher GS leads to a higher injected noise to cover the worst-case scenario implying more privacy but less utility. Then node-DP surpasses edge-Dp on the privacy side but has severe drawbacks on the utility side. This paper proposes an approach to get the same privacy level of node-DP and, at the same time, much better utility.

2.2 Blowfish privacy

A crucial weakness of DP is when dealing with correlated data [81]. Going back to the example in Figure 4 will clarify how DP has a real problem when dealing with this type of data where instead of having one graph, we have multiple graphs with the same nodes. The same problem appears when having multiple tuples related to each other by the same id, for example.

Another limitation of DP is that it aims to protect all the rows in the graph, which, in many cases, requires a lot of noises added to the graph, which might affect the utility of the released version. This issue was the motivation to propose another privacy definition called Blowfish Privacy (BP), considered a generalization of DP and has the same inequation but differs in defining the neighbor datasets. While in DP, two datasets are neighbors if they differ by just one row or tuple, BP proposes privacy policies to determine if two graphs are neighbors.

Every policy defines the secrets and the constraints of the graph. If the addition or removal of any tuple in the dataset is considered a secret and no constraints are listed in the policy, this is DP. For this reason, BP is considered as a generalization of DP. It relies on the same inequation but has a privacy policy P as an extension of DP. $P = (\mathcal{T}, G_S, \mathcal{I}_Q)$, where $\mathcal{T}$ denotes the domain of all possible tuples, $G_S = (V_S, e_S)$ is a discriminative secret graph with $V_S \subseteq \mathcal{T}$ and $\mathcal{I}_Q$ denotes the set of databases that are possible under the constraints Q. Let us say that in Figure 4, we have released noisy graphs that appear as if some professors or students are communicating from inside the university during some holidays where working is strictly forbidden, for tradition or religious reasons, or during a strike of the University Teachers Union. Then, releasing these graphs (showing calls between these phones in one of these days) may pose a problem for employees, students, or the university. Therefore, any CDR (Call Detail Records) graph that respects the privacy policy $P = (\mathcal{T}, G_S, \mathcal{I}_Q)$ should not show any call between phones on these specific days.

While in DP, the discriminative secret graph G_S is always a complete graph, BP may allow the public to distinguish between specific tuples. Besides, DP does not have any constraint Q. Since BP is a strong privacy definition, relaxation is needed in many cases to prove that a mechanism is relaxed BP. The data owner provides the upper bound of this relaxation. Before releasing the result, we must ensure that the noise is enough that the relaxation does not surpass the upper bound. The data owners also provide any background knowledge that they possess about their database. The service takes this knowledge as a constraint and checks if the noisy

graphs respect them. We will present a detailed explanation of BP while proposing how to apply it in the graph privacy use case.

CHAPTER 3

Differential and Blowfish Privacy for social networks

3.1 Introduction

Nowadays, data sharing is growing in popularity. It is shaping innovative business models and creating new marketplaces. Ad-based business models as Social network companies (Facebook, Twitter, Instagram, etc.) need user-centric designs to grow. Even companies that do not rely totally on an Ad-based model may also benefit from their users' data to make extra money like telecommunications corporations, video-on-demand service providers, etc. Then, a user-centric design benefits from its users and their data to make profits. In addition to that, communication patterns analysis is becoming crucial for global health security especially with the spread of epidemics such as COVID-19 by the means of social contact.

While it is bringing benefits to data providers, at the same time, it is putting a significant risk on the privacy of the users which is considered as an essential human right. This risk appears because the shared data contains sensitive information and needs to be appropriately protected. Several privacy breach scenarios have been widely cited, notably, the leaks of the medical records of the governor of Massachusetts [150, 109] and the AOL search data in 2006 [18], which are typical examples of privacy breaches caused by inappropriate protection of data.

This is the reason, privacy-preserving frameworks were proposed to enable communication graph analysis within formal privacy guarantees. Several techniques were proposed in the literature to protect various types of data. In k-anonymity [144], the values of the quasi-identifier attributes of the tuples are suppressed or generalized until each tuple is identical with at least $(k-1)$ other tuples on their quasi-identifier attributes. In l-diversity [111], a group of tuples is considered l-diverse if it contains at least l "well represented" values for the sensitive attribute. A table is l-diverse if every group is l-diverse. In t-closeness [96], the distribution of sensitive attributes in any group is close to its distribution in the full population. The distance between the distribution in a group and the population distribution should not exceed a distance of t. Differential Privacy (DP) [127] provides a mathematically provable guarantee that, whether or not, an individual's private information is included in the input of any DP algorithm, the output will lead to the same assumption about this individual's private information. In an attempt to fortify an individual's privacy, DP [50] has been proposed and has since garnered much attention among the privacy policymakers. DP is a privacy definition that aims to ensure a trade-off between privacy and utility by adding a small amount of noise enough to hide the adding or dropping of an individual from the database. DP provides ways for trading-off the privacy of individuals in a statistical database for the utility of data analysis. Current DP mechanisms have been applied on a variety of data structures such as images [1], location data [14, 72, 163], set-valued data [56, 34], relational data [86], and graph-based datasets [7, 100, 142, 157] (representing for instance social networks interactions and call detail records). In these graphs, vertices represent individuals, sometimes annotated with meta-information. Edges represent interactions among users and can be labeled as well.

The representation of the activities of individuals on social networks is crucial to benefit from the data provided by these networks in data analysis, social or health studies, etc. The

evolution of this representation makes the data more valuable. For this reason, we have seen progress in networks presentation forms from static graphs to dynamic graphs, then from offline to online dynamic graphs. This progress leads to more and more concerns about the privacy of individuals. At the same time, the attacks on graphs have also evolved from passive to active attacks. In the former, the adversary waits for the publication of the anonymized graph, then uses their auxiliary knowledge and some data science techniques to obtain the most achievable knowledge about the individuals in the graph and their relations. In the latter, the adversary doesn't wait for the publication. But instead, they take their first step even before the data owner collects the data from the community to create the graph. They form one or many fake accounts with specific characters connected to each other and, most importantly, connected to the possible victims of the attack. When the graph is published, these fake accounts are represented by nodes called Sybil nodes which, connected to each other, form one or many Sybil subgraphs. The adversary tries to recognize these subgraphs based on their characters which could lead to easy identification of the victim nodes.

3.2 RELATED WORK

To our knowledge, no previous work concerning Blowfish Privacy towards graph datasets has been considered in the literature. However, many differentially private mechanisms were proposed for graphs. In [69], Hay et al. divide these mechanisms into two types: edge-differential privacy and node-differential privacy.

Usually, a node in a graph represents a person while an edge represents a connection between two persons. The purpose of edge-differential privacy [22] is to prevent the usage of these connections for revealing the identity of a person. On the other hand, node-differential privacy achieve similar data protection by blurring node appearance in the graph. Node-differential privacy has much more sensitivity than edge-differential privacy, which is usually preferable.

In this section, we outline some techniques and algorithms dedicated to protect the graph datasets. The techniques could be divided into 4 categories: (1) identity and link disclosure [170, 47, 165], (2) dK-graph generation model [37, 156, 142], (3) platforms and programming languages [117, 132, 131], (4) static [125, 100] and dynamic graphs anonymization [165, 19, 20], (5) many types of k-anonymity approahces and finally, (6) anonymization techniques against active attacks.

3.2.1 Identity and link disclosure

Graph data disclosure can be divided into 3 categories [102]: 1) **Identity**: the identity of an individual associated with a node is revealed; 2) **Link**: the sensitive relationships between two individuals are disclosed; and 3) **Content**: the sensitive data associated with each node is compromised. We can list three privacy definitions to encounter identity disclosure:

- k-Candidate Anonymity [70, 165]: An anonymized graph satisfies k-candidate anonymity if, for a given structural query, no individual can be identified with a probability higher than $\frac{1}{k}$.
- k-Degree Anonymity [103]: An anonymized graph satisfies k-degree anonymity if every node in the graph has the same degree with at least $(k-1)$ other nodes.
- k-Neighborhood Anonymity [171]: A node is k-anonymous in a graph if there are at least $(k-1)$ other nodes such that the subgraphs constructed by the neighbors of each node are all isomorphic. A graph satisfies k-neighborhood anonymity if all the nodes are k-anonymous as defined above.

Other approaches can also be listed for the link disclosure:

- Link Reidentification [170]: Edges here are classified as either sensitive or observed. The goal is to minimize the probability of predicting sensitive edges based on the observed edges while keeping the number of observational edges removed small to preserve the utility.
- Privacy-Preserving Link Analysis [47]: This algorithm enables link analysis in dynamic graphs. Online computation of eigengaps (the difference between the largest and second-largest eigenvalues) with frequent updates. The algorithms address privacy concerns by applying encryption.
- Random Perturbation for Private Relationship Protection [165]: Two randomization techniques were studied. The first is based on adding then deleting edges randomly. The second one relies on switching two edges, e.g. we delete the two edges (v_1, v_2) and (v_3, v_4), then we add the two edges (v_1, v_3) and (v_2, v_4) where v_1, v_2, v_3 and v_4 are nodes in the graph.

The content disclosure is a significant problem, but, to the best of our knowledge, the literature does not consider the impact of graph structure on this category of disclosure.

3.2.2 dKgraph Generation model

Another type of DP mechanisms is based on the dK-graph generation model. In these mechanisms, various parameters are derived from the original graph. DP is applied on these parameters to create noisy versions, and finally, new dereived graphs are generated using a generation model. Chen et al. [37] present a method for publishing graphs under DP. They rely on a community-preserving generative model called CAGM. This model profits from some properties from the community as parameters to generate the graphs. Some differential private methods are applied to these properties to create the noisy parameters of CAGM. Another example is in [156, 142], where the authors propose an edge-DP graph generation mechanism relying on the dK-graph model. The mechanism generates a noisy graph based on a set of properties in the single static original graph. Thus, this mechanism cannot ensure the privacy of individuals in dynamic or multi-released graphs.

3.2.3 Anonymization of Dynamic graphs

Dynamic graphs are those subjected to changes in their structures or the weights of their edges. Less work follows the direction of dynamic graphs. The authors in [104] rely on DP for anonymizing dynamic social networks. Their approach, named DDPA (Dynamic Differential Privacy Algorithm), adds Laplacian noise to edge weights. DDPA tracks the edge weight information across the graph iterations and adds the privacy protection budget. However, it does not consider altering the graph topology. Very similar to our second scenario in this use case is [153], in this paper, social network graphs representing a time series of the corresponding social network's evolution are anonymized to a sequence of sanitized graphs released for further analysis. We share the same view that naively applying the existing approaches to each time-series graph will breach privacy purposes. However, our assumptions are more restricted since we assume that the attacker has external Background Knowledge about the graphs.

Li et al. [101] propose a privacy model k^m where k indicates the privacy level and m is a time period that an adversary can monitor a victim to collect the attack knowledge. A distributed algorithm is provided that adds nodes to the graph, then generates the noisy version. The distributed greedy merge noise node algorithm (DGMNNA) reduces the number of nodes added under satisfying the anonymous model. Qiuyang et al. [134] propose a dynamic algorithm that satisfies DP and protect social networks against attacks based on semantic information. They classify the original graph into several subgraphs according to some characteristics of nodes. The graph is represented as an adjacency matrix. Quad-tree is used to divide the dense area of each subgraph. DP noise is added to the tree's leaf nodes, and finally, the adjacency matrix is reconstructed and published.

Yue et al. [169] proposed local and global anonymity functions and a framework called APRI to apply sequential online anonymization on a set of graphs. They anonymize the degree of the node of the current graph locally, then they compare, via APRI using Kolmogorov-Smirnov Test, the distribution of node degrees of the current graph, and the set of previously anonymized graphs. When the difference is equal or greater than a given threshold, they restart the anonymization process for this graph. They use the global anonymity function to ensure the similarity in the distribution of node degrees between all the anonymized graphs.

Mcwan et al. [113] propose a clustering algorithm to group at least k nodes into k clusters based on their connectivity and anonymize each cluster for every instance of the graph. The algorithm supports the addition of nodes in new instances. Each cluster contains the nodes with close connectivity, and these nodes in the anonymized instance of the graph are assigned with the same label. Also, Yu et al. [168] propose a grouping mechanism for the nodes based on their properties. The mechanism guarantees that without a background knowledge, an attacker's probability of identifying a node involved in any edge is at most $\frac{1}{k}$. Also, the probability that an attacker identifies an edge between two nodes is at most $\frac{1}{k}$. Therefore, the goal of the mechanism is to protect edges against attacks without background knowledge dynamically.

All these mechanisms for sequentially released graphs focus on the properties of nodes, especially degrees of nodes. Thus, two of them [113, 169] might protect dynamic graphs against

a background knowledge. However, all of them do not aim to protect the dynamic graphs against the type of background knowledge containing information about the connections or the relation between two or more individuals. Returning to the example provided in the Introduction about the Doctor nad his two students, the background knowledge could be that Alice and John call Dr. Bob daily, especially in exams and project submissions. They do not usually communicate in summer, but we also know that Alice's birthday is on the 6th of August and Doctor Bob's birthday is on the 24th of August. We doubt that it is possible to prove that the mechanisms proposed to deal with the background knowledge of nodes could deal with an adversary having our type of background knowledge and trying to project it into the graphs by searching for three nodes communicating in the time of academic year especially in the periods of exams and projects. Then, a period of no communication in summer interrupted by calls on the 6th and 24th of August. Thus, our work is different from others by addressing the problem of facing this type of background knowledge and preserving an acceptable level of utility in the graph's released instances.

3.2.4 K-anonymity

In [103], Liu et al. proposed the k-degree anonymization for graphs by manipulating the degree of the nodes in such a way that the degree of any node in the graph is shared with at least $k-1$ other nodes. In their mechanism, they first anonymize the graph's degree sequence then build a k-degree anonymous graph based on the noisy degree sequence by adding edges to the original graph. Lu et al. [105] suggest a faster algorithm that combines the two phases of the previous algorithm in one stage by simultaneously applying the edge addition and the anonymization of the degree sequence.

The authors in [28, 29] assume that the previous algorithms are not efficient on large networks. They presented a polynomial-time algorithm that creates the k-degree graph by a minimum number of edge modifications. They apply a univariate micro-aggregation to anonymize the degree sequence. Instead of edge addition, Chester et al. [39] propose a node addition algorithm to reach k-degree anonymity.

Wu et al. in [160] propose a more general view than k-degree anonymization called k-symmetry anonymity to cover many properties of the nodes other than degree similarity. Ma et al. [107] propose the KDVEM algorithm to reach k-degree anonymity while providing high utility and minimum amount of distortion. The algorithm is formed of two phases, finding the best target degree of each node and deciding the best nodes candidates to add the edges to achieve the target.

In [140], the authors propose an algorithm to anonymize the degree sequence of the graph to maintain its coreness for a better utility. Maintaining the core number sequence assures the retaining of most of the graph information.

These works were all based on the degree of the nodes. In [171], Zhou et al. propose a 1-neighborhoods algorithm by grouping the nodes having similar neighborhoods and anonymizing them. The neighborhoods extraction is based on isomorphism tests. In [172], the

authors proposed k-automorphism to protect the published graphs against many structural attacks by applying an algorithm called KM. What's more interesting for us in their work is their suggested vertex Id generalization algorithm for k-automorphism applied on dynamically released graphs. This algorithm is applicable on the offline approach, then, we can't use it in our case where we are dealing with the online approach.

3.2.5 Dealing with active attack

To explain better the problem of active attack and how some related work in the literature dealt with it, we will use an example of a traveler aiming to enter a country in the era of the covid-19 pandemic. We list the scenarios proposed by each technique for the authorities to deal with this passenger.

3.2.5.1 Attacks

In [17], the authors listed several passive and active attacks on static graphs. They propose two active attacks, the walk-based, and the cut-based. Three main differences between these two approaches could be listed:

- Cut-based attack needs fewer Sybil nodes.
- Walk-based has a much more efficient algorithm to retrieve the Sybil subgraph.
- It's harder on the privacy-preserving mechanism to detect the walk-based attack.

Mauw et al. [115] define the robustness of the active attack stages, and they benefit from their optimized strategy to prove that the attack is robust and resilient to small graph perturbation.

The attack presented in [36] is the type of attack that we intend to encounter in this paper. The authors develop the attack shown in [17] and propose an attack on dynamic graphs relying on Sybil subgraphs. In the Subsection 3.5 .3, we present the active attack inspired by [17] with some modifications that we will encounter by our privacy-preserving approach.

3.2.5.2 SybilGuard

SybilGuard [167] is a protocol to defend against active attacks in a decentralized approach where each node decides on its own to accept a connection with another node or not without any interference from a trusted central authority. The idea is to divide the graph into two regions, the honest and the Sybil. The edge connecting a node from the Sybil region to an honest region is called an attack edge. The goal of the protocol is to create a random routing table for each node in a way that limits the possibility of accepting nodes from the Sybil region. SybilLimit [166] optimizes this guarantee to represents a near-optimal one. Nevertheless, both still have some drawbacks. The high false negativity of the SybilGuard and the unrealistic assumption of SybilLimit about the knowledge of the number of honest nodes in the network were the motivations for Daenzis and Mettal to propose the SybilInfer algorithm [43]. It relies on a

Bayesian inference approach that returns potential regions of dishonest nodes with a probability of certainty for each node.

In the Covid example, these techniques do a PCR test on the traveler, but he's allowed to enter the community even if the result is positive. The authorities warn the people to keep a distance from him because he's affected. But still, he resides in society and might form a threat.

These techniques guarantee that an honest node could be linked to most other honest nodes, and it will only accept a bounded number of Sybil nodes. The problem here is that no noise is added to the graph, then there is no difference between the original and published graph. So we could not prevent an adversary from locating its Sybil nodes, and the bounded number of Sybil nodes is enough to attack an honest node.

3.2.5.3 Edge-Differential Privacy

The vast majority of edge-DP mechanisms are dedicated to releases noisy graph properties as degree distribution [69, 142, 156], frequent graph patterns[146], counting queries for k-triangles and k-stars [77], or for subgraphs [136] and clustering coefficients [158]. An extension of DP called Blowfish Privacy [71] also has the ability, when applied on graphs, to provide noisy query results about these properties. For example, in [**?**], the authors provide methods to return anonymized query results under Blowfish Privacy applied on graphs where some nodes are more valuable than the others, implying that they should have, with their connections, a higher level of privacy than other nodes.

Some other works release an anonymized version of the graph. The authors in [125], and [119] proposed DP edge flipping algorithms based on a linear time algorithm for the first and a randomized response technique for the second. These two mechanisms, TmF and EdgeFlip, return an anonymized graph instead of a query result.

Many works propose to generate noisy degree distributions or similar noisy statistics from graphs under differential privacy. Generative methods are then used to create output graphs fulfilling noisy input distributions [45]. Qin et al. [133] propose LDPGen for decentralized social networks. LDPGen collects neighbor lists of the nodes and reconstructs the graph in two phases under local edge-differential privacy. Finally, Karma et al. [77] presents efficient algorithms to provide noisy answers tosub graph counting queries under a relaxed version of edge-differential privacy.

The advantage of edge-DP is that it doesn't need considerable noise to be achieved. Then, the results keep a high level of utility.

To explain the disadvantage of edge-DP, let's go back to our Covid example. Edge-DP allows an infected passenger from directly entering society. But it tries, for some level, to control his connections. But this won't prevent him from infecting other people (the taxi driver driving him from the airport, the delivery man,...).

Then, this approach cannot protect the individuals in the graph from an active attack. For example, four nodes are new (didn't appear in the first release) in the second release of a dynamic graph. By manipulating the existence and absence of the edges, Edge-DP will affect

the degrees of the nodes. Table 2 shows the original and the published degrees of the new nodes.

If the adversary has injected one node of degree 1000 before this release, their goal is to retrieve that node. It's evident that v_6 is the node injected by the adversary as its published degree is close to the original one while the three others are very far. Therefore, edge-DP cannot be trusted to face an active attack on a dynamic graph despite its capacity on the utility side.

Original Degree	Published Degree
$Deg(v_6) = 1000$	$Deg(v_6) = 950$
$Deg(v_7) = 120$	$Deg(v_7) = 112$
$Deg(v_8) = 85$	$Deg(v_8) = 92$
$Deg(v_9) = 97$	$Deg(v_9) = 101$

Table 2: Original and published degree of four new nodes under edge-DP.

3.2.5.4 Node-Differential Privacy

Node-Differential Privacy [69] presents a strictly stronger guarantee than Edge-Differential Privacy. The guarantee is to provide enough noise to protect the existence or non-existence of a node in two neighboring graphs that differ, in the worst-case scenario, by a node connected to all other nodes in the graph where it exists. This technique ensures a high level of privacy but, on the other hand, might severely damage the utility of the output.

Differential privacy has two approaches, the interactive and non-interactive [49, 51, 89]. All the proposed node-Differential Privacy techniques and algorithms that we have found in the literature adopt the interactive approach where their objectives are to return a noisy result of queries. In contrast, our technique adopts the non-interactive setting where the output is a noisy dataset instead of a query result. Therefore, we assume that our work will be the first to present a solution to provide node protection under the non-interactive approach.

The main disadvantage of the node-DP is in the accuracy of the result. Deleting and adding nodes with their entire ego network is very expensive in the term of utility.

For this reason, many of these algorithms are relied on graph projection technique to apply node-Differential Privacy. A parameter α is used to transform a graph to be α-degree-bounded. These graphs have a limit in the number of allowed connections for the nodes. In this way, it's guaranteed that adding or deleting a node has a limited effect on the utility. For example, the authors in [78] propose several techniques for interactive node-differential private techniques for degree-bounded graphs and present a methodology to analyze the accuracy of the results. The main idea in their work is to remove all the nodes with a higher degree than α, which causes a much higher number of edges to be removed than necessary.

Another way to tackle the problem of low utility was suggested by Blocki et al. in [23]. To compute how much noise is needed in each case, we have first to calculate the global sensitivity GS of the query. GS in node-Differential Privacy is the maximum difference in the query result between the original graph and any neighbor graphs that differ by just one node from the original one. The authors in [23] propose a Restricted Sensitivity instead of Global Sensitivity. They achieve that by reducing the set of neighbor graphs to just the ones having a distance

less than d from the original graph, where d is calculated based on several projection-based techniques.

Also, based on projection techniques, Day et al. [45] present two approaches to publish degree histograms and cumulative degree histograms under node-differential privacy. Chen et al. [35] proposed a node-Differential privacy mechanism supporting equijoins to answer subgraph counting queries.

Back to our Covid example, Node-DP forbids any person with positive PCR results from entering the country. While this provides a high level of protection for the citizens, it leads to a significant economic loss. This person could be a tourist, a businessman, or a skillful person that might form a high value for the business sector.

On the graph side, the technique gives us the possibility of deleting the new node with its ego network, adding a fake node and create its ego network, or manipulating the ego network of the real node. The advantage is that the adversary has a high uncertainty about the new node appearing in the published graph. Is it the real Sybil node or a fake node and the real one was deleted?

The disadvantages are:

- the need for a very high noise, which keeps us with very little utility,
- to the best of our knowledge, no node-DP mechanism was proposed, which adapts the non-interactive approach providing an anonymized version for the original graph. All the known mechanisms provide a noisy query result, mainly on the node degrees.

Therefore, we assume that our technique is the first to offer the same level of protection for the individuals and, at the same time, better utility and adapt the non-interactive approach.

SUBCHAPTER 3.3.3

Blowfish Privacy for VIP nodes

3.3 .1 Introduction

In this subchapter, we present a summary of Blowfish privacy and explore the possibility of applying it in the context of undirected communication graphs. Communication graphs represent social contact or call detail records databases. We define the notions of neighborhood, discriminative secrets, and policies for these graphs. We study several examples of queries and compute their sensitivity. Even though not addressed in the original blowfish privacy paper, we explore the idea of having a discriminative secret graph per individual. This allows us to treat some persons as VIP and put their privacy on top priority, where other persons can have lower privacy constraints. This may help to offer privacy as a service and increase the utility of the anonymized communication graph to an appropriate level. Differential privacy has a single tuning knob, namely ϵ, sometimes two (ϵ and δ). For example, increasing ϵ means more utility and less privacy. The idea of Blowfish privacy is to provide more tuning knobs by introducing policies [71].

In Blowfish privacy, a *policy* specifies:

- *secrets*: information that must be kept secret. Since not all the information has to be secret, we can increase the utility of the data by lessening the protection of certain properties.
- and *constraints*: known properties about the data. Constraints add protection against an adversary who knows these constraints.

Therefore, differential privacy can be considered as an instance of Blowfish privacy where:

- every property about an individual's record is protected,
- every individual is independent of all the other individuals in the dataset. There is no correlations.

Because of its generalized framework and powerful expressiveness of adversarial knowledge, we expect that DP and BP can solve privacy challenges in graph-based databases. In this part, we explore the application of these two privacy-preserving techniques to communication graphs such as social networks and call detail records databases. We model the

Table 3: Notation of DP and BP, secrets, and discriminative pairs

Symbol	Description
D	Database of n tuples
$\mathcal{T} = A_1 \times A_2 \times \ldots \times A_m$	Domain of m categorical attributes
$t \in \mathcal{T}$	A single tuple
$t.\_id$	Id of the tuple's real owner
$t.A_i$	Value of the ith attribute in tuple t
$\mathcal{I}_n$	Set of all possible datasets with size n ($\|D\| = n$)
$(D_1, D_2) \in N$	In DP, D_1 and D_2 are neighbors, they differ in the value of one tuple
$\mathcal{M}$	A randomized mechanism, for example adding random noise to the result of a query
$S \subseteq \text{range}(\mathcal{M})$	A set of the outputs generated by $\mathcal{M}$
ϵ-differential privacy	For every S and every two neighbors (D_1, D_2): $\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \times \Pr[\mathcal{M}(D_2) \in S]$
$f : \mathcal{I}_n \to \mathbb{R}^d$	A function that takes a database as input and returns a vector of real numbers as output, for example a `countIf` query
$S(f)$	The global sensitivity of f is the max Manhattan distance between the outputs for any two neighbor databases: $S(f) = \max_{(D_1,D_2)\in N} \|\|f(D_1) - f(D_2)\|\|_1$
$\mathcal{M}^{\text{Lap}}$	The Laplace Mechanism adds $\eta \in \mathbb{R}^d$ to $f(D)$, where η is a vector of independent random variables. Each η_i is drawn from the Laplace distribution with parameter $S(f)/\epsilon$: $\Pr[\eta_i = z] \propto e^{-z.\epsilon/S(f)}$
$\mathcal{P} = (P_1, \ldots, P_k)$	A partitioning of the domain $\mathcal{T}$
$h_{\mathcal{P}} : \mathcal{I}_n \to \mathbb{Z}^k$	A histogram query. $h_{\mathcal{P}}(D)$ outputs for each P_i the number of times values in P_i appear in D. The sensitivity of histogram queries is $S(h_{\mathcal{P}}) = 2$ since replacing a tuple by another one may decrease the count of a partition and increase the count of another partition.
$h_{\mathcal{T}}$	The complete histogram query, it outputs for each $t \in \mathcal{T}$ the number of times it appears in D
$\mathcal{E}_{\mathcal{M}}(D)$	The expected mean squared error of $\mathcal{M}$: $\mathcal{E}_{\mathcal{M}}(D) = \sum_i \mathbb{E}[(f_i(D) - \bar{f}_i(D))^2]$ where $f_i(D)$ and $\bar{f}_i(D)$ are the ith components of the true answer and the noisy answer, respectively. For Laplace mechanism and histogram queries, this error is: $\mathcal{E}_{\mathcal{M}^{\text{Lap}}_{h_{\mathcal{P}}}}(D) = \|\mathcal{T}\|.\mathbb{E}(\text{Laplace}(2/\epsilon))^2 = 8\|\mathcal{T}\|/\epsilon^2$. A large epsilon means less error, hence more utility.

secrets and the auxiliary knowledge in terms of the Blowfish privacy model and give numerous examples.

3.3 .2 Notation

The Blowfish privacy notation is based on the differential privacy notation as summarized in Table 3.

Table 4: Notions of secrets

Symbol	Description & Examples
s	An arbitrary statement over the values in the database. *Example1*: $t.\_id = \text{'Bob'} \wedge t.disease = \text{'cancer'}$. *Example2*: $t.\_id = \text{'Bob'} \wedge t_2.\_id = \text{'Alice'} \wedge t_1.disease = t_2.disease$
S	A set of secrets that the data owner would like to protect, e.g. {*Example1*, *Example2*}
$(s, s') \in S \times S$	A pair of secrets, e.g. (*Example1*, *Example2*)
A discriminative pair of secrets (s, s')	A mutually exclusive pair of secrets. Two statements that cannot be true at the same time. An adversary must not be able to distinguish which one is true and which one is false, e.g. ($t.\_id = \text{'Bob'} \wedge t = x$, $t.\_id = \text{'Bob'} \wedge t = y$)
s_x^i	The secret $t.\_id = i \wedge t = x$ where $x \in \mathcal{T}$, e.g. $s^{\text{'Bob'}}_{(\text{'cancer'},65)}$
$\mathcal{S}_{\text{pairs}}$	A set of discriminative pairs of secrets, e.g. $\mathcal{S}^{\text{full}}_{\text{pairs}}$, $\mathcal{S}^{\text{attr}}_{\text{pairs}}$, $\mathcal{S}^{\mathcal{P}}_{\text{pairs}}$, $\mathcal{S}^{d,\theta}_{\text{pairs}}$
$\mathcal{S}^{G}_{\text{pairs}}$	A set of discriminative pairs of secrets based on graph $G(V, E)$, i.e. $\{(s_x^i, s_y^i) \mid \forall i, \forall (x, y) \in E\}$
Full domain: $\mathcal{S}^{\text{full}}_{\text{pairs}}$	For every individual, the value is not known to be x or y, i.e. $\{(s_x^i, s_y^i) \mid \forall i, \forall (x, y) \in \mathcal{T} \times \mathcal{T}\}$
Attributes: $\mathcal{S}^{\text{attr}}_{\text{pairs}}$	For every individual and every two tuples differing in the value of only one attribute A where one of them is real, the real tuple is not known. The privacy definition is weaker than in full domain $\mathcal{S}^{\text{full}}_{\text{pairs}}$ since the real tuple is distinguishable if more than one attribute differs, i.e. $\{(s_x^i, s_y^i) \mid \forall i, \exists A, x[A] \neq y[A] \wedge x[\bar{A}] = y[\bar{A}]\}$
Partitioned: $\mathcal{S}^{\mathcal{P}}_{\text{pairs}}$	For every individual and every two tuples coming from the same partition where one of them is real, the real tuple is not known, i.e. $\{(s_x^i, s_y^i) \mid \forall i, \exists j, (x, y) \in P_j \times P_j\}$. This privacy definition is very useful for location data.
Distance threshold: $\mathcal{S}^{d,\theta}_{\text{pairs}}$	For every individual and every two tuples having their distance less than or equal to a threshold θ where one of them is real, the real tuple is not known, i.e. $\{(s_x^i, s_y^i) \mid \forall i, d(x, y) \leq \theta\}$

3.3 .3 Secrets

In addition, Blowfish defines secrets and discriminative pairs of secrets as shown in Table 4. We give examples of secrets and pairs of secrets over a communication graph in Table 5.

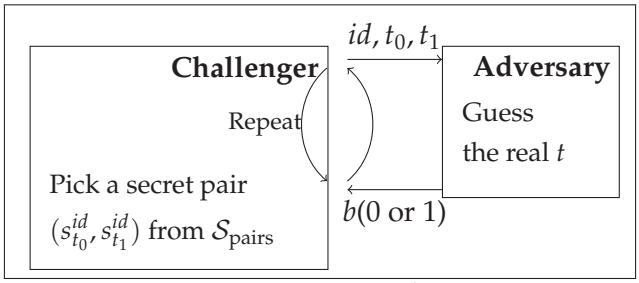


Figure 10: Discriminative pair of secrets as a game

The *discriminative secret graph* generalizes the specification of discriminative pairs of secrets. It is a graph where vertices represent secrets and edges link only the discriminative pairs of secrets. More formally it is denoted $G_S = (V_S, E_S)$ where $V_S = \mathcal{T}$ and $E_S \subseteq \mathcal{T} \times \mathcal{T}$. Even though not addressed in the original blowfish privacy paper, we explore the idea of having a

Table 5: Examples of notions of secrets for a communication database

Symbol	Description - Example
Secret: s	Bob has talked to Alice: $t_i.\_id = \text{'Bob'} \wedge t_j.\_id = \text{'Alice'} \wedge t_i[j] = t_j[i] = 1$
A discriminative pair of secrets (s, s')	Given two communication tuples (ego networks), we cannot distinguish which one of them belongs to Bob, for example, $(t.\_id = \text{'Bob'} \wedge t = (0,1,1,1),\ t.\_id = \text{'Bob'} \wedge t = (0,0,0,1))$
s_x^i	The secret where individual i has ego network x, for example, $s_{(0,1,1,1)}^{\text{'Bob'}}$
$\mathcal{S}_{\text{pairs}}^{\text{full}}$	For an individual, all ego networks are discriminative
$\mathcal{S}_{\text{pairs}}^{\text{attr}}$	For an individual and two vectors that differ in only one communication, we cannot tell which one is real.
$\mathcal{S}_{\text{pairs}}^{\mathcal{P}}$	For an individual and two tuples belonging to the same partition, we cannot tell which tuple is the real one.
$\mathcal{S}_{\text{pairs}}^{(d,\theta)}$	Given a distance metric and a threshold. The privacy game is to challenge the adversary with one individual and two records having their distance less than or equal to threshold. A suitable distance for communication graphs is the Hamming distance (or the number of different bits), which is equivalent to the Manhattan distance in this case.

discriminative secret graph per individual. This allows us to treat some persons as VIP and put their privacy on top priority, where other persons can have lower privacy constraints. This may help increase the utility of the communication graph to an appropriate level.

In this direction, the idea of a discriminative secret is very similar to what consists a game in cryptography. We prefer to call it a privacy game here and represent it as shown in Figure 10. In this game, a challenger picks an Id (e.g. Bob) and a pair of discriminative secrets at random (e.g. "Bob has called Alice" or "Bob has not called Alice"). The pair is represented by two tuples, or an edge in the discriminative secret graph of the Id. The edge vertices identify the two tuples. The challenger sends the Id and the two tuples to the adversary (e.g. which one does belong to Bob?). The adversary has to guess which of the two tuples belongs to the id and responds with only 1 bit b. $b = 0$ is chosen for t_0 and $b = 1$ for t_1.

Our goal is to make the probability of the adversary guessing the assumed right tuple not significantly different than a coin flip.

An important remark about undirected communication graphs is that not all the graphs are feasible. If Bob has talked to Alice, it means that Alice has talked to Bob. The database matrix is symmetric. Another constraint is that $t_i[i]$ must be 0, and all other entries are either 0 or 1. The Blowfish privacy framework allows to define constraints about the dataset, and redefines the notion of neighborhood databases by excluding intermediate, yet infeasible ones. Therefore, we suggest that Blowfish privacy is a more suitable framework for communication graphs than its differential privacy predecessor.

3.3 .4 Examples of policies

Examples of discriminative secret graphs for a communication graph of three nodes (for simplicity) and different policies are shown in Table 6. To explain more these policies we

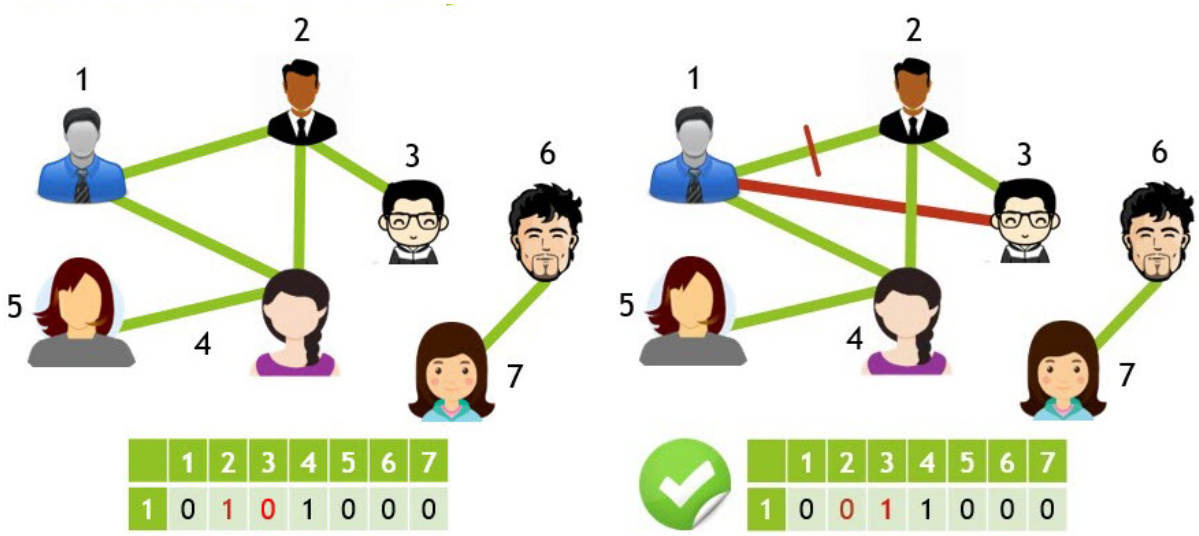


Figure 11: Example of full policy.

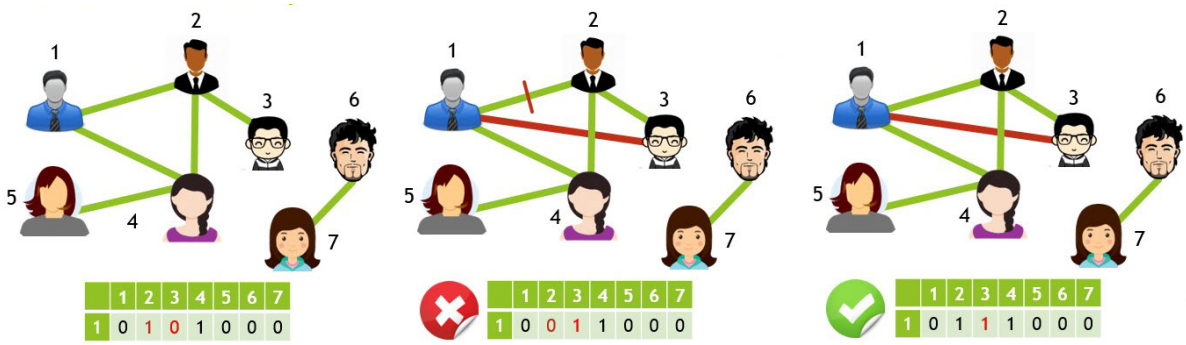


Figure 12: Example of attribute policy.

present examples of communications graphs neighboring for each policy:

- Full policy: a change in the ego network of any vertex is considered as a secret that should be protected regardless of how many edges has changed in this ego network. For example, in the graphs in Figure 11, the ego network of vertex 1 has changed, than this is a secret that should be protected regardless of the number of edges affected by this change.

- Attribute policy: The changes that occur in Figure 12 to the ego network between the first and the second graph is not considered as a secret under attribute policy, and therefore these two graphs are not neighbors, because the ego network of vertex 1 is changed by two edges, while, to be considered as a secret under attribute policy, an ego network should be changed by just one edge. Then the first and third graphs are neighbors.

- Distance policy: let's say we aim to protect the connectivity of the graph with a threshold 1. If we have a query asking about the number of components, we have to protect the edges that might change the result by 0 or by 1. For example, in the first two graphs of Figure 13, we have removed two edges which changed the number of components from 2 to 3. Then the adding or removing of these two edges together is a secret that should be protected, and these two graphs are neighbors. However, removing the two edges in third graph changes the number of components by 2, then, we are not interested in protecting

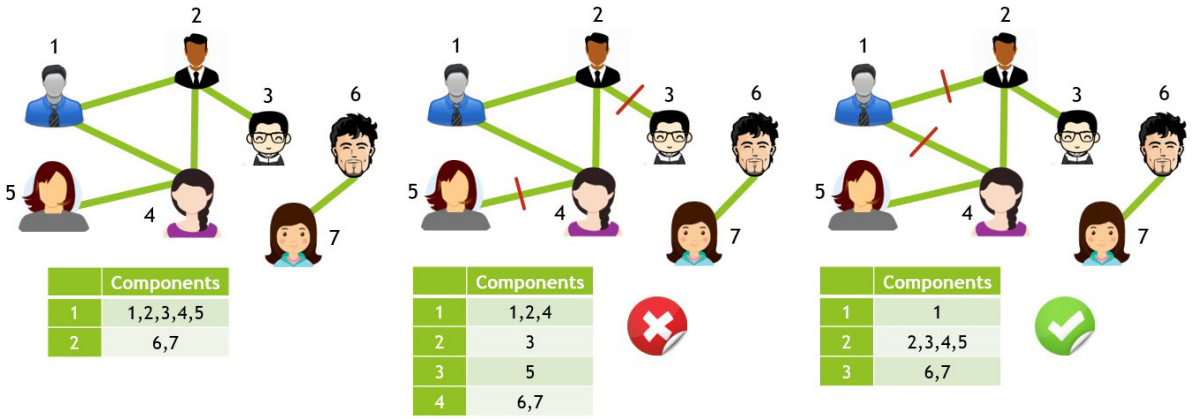


Figure 13: Example of distance policy.

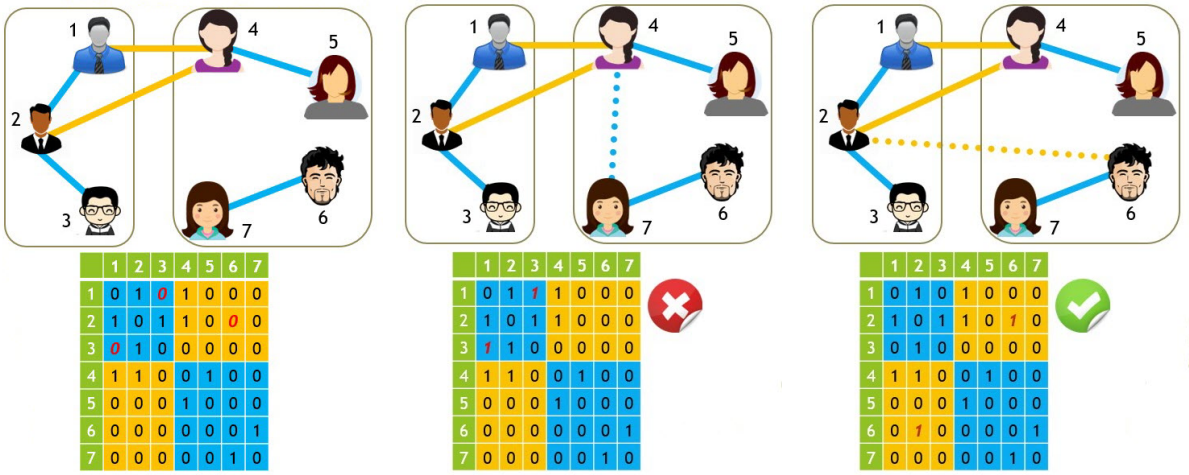


Figure 14: Example of partition policy.

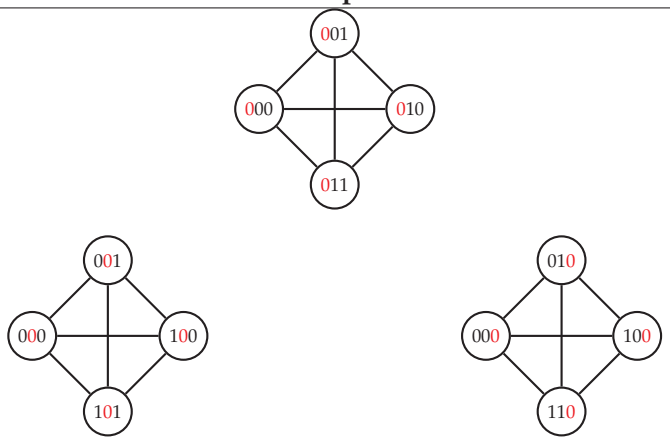
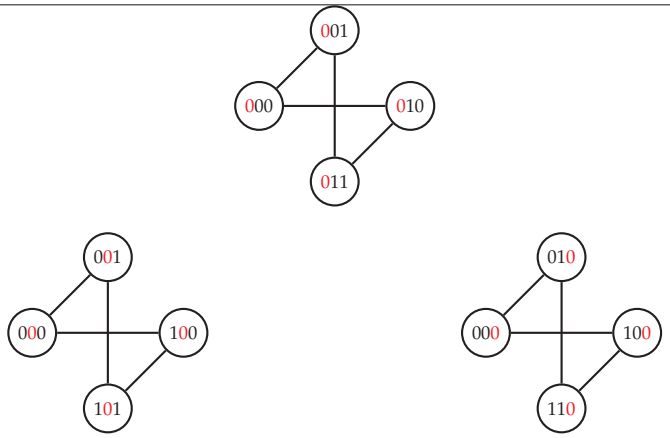
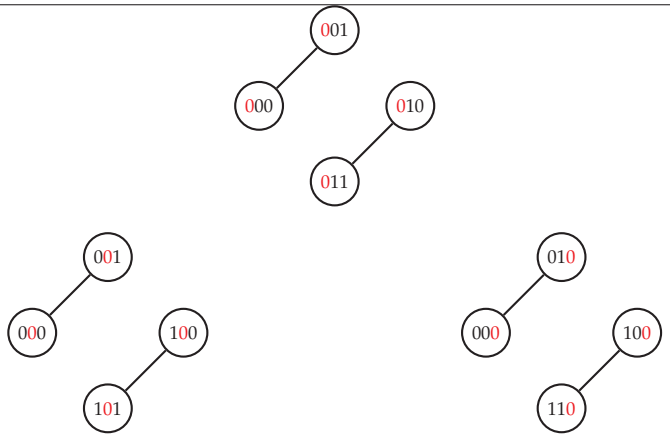
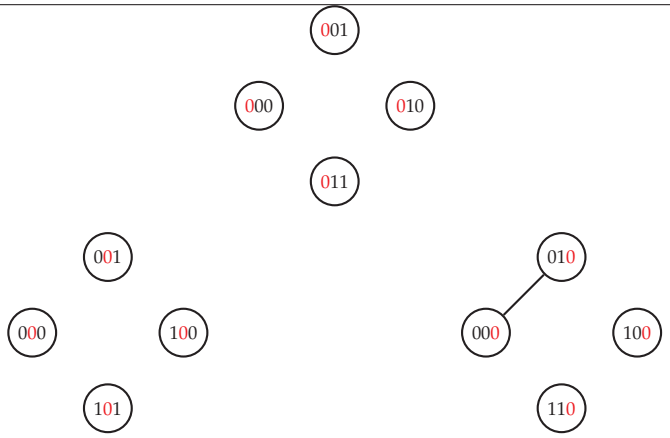
the existence or non-existence of these two edges together. Thus, the first and third graphs are not neighbors.

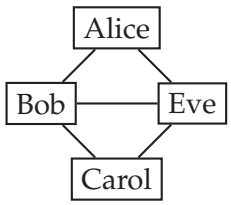
- Partition policy: For partitioned policy, we take a scenario where the vertices are divided into two groups. Then the edges are also divided into two partitions: intra-groups which connect two vertices in the same group and inter-groups which connects a vertex in the first group with a vertex in the second group. Here we are interested in protecting the inter-group (orange edges). Then, the first two graphs in Figure 14 that differ by just on edge are not neighbors because adding or removing an intra-group edge doesn't form a secret, while, the first and third graphs are neighbors because they differ by one inter-group edge which form a secret.

3.3 .5 Auxiliary knowledge

Auxiliary knowledge is usually formalized using correlations, for example $c(R = r_1) + c(R = r_2) = a_1$ where $c(r_1)$ is the count of records having the attribute R equal to r_1, $c(r_2)$ is the count of records having the attribute R equal to r_2, and a_1 is known. Blowfish suggests to formalize auxiliary knowledge in terms of a set of constraints Q that a database D must satisfy. It denotes

Table 6: Examples of discriminative secret graphs for a communication graph of three nodes

Policy	Graph
$G_S^{\text{full}} \equiv G_S^{(d=L_1,\theta=2)}$	
$G_S^{\text{attr}} \equiv G_S^{(d=L_1,\theta=1)}$	
$G_S^{\mathcal{P}}$ The partition is based on the value of the first non-ego attribute	
$G_S^{(d=L_1,\theta=3)}$ No challenges	



id	tuple
Bob	(0, 1, 1, 1)
Alice	(1, 0, 1, 0)
Eve	(1, 1, 0, 1)
Carol	(1, 0, 1, 0)

Figure 15: Communication graph and its corresponding database

$\mathcal{I}_Q \subset \mathcal{I}_n$ the subset of all possible database instances. In the case of undirected communication graphs, we have two inherent constraints:

- the matrix of D is symmetric: $t_j^i = t_i^j$
- the ego attributes are zero: $t_i^i = 0$

It is also possible to use directed communication graphs where a directed edge from Bob to Alice means that Bob has called Alice, or initiated a session. In this case the first constraint above is not considered.

Additional constraints which are not necessarily inherent to the graph representation can be considered, for example:

- *Count queries*: the number of individuals that have 5 neighbors.
- *Marginal constraints*: A marginal is the projection of the database on a given subset of columns. Rows having the same projection are grouped in one record along with their count. In our context we project on a subset of nodes. For example let's project the database in Figure 15 on Bob and Eve only (columns 1 and 3). Alice and Carol have the same projection since both have called Bob and Eve. Therefore the projection have 3 rows: (Bob,1), (Eve,1) and (Alice-Carol,2).
- *Meta-node constraints*: A meta-node is a node representing a sub-graph or a group of individuals. Meta-node auxiliary knowledge is for example the number of people calling a group of individuals, or the number of calls in between two groups of individuals. The adversary may know that the group Bob-Carol and the group Alice-Eve have three calls linking them.
- *Clique constraints*: A clique is a complete graph. The adversary may know that a group of nodes makes a clique or nearly a clique. For example Bob, Alice and Eve form a clique.

3.3 .6 Blowfish policy and privacy definitions

To apply Blowfish privacy, one must define a policy $P(\mathcal{T}, G_S, \mathcal{I}_Q)$ which is composed of a set of tuples $\mathcal{T}$, a discriminative secret graph $G_S(V_S, E_S)$ based on sets of discriminative pairs S_{pairs},

and a set of possible database instances $\mathcal{I}_Q$ under the auxiliary knowledge constraints. One also has to devise a randomized mechanism $\mathcal{M}$ that satisfies (ϵ, P)-Blowfish privacy. Concretely, for every pair of neighboring databases, denoted $(D_1, D_2) \in N(P)$, and every set of outputs $S \subseteq \text{range}(\mathcal{M})$, we have:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D_2) \in S]$$

To see how it differs from differential privacy, let's consider $D_1 = D \cup \{x\}$ and $D_2 = D \cup \{y\}$, two databases that differ in one tuple, and suppose $P = (\mathcal{T}, G_S, \mathcal{I}_n)$, i.e., no constraints. D_1 and D_2 are not considered neighbors unless $(s_x^i, s_y^i) \in S_{\text{pairs}}^G$. Otherwise, having $\mathcal{M}$ that satisfies (ϵ, P)-Blowfish privacy means that:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon . d_G(x,y)} \Pr[\mathcal{M}(D_2) \in S]$$

since Blowfish privacy is shown to satisfy sequential composition. Similarly to increasing ϵ, the chance of an attacker to distinguish between pairs farther apart in the graph is higher. We gain overall utility by scarifying local privacy of some users.

The Laplace mechanism $\mathcal{M}^{\text{Lap}}$ ensures $(\epsilon,\ P(\mathcal{T}, G_S, \mathcal{I}_Q))$-Blowfish privacy for any query function, $f : \mathcal{I}_Q \to \mathbb{R}^d$, by outputting $f(D) + \eta$ where $\eta \in \mathbb{R}^d$ is a vector of independent random numbers drawn from $\text{Lap}(S(f, P)/\epsilon)$. $S(f, P)$ is the policy-specific global sensitivity and is defined as:

$$\max_{(D_1,D_2)\in N(P)} \|f(D_1) - f(D_2)\|_1$$

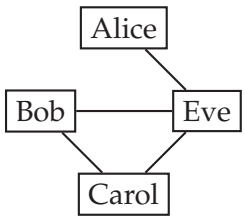
.

Following the definition of neighbors in [71], let $T(D_1, D_2)$ the set of discriminative pairs (s_x^i, s_y^i) such as the ith tuples in D_1 and D_2 are x and y. Let $\Delta(D_1, D_2) = D_1 \backslash D_2 \cup D_2 \backslash D_1$. D_1 and D_2 are neighbors, if:

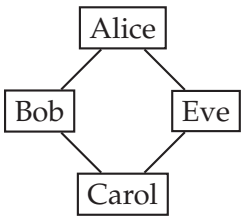
1. they both comply to the constraints,
2. $T \neq \emptyset$,
3. T has the smallest size, there is no feasible database D_3 such that $T(D_1, D_3) \subset T(D_1, D_2)$ or $T(D_1, D_3) = T(D_1, D_2)$ & $\Delta(D_1, D_3) \subset \Delta(D_1, D_2)$.

In our communication graph representation, two databases are candidate neighbors if they differ by the ego network of one individual, and this difference is represented in the security graph of that individual. Note that this means that one or several edges might be added or removed between two neighbor communication graphs.

To give an example, the two graphs in Figure 16 are different in three tuples: $|\Delta(D_1, D_2)| = 6$. If only one of the different pairs is in the security graph, for instance Bob's pairs, we have $|T| = 1$. There is no database having a non-empty subset of T, and no feasible database with same T and a subset of Δ. (To do so, we need to make Alice's ego network indifferent, or Eve's ego network indifferent, which is not possible due to symmetry constraints). We consider that these two graphs are neighbors.



id	tuple
Bob	(0, 0, 1, 1)
Alice	(0, 0, 1, 0)
Eve	(1, 1, 0, 1)
Carol	(1, 0, 1, 0)



id	tuple
Bob	(0, 1, 0, 1)
Alice	(1, 0, 1, 0)
Eve	(0, 1, 0, 1)
Carol	(1, 0, 1, 0)

Figure 16: Two neighboring graphs and their corresponding databases. The ego network of Bob has changed, and Bob has a G_S^{full} policy.

Under $P(\mathcal{T}, G_S^{\text{full}}, \mathcal{I}_Q)$, we can obtain two neighbor communication graphs by taking one vertex and changing its ego network. Any two communication graphs that differ in $n+1$ tuples where n tuples differ in one bit and one tuple differs in n bits are considered neighbors under G_S^{full}.

To make the concept of neighbor databases used throughout the paper more straightforward, we demonstrate the following result:

Theorem 1. *Given a communication graph $G(V, E)$, its database/matrix representation $M(G)$ and the policy $P(\mathcal{T}, G_S, \mathcal{I}_Q)$, where $\mathcal{T}$ represents all binary vectors of size $|V|$, G represents the overall graph of discriminative secret graphs for all the nodes, and $\mathcal{I}_Q$ constrains the possible databases to have: (1) $\forall i \neq j, M_{i,j} = M_{j,i} = 0$ or $M_{i,j} = M_{j,i} = 1$ and (2) $\forall i, M_{i,i} = 0$. If $G_S = G_S^{attr}$ or G_S is any non-empty subset of G_S^{attr}, we have that: Two graphs G^1 and G^2 are neighbors, i.e., $(G^1, G^2) \in N(P)$, if they differ by one and only one edge $e(i, j) = e(j, i)$ and for at least one vertex of the edge (either i or j) the discriminative secret pair (s_x^i, s_y^i) (where x and y differ at the bit j) or (s_a^j, s_b^j) (where a and b differ at the bit i) is in the security graph G_S.*

Proof. G_S^{attr} means that two tuples form a discriminative secret pair if they differ by only one attribute. This difference is reflected in the communication graph by the addition or removal of one edge.

If G^1 and G^2 differ by one or more edges that do not correspond to discriminative secret pairs in the security graph G, then $T(G^1, G^2) = \emptyset$ and the graphs are not neighbors.

If G^1 and G^2 differ by many edges that affect many secret pairs in G, then we can build a graph G^3 that takes only one of these edges that affects one (or two) secret pairs in G to form a subset of $T(G^1, G^2)$, and therefore the two graphs are not neighbors.

For the case where G^1 and G^2 differ by many edges and for only one of them $e(i, j)$ we have $(s_x^i, s_y^i) \in G$ or $(s_a^j, s_b^j) \in G_S$ or both, then $T(G^1, G^2)$ is the minimal possible set. But we can find a sub-graph G^3 of G^2 where $T(G^1, G^2) = T(G^1, G^3)$ by removing the extra edges which do not have any discriminative secret pairs that belong to the security graph. Then, G^1 and G^2 are not neighbors.

For the case where G^1 and G^2 differ by only one edge $e(i,j)$, and we have $(s_x^i, s_y^i) \in G_S$ or $(s_a^j, s_b^j) \in G_S$ or both, then $T(G^1, G^2)$ is minimal and there is no feasible intermediate database. Only in this remaining case G^1 and G^2 are neighbors.

□

3.3 .7 Blowfish with individualized security graphs

We consider the possibility that different individuals may have different security graphs. For example, we can divide the users into two extreme sub-groups: VIP and Standard. The discriminative secret graph for a VIP user is complete (i.e. full protection) or attribute-based. The discriminative secret graph for a standard user has 0 edges (i.e. null protection).

The application of Theorem 1 to the case where standard nodes' security graph is G_S^{empty} and VIP nodes' security graph is G_S^{attr} can be explained as follows. Take two communication graphs that differ by only one edge:

- **Case I:** If the vertices of the edge are standard nodes, then $T(G^1, G^2) = \emptyset$ and the two graphs are not neighbors.
- **Case II:** If one of the vertices is VIP and the other is standard, then the size of T is 1 and the two graphs are neighbors.
- **Case III:** If the vertices of the edge are two VIP nodes, then the size of T is 2 and the two graphs are neighbors. Any intermediate database that makes $\mid T \mid = 1$ is infeasible.

3.3 .8 Blowfish with double security graphs

In the previous subsection, we focused on VIP nodes having security graph, while Standard nodes are left with no protection.But, Standard nodes might need some level of protection even if it's less than the VIP nodes protection. Thus, in this subsection, we propose using a security graph for each type of nodes, i.e., Full Policy for VIP and Attribute Policy for Standard.

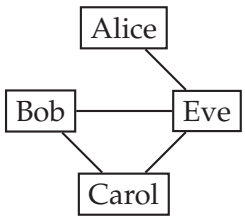
Let T_{VIP} and T_{STD} be the sets of discriminative pairs under the policies applied on VIP and Standard nodes respectively. The two graphs G^1 and G^2 are neighbors under Duo Policy of Blowfish Privacy if:

1. they both comply to the constraints,
2. $T_{VIP} \neq \emptyset$, $T_{STD} \neq \emptyset$
3. $T_{VIP} \cup T_{STD}$ has the smallest size,it means that each of the two sets has a size of 1.

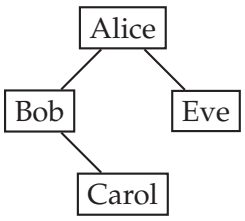
Theorem 2. *Given a communication graph $G(V,E)$, its database/matrix representation $M(G)$ and the policy $P(\mathcal{T}, G_{S_{VIP}}, G_{S_{STD}}, \mathcal{I}_Q)$, where $\mathcal{T}$ represents all binary vectors of size $|V|$, $G_{S_{VIP}}$ and $G_{S_{STD}}$ represent the overall graph of discriminative secret graphs for the VIP and Standard nodes respectively, and $\mathcal{I}_Q$ constrains the possible databases to have: (1) $\forall i \neq j, M_{i,j} = M_{j,i} = 0$ or $M_{i,j} = M_{j,i} = 1$ and*

(2) $\forall i, M_{i,i} = 0$. If $G_{S_{VIP}} = G_S^{full}$ and $G_{S_{STD}} = G_S^{attr}$ or $G_{S_{VIP}}$ and $G_{S_{STD}}$ are non-empty subsets of G_S^{full} and G_S^{attr} respectively, we have that: Two graphs G^1 and G^2 are neighbors, i.e., $(G^1, G^2) \in N(P)$, if they differ by

- *one or more edges that form the ego network of one and only one node z, we have (s_a^z, s_b^z) (where a and b differ at the bit t and t is a node connected to z in G^1 or G^2) is in the security graph G_S^{full}.*
- *in addition to one edge $e(i,j) = e(j,i)$ connecting two Standard nodes and for at least one vertex of the edge (either i or j) the discriminative secret pair (s_x^i, s_y^i) (where x and y differ at the bit j) or (s_a^j, s_b^j) (where a and b differ at the bit i) is in the security graph G_S^{attr}.*



id	tuple
Bob	(0, 0, 1, 1)
Alice	(0, 1, 0)
Eve	(0, 1)
Carol	(0)



id	tuple
Bob	(0, 1, 0, 1)
Alice	(0, 1, 0)
Eve	(0, 0)
Carol	(0)

Figure 17: Two neighboring graphs and their corresponding databases. The ego network of Bob has changed, and Bob has a G_S^{full} policy.

In Figure 17, we consider Bob as a VIP node and the three others as Standards. We consider two policies to protect these graphs, the Full Policy for VIP nodes and Attribute Policy for Standard nodes. The two graphs shown are neighbors because just one ego network of a VIP node (Bob) has changed and one edge between two Standard nodes (Carol-Eve) has changed.

Take two communication graphs that differ by only one ego network of v_{VIP_0} and one edge e_{dist} connecting v_{STD_0} to another node:

- **Case I:** If the node e_{dist} connects v_{STD_0} to a VIP node other than v_{VIP_0}, than two VIP ego networks are changed, $\mid T_{VIP} \mid = 2$ then the two graphs are not neighbors.
- **Case II:** If the node e_{dist} connects v_{STD_0} to v_{VIP_0}, in other words, just the ego network of v_{VIP_0} has changed, $\mid T_{VIP} \mid = 1$ and $\mid T_{STD} \mid = 0$, then the two graphs are not neighbors.
- **Case III:** If the node e_{dist} connects v_{STD_0} to another Standard node, $\mid T_{VIP} \mid = 1$ and $\mid T_{STD} \mid = 1$, then the two graphs are neighbors.

3.3 .9 Global Sensitivities of graph queries and measures

To apply Blowfish privacy given a query or a function f over the protected database D, one has to determine first the global sensitivity [54] of f, based on the privacy policy $P = (\mathcal{T}, G_S, \mathcal{I}_Q)$:

$$S(f,P) = \max_{(D_1,D_2)\in N(P)} ||f(D_1) - f(D_2)||_1$$

Once the global sensitivity $S(f, P)$ is identified, outputting $f(D) + \eta$ ensures (ϵ, P)-Blowfish privacy if $\eta \in \mathbb{R}^d$ is a vector of independent random numbers drawn from $Lap(S(f, P)/\epsilon)$.

3.3 .9.1 Histogram graph queries

In the coming examples, we compute the global sensitivity of four histogram queries to compare their values under Full and Attribute policies, and in flat graph and graph containing VIP and Standard nodes.

3.3 .9.1.1 Example 1: Complete histogram query for degrees of vertices

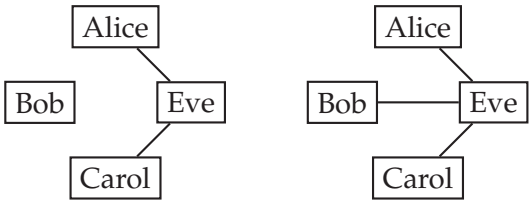
Under P_{attr}, two graphs G^1 and G^2 are neighbors if $G^2 = G^1 \cup \{e\}$. Assume $DV = dv_1, ..., dv_{|DV|}$ is the set of all possible degrees for the vertices in these graphs.

If the edge e is added between node a having degree dv_i and node b with degree dv_j, $i \neq j$, then the count of dv_i and dv_j will decrease each by 1 while dv_{i+1} and dv_{j+1} will increase each by 1, as shown in Figure 18.

If the edge e is added between two nodes both having the same degree dv_i, then the count of dv_i will decrease by 2 and dv_{i+1} will increase by 2, as shown in Figure 19.

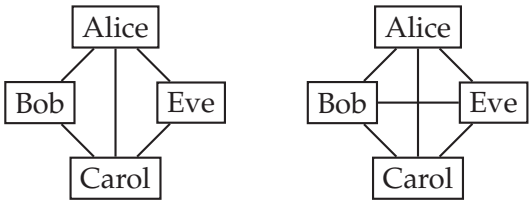
Taking both cases into account, the global sensitivity is $S(f_{\text{complete}}, P_{\text{attr}}) = 4$.

Under P_{full}, two graphs G^1 and G^2 are neighbors if they differ by the ego network of one vertex. In the worst case, the vertex passes from degree 0 to degree $n-1$, where n is the number of vertices in the graph. All the other vertices have their degrees shifted by $+1$. In total, $2n$ bins are affected and the sensitivity is $2n$.



degree	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
count	1	2	1	0	0	3	0	1
cumulative count	1	3	4	4	0	3	3	4

Figure 18: Counts and cumulative counts of node degree for graphs G^1 and G^2, the added edge e connects two nodes of different degrees.



degree	**2**	**3**	**2**	**3**
count	2	2	0	4
cumulative count	2	4	0	4

Figure 19: Counts and cumulative counts of node degree for graphs G^1 and G^2, the added edge e connects two nodes of the same degree.

3.3 .9.1.2 Example 2: Cumulative histogram query for degrees of vertices

Under P_{attr}, adding edge e between two nodes a of degree dv_i and b of degree dv_j decreases the cumulative count of dv_i (respectively dv_j) by 1, yet the cumulative count of dv_{i+1} (respectively dv_{j+1}) stays unchanged, as shown in Figure 18.

Adding edge e between two nodes of the same degree dv_i decreases the cumulative count of dv_i by 2, yet the cumulative count of dv_{i+1} stays unchanged, as shown in Figure 19. In both cases, the global sensitivity of a cumulative histogram query is $S(f_{\text{cumulative}}, P_{\text{attr}}) = 2$.

Under P_{full} and a worst case scenario, n vertices change their degrees and move from one bin to another, however the receptive bin does not change its count. The sensitivity is n.

3.3 .9.1.3 Example 3: Histogram of degrees of vertices for standard nodes

In this exercise, we divide the communication graph vertices into two groups: VIP nodes and standard nodes. The discriminative secret graph for a VIP node is built as follows: There is no edge between two tuples if they differ by more than one attribute (i.e. G_S^{attr}). In addition, we consider only attributes that belong to a VIP vertex. Two tuples differing by an attribute corresponding to a standard node are not connected in the discriminative secret graph. We denote this set of secret pairs: $\mathcal{S}_{\text{pairs}}^{\text{attr,VIP}}$.

Consider the following query: "Histogram of degrees of vertices for standard nodes". To compute their sensitivity we examine the three cases:

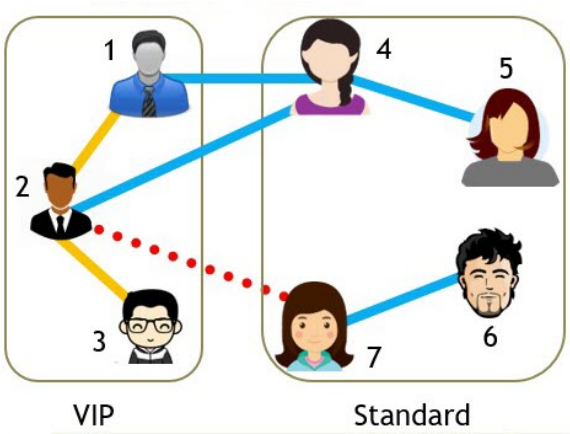
1. the edge we add/remove is between two VIP nodes: nothing will change in the histogram of the query,
2. the edge we add/remove is between one VIP node and one standard node: one of the bins in the histogram will decrease by 1 and its right-hand neighbor will increase by 1, as we can see in Figure 20 where we count the degrees of just the standard nodes,
3. the edge we add/remove is between two standard nodes. This edge does not correspond to a secret pair. It means that this case will not occur for two neighbor graphs and can therefore be ignored.

It follows that the sensitivity of this query under $\mathcal{S}_{\text{pairs}}^{\text{attr,VIP}}$ is only 2. The sensitivity is reduced by 50% in comparison to the full histogram query. By limiting the privacy focus to the VIP nodes, we gain in terms of utility for queries over the standard nodes.

3.3 .9.1.4 Example 4: Histogram of the number of connections between VIP nodes and standard nodes

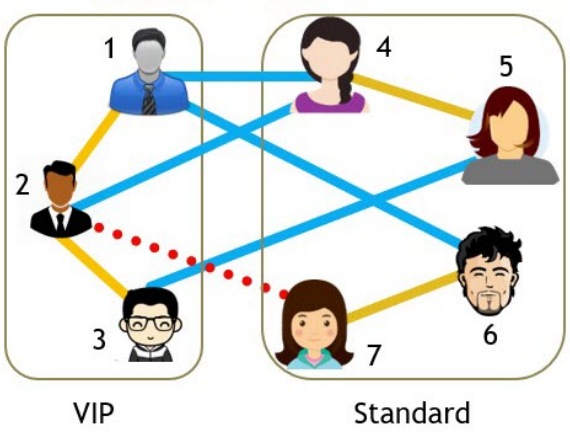
A similar query is the "Histogram of the number of connections between a VIP node and standard nodes" or "Histogram of the number of connections between a standard node and VIP nodes" . To compute their sensitivity we examine the three cases:

1. the edge we add/remove is between two VIP nodes: nothing will change in the query's result,



Degree	1	2	3
Count of nodes for G_1^C	3	0	1
Count of nodes for G_2^C	2	1	1
Difference	1	1	0

Figure 20: Example of changes in histogram of degrees of vertices for standard nodes.



Degree	0	1	2
Count of nodes for G_1^C	1	4	2
Count of nodes for G_2^C	0	3	2
Difference	1	1	0

Figure 21: Example of changes in histogram of the number of connections between VIP nodes and standard nodes.

2. the edge we add/remove is between one VIP node and one standard node: two of the bins in the histogram will vary by ± 1, as we can see in Figure 21 where we take into consideration just the inter-group edges,

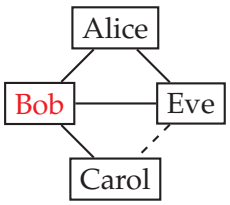
3. the edge we add/remove is between two standard nodes. This edge does not correspond to a secret pair and does not change the query result in the same time.

The sensitivity of these queries under $\mathcal{S}_{\text{pairs}}^{\text{attr,VIP}}$ is 2.

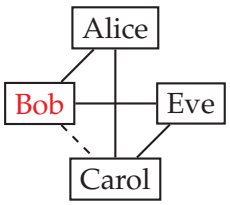
These queries are useful in a graph where the standard nodes are the members of a company's support team and the VIP nodes are the customers. The calls between a support team member and the customers are the target. We aim to study, for example, if a load balancing strategy works well, or how many clients a support member is serving in average. At the same time, we are protecting the privacy of the customers.

3.3 .9.2 Global and Local Clustering Coefficient

The clustering coefficient is a major descriptive statistics and one of the most important properties in graphs. It quantifies how well connected are the neighbors of the nodes in the graph. In this subsection we will compute the global sensitivity for three types of clustering coefficient.



(a) First case.



(b) Second case.

Cases	T^1_Δ	T^2_Δ	$\Delta\mathsf{T}_\Delta$	T^1_3	T^2_3	$\Delta\mathsf{T}_3$
First	1	2	1	3	3	0
Second	0	2	2	1	3	1

Figure 22: Two cases for G^1 and G^2 as neighbors under P_{distance}.

3.3 .9.2.1 Local clustering coefficient

This coefficient is the likelihood that the neighbours of v_{VIP} are connected between each others. Let v_{VIP} be the only VIP node in a graph G^1 while all other nodes are labeled as standard. Let $lcc(v_{VIP})$ be the proportion of neighbors of v_{VIP} that are also connected to each other.

$$lcc(v_{VIP}) = \frac{\mathsf{T}_\Delta(v_{VIP})}{\mathsf{T}_3(v_{VIP})}$$

where $\mathsf{T}_\Delta(v_{VIP})$ is the number of triangles formed by v_{VIP} and two of its neighbors, and $\mathsf{T}_3(v_{VIP})$ is the number of triplets in which node v_{VIP} is the middle node.

Therefore, we propose an attribute policy where the tuples of the discriminative graph represent the relation between the v_{VIP} and all other nodes. In case the attribute indicates that the node is a neighbor of v_{VIP}, then sub-attributes exist to represent the relation between this neighbor and all other neighbors. Two tuples are discriminative secrets pair if just one of their attributes or sub-attributes are different. By making the relation between v_{VIP} and its neighbors a secret, we are protecting the real value of $\mathsf{T}_3(v_{VIP})$. In addition, by making also the relation between neighbors a secret, we are protecting the real value of $\mathsf{T}_\Delta(v_{VIP})$.

Let G^1 and $G^2 = G^1 \cup \{e_{dist}\}$ be two graphs that have one VIP node each and differ in one edge e_{dist}. G^1 and G^2 are neighbors, if e_{dist} connects v_{VIP} to another node, or connects two neighbors of v_{VIP}.

Because we have two cases for distinct edge e_{dist} between G^1 and G^2, then we have to compute the sensitivity of each case. In the first case, where e_{dist} connects two neighbors nodes of v_{VIP}, the number of triplets T^1_3 in G^1 and T^2_3 in G^2 are similar while the number of triangles increments by 1.

$$\begin{aligned} S(f_{\text{lcc}}, P_{\text{attr}})_1 &= \left| \frac{\mathsf{T}^2_\Delta}{\mathsf{T}^2_3} - \frac{\mathsf{T}^1_\Delta}{\mathsf{T}^1_3} \right| \\ &= \left| \frac{\mathsf{T}^1_\Delta + 1}{\mathsf{T}^1_3} - \frac{\mathsf{T}^1_\Delta}{\mathsf{T}^1_3} \right| \\ &= \frac{1}{\mathsf{T}^1_3} \end{aligned}$$

In Figure 22, in the first case where node *Bob* is the VIP node and $edge(Carol, Eve)$ is the distinct edge between G^1 and G^2, we can see that $\Delta \mathsf{T}_\Delta$ is 1 and $\mathsf{T}_3^1 = \mathsf{T}_3^2 = 3$, then $S(f_{\text{lcc}}, P_{\text{attr}})_1 = \frac{1}{3}$.

In the second case, e_{dist} connects v_{VIP} to a standard node. It means that the degree and the number of triplets for v_{VIP} will increment between G^1 and G^2. When adding e_{dist} to v_{VIP}, this new edge will form a new triplet with each of the other adjacent edge connected to v_{VIP}. Then, by adding one edge, the number of triplets will increment by deg where deg is the degree of v_{VIP} in G^1. Thus, $\mathsf{T}_3^2 = \mathsf{T}_3^1 + deg$.

On another hand, the number of new triangles created by adding e_{dist} that connects v_{VIP} to v_{std} is the number of common neighbors $N_{com}(v_{std})$ of v_{VIP} and v_{std}. For example, in the second case in Figure 22, where *Bob* is the VIP node and $edge(Bob, Carol)$ is the distinct edge between G^1 and G^2, *Alice* and *Eve* are two common neighbors for *Bob* and *Carol*, thus, $\mathsf{T}_3^2 = \mathsf{T}_3^1 + 2$. Therefore, the sensitivity is:

$$\begin{aligned} S(f_{\text{lcc}}, P_{\text{attr}})_2 &= \mid \frac{\mathsf{T}_\Delta^2}{\mathsf{T}_3^2} - \frac{\mathsf{T}_\Delta^1}{\mathsf{T}_3^1} \mid \\ &= \mid \frac{\mathsf{T}_\Delta^1 + N_{maxcom}}{\mathsf{T}_3^1 + deg} - \frac{\mathsf{T}_\Delta^1}{\mathsf{T}_3^1} \mid \end{aligned} \tag{1}$$

where $N_{maxcom} = \underset{std=1,\ldots,N_s}{max}(N_{com}(v_{std}))$ and N_s is the number of standard nodes in G^1 except the number of v_{VIP} neighbors.

In the second case of Figure 22, by applying the sensitivity on that example, we get $S(f_{\text{lcc}}, P_{\text{attr}})_2 = \frac{0+2}{1+2} - \frac{0}{1} = \frac{2}{3}$.

Finally, as a result of having two sensitivities in two cases that differ by the type of nodes connected by e_{dist}, the sensitivity of local clustering coefficient for a VIP node is:

$$\begin{aligned} &S(f_{\text{lcc}}, P_{\text{attr}}) = \\ &max(S(f_{\text{lcc}}, P_{\text{attr}})_1, S(f_{\text{lcc}}, P_{\text{attr}})_2) \end{aligned}$$

To ensure that T_3^1 is never equal to 0, the degree of v_{VIP} should be greatest or equal to 2, as a constraint Q applied on the communication graph.

3.3 .9.2.2 Global Clustering Coefficient

The global clustering coefficient gcc is the number of closed triplets (3×triangles) over the total number of triplets (both open and closed).

$$gcc = \frac{3 \times \mathsf{T}_\Delta}{\mathsf{T}_3}$$

where T_Δ is the total number of triangles in the graph and T_3 is the total number of triplets.

In this subsection, we compute the global sensitivity of average local clustering coefficient of two neighbors graphs G^1 and $G^2 = G^1 \cup \{e_{dist}\}$ under Attribute Policy P_{attr}. Adding or removing an edge between two nodes v_t and v_z will affect the number of open and closed triplets containing at least one of these nodes or one of their neighbors.

v_t, v_z and each of their common neighbors represents an open triplet. By adding an edge between v_t and v_z in G^2, these open triplets turn into triangles. Therefore, The number of triangles in G^2 increments by $N_{com}(v_t, v_z)$, the number of common neighbors for v_t and v_z.

Each neighbor of just one of the two nodes v_t and v_z in G^1 creates an open triplet, with both of them, in G^2. Thus, the number of triplets in G^2 increases by $N_{dist}(v_t, v_z) + N_{dist}(v_z, v_t)$, the number of distinct neighbors for v_t and v_z in G^1.

The difference in global clustering coefficient between G^1 and G^2, if $e_{dist} = (v_t, v_z)$ is:

$$\left| \frac{3 \times \mathsf{T}^2_\Delta}{\mathsf{T}^2_3} - \frac{3 \times \mathsf{T}^1_\Delta}{\mathsf{T}^1_3} \right| = \left| \frac{3 \times (\mathsf{T}^1_\Delta + N_{com}(v_t, v_z))}{\mathsf{T}^1_3 + N_{dist}(v_t, v_z) + N_{dist}(v_z, v_t)} - \frac{3 \times \mathsf{T}^1_\Delta}{\mathsf{T}^1_3} \right| \tag{2}$$

Therefore, the global sensitivity of global clustering coefficient in a graph under Full Policy is:

$$S(f_{\text{gcc}}, P_{\text{full}}) = \max_{t,z \in E_{dist}} \left(\left| \frac{3 \times (\mathsf{T}^1_\Delta + N_{com}(v_t, v_z))}{\mathsf{T}^1_3 + N_{dist}(v_t, v_z) + N_{dist}(v_z, v_t)} - \frac{3 \times \mathsf{T}^1_\Delta}{\mathsf{T}^1_3} \right| \right) \tag{3}$$

where E_{dist} is the set of all possible edges of e_{dist}, in other words, E_{dist} is the set of nodes in G^1 that don't form an edge: $\forall G^1(V_1, E_1), E_{dist} = \{(v_t, v_z) \notin E_1 \mid v_t, v_z \in V_1\}$.

3.3 .9.2.3 Average Local Clustering Coefficient

Adding or removing an edge between two nodes will affect their local clustering coefficient in addition to their neighbors'. In this subsection, we compute the average local sensitivity of two neighbors graphs G^1 and $G^2 = G^1 \cup \{e_{dist}\}$, relying on an attribute policy similar to the one used in the Global Clustering Coefficient.

By adding one edge e_{dist} connecting v_t and v_z, the number of triangles for each of the two nodes increased by $N_{com}(v_t, v_z)$, and the number of triplets for each of them increases by $N_{dis}(v_t, v_z)$ and $N_{dis}(v_z, v_t)$ respectively.

The local clustering coefficients of the neighbors of v_t and v_z are also affected by e_{dist}. These neighbors are two types, common and distinct neighbors. For each common neighbor, e_{dist} will form a new triangle, while for a distinct neighbor, it will form a new open triplet. Therefore, by adding this edge, the average local clustering coefficient will differ between G^1 and G^2 by:

$$\begin{aligned} \Delta_{alcc}(v_t, v_z) = \tfrac{1}{N} \times \Bigg(& \left| \tfrac{T^1_\Delta + N_{com}(v_t,v_z)}{T^1_3 + N_{dis}(v_t,v_z)} - \tfrac{T^1_\Delta}{T^1_3} \right| \\ & + \left| \tfrac{T^1_\Delta + N_{com}(v_t,v_z)}{T^1_3 + N_{dis}(v_z,v_t)} - \tfrac{T^1_\Delta}{T^1_3} \right| + \tfrac{N_{com}(v_t,v_z)}{T^1_3} \\ & + \Big((N_{dis}(v_t, v_z) + N_{dis}(v_z, v_t)) \times (\tfrac{T^1_\Delta}{T^1_3 + 1} - \tfrac{T^1_\Delta}{T^1_3}) \Big) \Bigg) \end{aligned} \tag{4}$$

where N is the number of nodes in the graph.

Finally, the global sensitivity of average local clustering coefficient is:

$$S(f_{\text{alcc}}, P_{\text{full}}) = \max_{(v_t,v_z) \in E_{dist}} \Delta_{alcc}(v_t, v_z) \tag{5}$$

Computing the sensitivities for a flat graph or for a graph containing VIP and Standard nodes are similar but the number of nodes is reduced from N to N_{VIP}. In the experiments,

we extract subgraphs containing the VIP nodes, then we compute and compare the global sensitivities of the complete graphs and the VIP subgraphs.

3.3 .9.3 Queries of shortest paths in graphs

We present in this subsection the calculations of global sensitivities for some shortest paths queries which will help in calculating the sensitivities for some Centralities queries in the next subsection.

3.3 .9.3.1 Length of shortest-paths of two VIP nodes

Let v^1_{VIP} and v^2_{VIP} be the only VIP nodes in a graph G^1. The query in this example is to compute the length between the two VIP nodes in the graph. We use an attribute policy $P_{\text{attr}} = (\mathcal{T}, G^{attr}_S, \mathcal{I}_Q)$, where $\mathcal{T}$ is the universe of tuples containing attributes that represent all the possible edges in the communication graph G^1. $G^{attr}_S = (V, E)$ where $V = \mathcal{T}$ and every edge in E connects two tuples that have just one attribute flipped. In this way, we ensure that G^1 and $G^2 = G^1 \cup \{e_{dist}\}$ are always neighbors regardeless of the nodes connected by e_{dist}. Let $sp^1(v^1_{VIP}, v^2_{VIP})$ be the length of shortest-path between v^1_{VIP} and v^2_{VIP} in G^1, $S(f^1_{\text{sp}}, P_{\text{attr}}) = sp^1(v^1_{VIP}, v^2_{VIP}) - 1$ because the largest possible reduction of shortest path between v^1_{VIP} and v^2_{VIP} is realised by adding an edge connecting these two nodes which makes their shortest-path equal to 1.

3.3 .9.3.2 Sum of Lengths of shortest-paths between all VIP nodes

In this example, the query is to compute the sum of lengths of shortest-paths between all the VIP nodes. Under the same attribute policy P_{attr} defined in Subsection 3.3 .9.3.1 , the length of shortest-path in G^2 between two VIP nodes v_t and v_z is :

$$\begin{aligned} sp^2(v_t, v_z) = min(&sp^1(v_t, v_z), \\ &sp^1(v_t, v_i) + sp^1(v_j, v_z) + 1, \\ &sp^1(v_t, v_j) + sp^1(v_i, v_z) + 1) \end{aligned}$$

where $sp^1(v_i, v_j)$ is the length of shortest-path between v_i and v_j in G^1 and e_{dist} connects v_i and v_j in G^2. The sensitivity of the query is:

$$\begin{aligned} S(f^2_{sp}, P_{attr}) = \max_{i,j=1,\ldots,N_{VIP}} \Bigg(&\sum_{t=1}^{N_{VIP}-1} \Bigg(\sum_{z=t+1}^{N_{VIP}} \Big(\\ &sp^1(v_t, v_z) - sp^2(v_t, v_z) \Big) \Bigg) \Bigg) \end{aligned}$$

3.3 .9.4 Graph Centralities

In this subsection, we present the global sensitivities under Attribute Policy for some graph centralities.

3.3 .9.4.1 Farness and Closeness of v_0 to other VIP nodes

Let G^1 be a graph composed of VIP nodes, standard nodes and a node v_0. The Farness of v_0 is to compute the sum of shortest-paths between v_0 and all VIP nodes under Blowfish Privacy, while the closeness is the reciprocal of the Farness. We compute the sensitivity of Farness and Closeness for node v_0 under the same policy in Subsection 3.3 .9.3.1 .

Let e_{dist} be (v_i, v_j), the length of shortest-path in G^2 between v_0 and v_z is :

$$\begin{aligned} sp^2(v_0, v_z) = min(&sp^1(v_0, v_z), \\ &sp^1(v_0, v_i) + sp^1(v_j, v_z) + 1, \\ &sp^1(v_0, v_j) + sp^1(v_i, v_z) + 1) \end{aligned}$$

because $sp^2(v_i, v_j) = 1$

Therefore the sensitivity of Farness under P_{attr} is:

$$\begin{aligned} &S(f_{farness}, P_{distance})(v_0) = \max_{i=1,\ldots,N_{VIP}-1} \Bigg(\\ &\max_{j=i+1,\ldots,N_{VIP}} \Bigg(\sum_{z=1}^{N_{VIP}} \Big(sp^1(v_0, v_z) - sp^2(v_0, v_z) \Big) \Bigg) \Bigg) \end{aligned}$$

where N_{VIP} is the number of VIP nodes in G^1 and $sp^1(v_i, v_i) = 0$.

As closeness is the reciprocal of farness, then its sensitivity is the reciprocal of the minimal sum of differences between lengths of shortest-paths in G^1 and G^2:

$$\begin{aligned} &S(f_{closeness}, P_{attr})(v_0) = \Bigg(\min_{i=1,\ldots,N_{VIP}-1} \\ &\Bigg(\min_{j=i+1,\ldots,N_{VIP}} \Bigg(\sum_{z=1}^{N_{VIP}} \Big(sp^1(v_0, v_z) - sp^2(v_0, v_z) \Big) \Bigg) \Bigg) \Bigg)^{-1} \end{aligned}$$

3.3 .9.4.2 Closeness Centrality for VIP nodes

In Subsection 3.3 .9.4.1 , we have found the sensitivity of closeness for one node v_0. In this subsection, we compute the sensitivity of closeness for each VIP node in the same way of Subsection 3.3 .9.4.1 , then, the sensitivity of Closeness Centrality for VIP nodes in a graph G^1 under P_{attr} is the maximum of all sensitivities of VIP nodes:

$$\begin{aligned} &S(f_{closeness\_centrality}, P_{attr})(v_0) = \\ &\max_{m=1,\ldots,N_{VIP}} (S(f_{closeness}, P_{attr})(v_m)) \end{aligned}$$

3.3 .9.4.3 Graph Degree Centrality

The degree centrality of nodes could be extended to measure the degree centrality of the whole graph known as the graph centralization. Let us say that G^1 and $G^2 = G^1 \cup \{e_{dist}\}$ are neighbors under an Attribute Policy P_{attr}. Let the query be to compute the graph centralization of G^1 based on just the VIP nodes under P_{attr}. The formula of the graph centralization of G^1 is

$$C_D(G^1) = \frac{\sum_{i=1}^{N_{VIP}} [deg(v*) - deg(v_i)]}{N_{VIP}^2 - 3N_{VIP} + 2} \tag{6}$$

where $v*$ is the VIP node with the highest degree.

To compute the global sensitivity, we have to find the case of largest possible difference between $C_D(G^1)$ and $C_D(G^2)$. The denominators of the two centralizations are obviously the same in all the cases because the number of VIP nodes does not change in the two neighbor graphs. Thus, we will focus on the difference of their numerators. We can list four cases for e_{dist}:

- The two nodes of e_{dist} are standards, the degrees of VIP nodes are the same in both graphs. Thus, $C_D(G^1) - C_D(G^2) = 0$.

- The two nodes of e_{dist} are both VIP (v_i and v_j but not $v*$).

$$\mid C_D(G^1) - C_D(G^2) \mid =$$

$$\mid \frac{[2\times deg(v*) - deg(v_i) - deg(v_j)] - [2\times deg(v*) - (deg(v_i)+1) - (deg(v_j)+1)]}{N_{VIP}^2 - 3N_{VIP} + 2} \mid$$

$$= \frac{2}{N_{VIP}^2 - 3N_{VIP} + 2}$$

- e_{dist} connects a VIP node v_i to a Standard node.

$$\mid C_D(G^1) - C_D(G^2) \mid =$$

$$\mid \frac{[deg(v*) - deg(v_i)] - [deg(v*) - (deg(v_i)+1)]}{N_{VIP}^2 - 3N_{VIP} + 2} \mid$$

$$= \frac{1}{N_{VIP}^2 - 3N_{VIP} + 2}$$

- e_{dist} connects $v*$ to a VIP node v_i.

$$\mid C_D(G^1) - C_D(G^2) \mid =$$

$$\mid \frac{deg(v*) - deg(v_i) + \sum_{j=1}^{N_{VIP}-1}[deg(v*) - deg(v_j)]}{N_{VIP}^2 - 3N_{VIP} + 2} -$$

$$\frac{(deg(v*)+1) - (deg(v_i)+1) + \sum_{j=1}^{N_{VIP}-1}[(deg(v*)+1) - deg(v_j)]}{N_{VIP}^2 - 3N_{VIP} + 2} \mid$$

$$= \frac{N_{VIP} - 1}{N_{VIP}^2 - 3N_{VIP} + 2}$$

- e_{dist} connects $v*$ to a Standard node.

$$\mid C_D(G^1) - C_D(G^2) \mid =$$

$$\mid \frac{\sum_{j=1}^{N_{VIP}}[deg(v*) - deg(v_j)]}{N_{VIP}^2 - 3N_{VIP} + 2} -$$

$$\frac{\sum_{j=1}^{N_{VIP}}[(deg(v*)+1) - deg(v_j)]}{N_{VIP}^2 - 3N_{VIP} + 2} \mid$$

$$= \frac{N_{VIP}}{N_{VIP}^2 - 3N_{VIP} + 2}$$

Then e_{dist} produce the largest difference in centralization between the two neighbors when it connects $v*$ to a standard node. Therefore, the global sensitivity of graph centralization is

$$S(f_{centralization}, P_{attr}) = \frac{N_{VIP}}{N_{VIP}^2 - 3N_{VIP} + 2} \tag{7}$$

3.3 .9.5 Efficiency

The efficiency measure is divided to three types [41]: nodal, local and global.

Nodal efficiency is a measure of the efficiency of information transfer between one VIP node v_t and all other VIPs.

$$E_{nodal} = \frac{1}{N_{VIP} - 1} \sum_{z=1}^{N_{VIP}-1} \frac{1}{sp(v_t, v_z)} \tag{8}$$

Local Efficiency is a measure of the efficiency of information transfer for the VIP neighbors N_{nbg} of v_t, excluding v_t.

$$E_{local} = \frac{1}{N_{nbg}(N_{nbg} - 1)} \sum_{z=1}^{N_{nbg}-1} \frac{1}{sp(v_t, v_z)} \tag{9}$$

And finally Global Efficiency is a measure of the efficiency of information for all the pairs of nodes on the graph and it represents an important measure for the robustness of the graph.

$$E_{global} = \frac{2}{N_{VIP}(N_{VIP} - 1)} \sum_{t=1}^{N_{VIP}-1} \sum_{z=t+1}^{N_{VIP}} \frac{1}{sp(v_t, v_z)} \tag{10}$$

Next we compute the difference between the three efficiencies of G^1 and $G^2 = G^1 \cup \{e_{dist}\}$.

$$\Delta E_{nodal}(e_{dist}) = \frac{1}{N_{VIP}} \sum_{z=1}^{E_d} \mid \frac{1}{sp^1(v_t, v_z)} - \frac{1}{sp^2(v_t, v_z)} \mid \tag{11}$$

$$\Delta E_{local}(e_{dist}) = \frac{1}{N_{nbg}(N_{nbg} - 1)} \sum_{z=1}^{E_d} \mid \frac{1}{sp^1(v_t, v_z)} - \frac{1}{sp^2(v_t, v_z)} \mid \tag{12}$$

$$\Delta E_{global}(e_{dist}) = \frac{2}{N_{VIP}(N_{VIP} - 1)} \sum_{t=1}^{|E_d|-1} \sum_{z=t+1}^{|E_d|} \mid \frac{1}{sp^1(v_t, v_z)} - \frac{1}{sp^2(v_t, v_z)} \mid \tag{13}$$

where E_d is the set of VIP pairs (t, z) having $sp^1(v_t, v_z) \neq sp^2(v_t, v_z)$. In other words, it is the set of VIP pairs which their shortest paths in G^2 pass by e_{dist}.

Thus, under Attribute Policy P_{attr}, the global sensitivity for any of these efficiencies is

$$S(f_{efficiency}, P_{attr}) = \max_{e_{dist} \in E_{dist}} \Delta E(e_{dist}) \tag{14}$$

3.3 .10 Experiments and Results

In this section, we present the results of our experiments to evaluate the Blowfish Privacy on graphs both in terms of utility and privacy. We compare G_S^{attr} and G_S^{full} for histogram queries. In addition, we show the utility of some queries under $G_S^{\text{attr,VIP}}$.

Our experiments use three graph datasets collected from the music streaming service Deezer (November 2017) [141]. These datasets represent the friendship Network of users from Croatia (HR) of 54,573 nodes and 498,202 edges, Hungary (HU) of 47,538 nodes and 222,887 edges and Romania (RO) of 41,773 nodes and 125,826 edges.

3.3 .10.1 Mean Squared Error of Complete Histogram Queries

The Mean Squared Error (MSE) for complete histogram queries under G_S^{attr} and G_S^{full} are:

$$\mathcal{E}_{\mathcal{M}^{\text{Lap}}_{h_{\mathcal{P}_{\text{attr}}}}}(D) = b\mathbb{E}[\text{Laplace}(4/\epsilon)^2] = 32b/\epsilon^2$$
$$\mathcal{E}_{\mathcal{M}^{\text{Lap}}_{h_{\mathcal{P}_{\text{full}}}}}(D) = b\mathbb{E}[\text{Laplace}(2n/\epsilon)^2] = 8n^2b/\epsilon^2$$

where n is the number of vertices and b is the number of bins. We empirically sample the MSE of the complete histogram query for a given ϵ by generating the real histogram of each graph (HR, HU, and RO) and k noisy versions. We compute the MSE of the noisy versions as follows:

$$\text{MSE}(H, H_\epsilon) = \sum_{i=1}^{b} \text{mean}_k[(H(i) - H_\epsilon(i))^2]$$

where H is the original histogram and and H_ϵ is its ϵ-noisy version. We choose ten epsilon values: 0.1, 0.2, ..., 1. The number of bins b is equal to 421, 113 and 113 for the graphs HR, HU and RO respectively. The results are shown in Figure 23 under G_S^{full} and in Figure 24 under G_S^{attr}. G_S^{full} has null utility whereas for G_S^{attr} we expect the standard deviation per bin to be around 32 to 56 (depending on ϵ). Using coarser bins may reduce this error by decreasing b.

3.3 .10.2 Mean Squared Error of Cumulative Histogram Queries

The MSE for cumulative histogram queries under G_S^{attr} and G_S^{full} are:

$$\mathcal{E}_{\mathcal{M}^{\text{Lap}}_{h_{\mathcal{P}_{\text{attr}}}}}(D) = b\mathbb{E}(\text{Laplace}(2/\epsilon))^2 = 8b/\epsilon^2$$
$$\mathcal{E}_{\mathcal{M}^{\text{Lap}}_{h_{\mathcal{P}_{\text{full}}}}}(D) = b\mathbb{E}(\text{Laplace}(n/\epsilon))^2 = 2n^2b/\epsilon^2$$

The difference in MSE under G_S^{full} and G_S^{attr} is also clear for cumulative histogram queries as shown in Figure 25 and Figure 26.

3.3 .10.3 Simulating sensitivity results

In this experiment, we sample neighbors of our input graphs and compute the difference in the values of the histogram queries. We compare the obtained values with our derived sensitivity formulas. We sample the sensitivity values for our input graphs as follows:

1. take an input graph,
2. compute its histogram query H,
3. take a vertex at random,

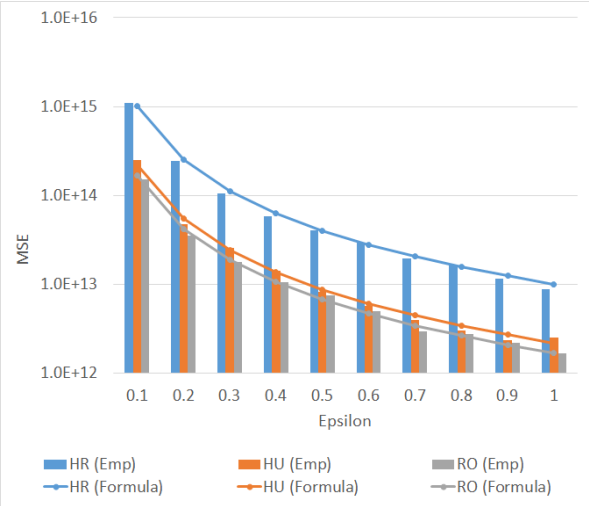


Figure 23: MSE of **complete** histogram queries under Full Policy ($k = 10$).

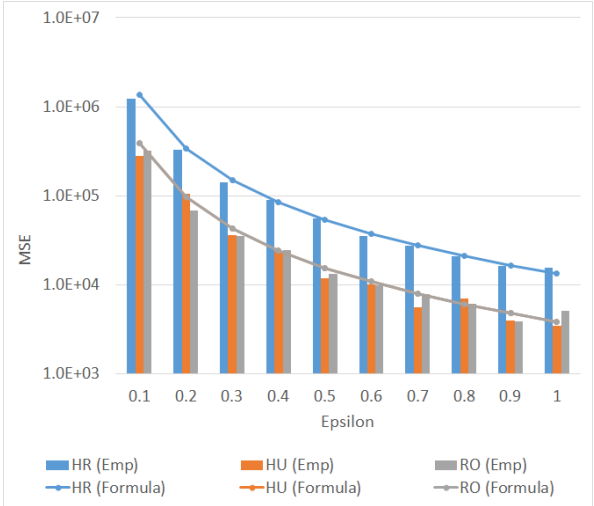


Figure 24: MSE of **complete** histogram queries under Attribute Policy ($k = 10$).

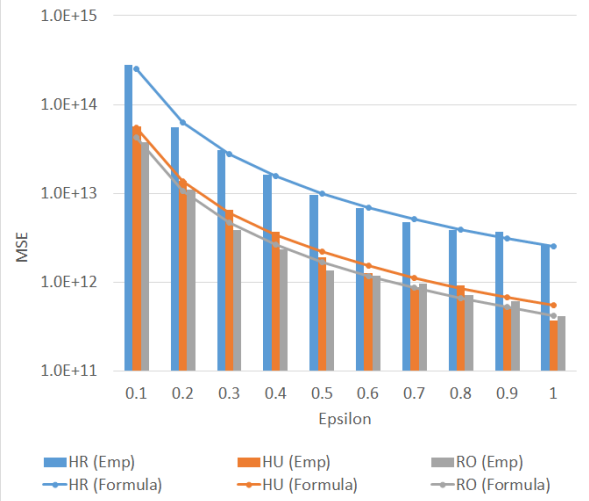


Figure 25: MSE of **cumulative** histogram queries under Full Policy ($k = 10$).

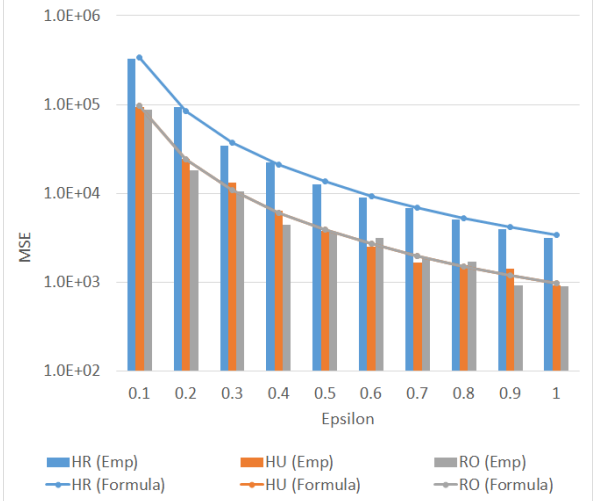


Figure 26: MSE of **cumulative** histogram queries under Attribute Policy ($k = 10$).

4. randomly change the ego network of the vertex for G_{full}, change the value of only one edge for G_{attr},

5. compute the histogram query for the obtained graph H',

6. compute the L1-norm $||H - H'||_1$,

7. repeat starting at (c),

8. take max $||H - H'||_1$ and compare to the corresponding sensitivity: $2n$ for G_{full} and just 4 for G_S^{attr},

9. repeat starting at (a),

Different strategies can be adopted to change the ego network under G_{full}:

- Take-out: The chosen vertex is taken out by removing all its connections similarly to node-based differential privacy.

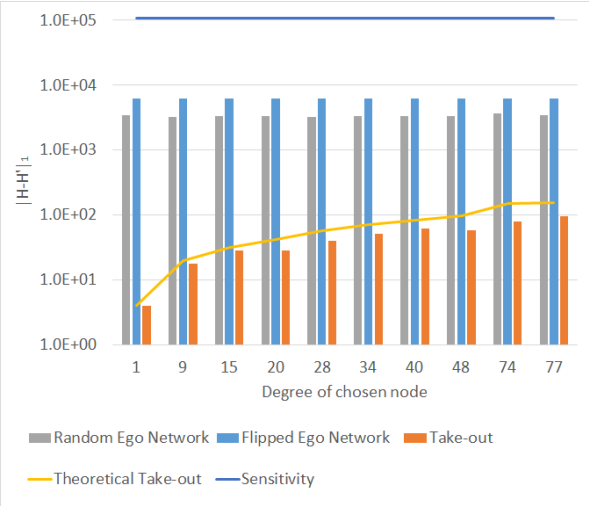


Figure 27: Simulation of **complete** histogram queries under Full Policy.

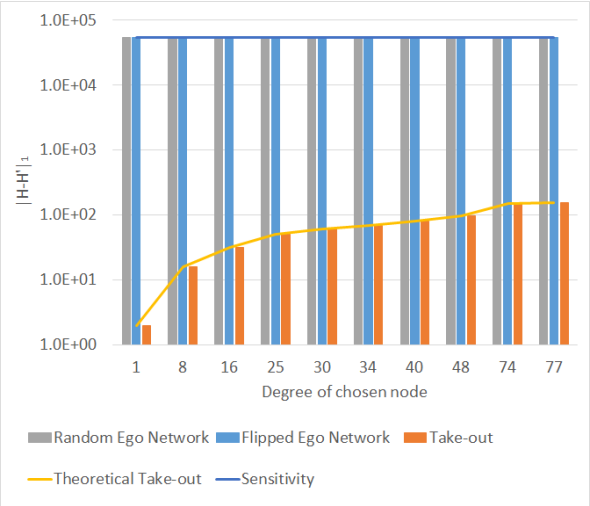


Figure 28: Simulation of **cumulative** histogram queries under Full Policy.

- Random Ego Network: The chosen vertex samples a Bernoulli distribution with predefined probability p to decide on linking to each node in the graph (we take $p = 0.5$).
- Flipped Ego Network: The chosen vertex deletes all its neighbors and connects to all non-neighbors.

We apply the above strategies to the complete histogram queries and vertices of different degrees. The results are shown in Figure 27. In addition we show the derived sensitivity formula and the worst case take-out difference $(2 \times (\deg_v + 1))$. It is clear that G_S^{full} is unreasonably pessimistic about the sensitivity of complete histograms queries. We can gain more utility by deriving more specific discriminative secret graphs (based on the strategy of ego network manipulation) or adding constraints and auxiliary knowledge to the policy. For instance, it is unreasonable that a single node would be connected to all the other nodes in the graph.

Similar experiments for the cumulative histogram queries are shown in 28. In contrast, random and flipped ego network values are very close to the derived sensitivity formula.

3.3 .10.4 Extrapolation of queries under $G_S^{\text{attr,VIP}}$

We have shown that limiting privacy to some VIP nodes provides utility for queries such as "Histogram of degrees of vertices for standard nodes" and "Histogram of the number of connections between a VIP node and standard nodes". In this experiment, we show that these queries can be exploited to estimate information about the complete graph. For instance, the histogram of degrees of vertices for standard nodes can be extrapolated to estimate the histogram of degrees of vertices for all the nodes. The histogram of the number of connections between a VIP node and standard nodes can be extrapolated to estimate the histogram of degrees of VIP nodes, and so on.

The histogram of degrees of vertices for standard nodes can be extrapolated as follows:

$$H_{\text{all nodes}}(i) = H(i)_{\text{standard}} \times \frac{100}{100 - \%_{VIP}}$$

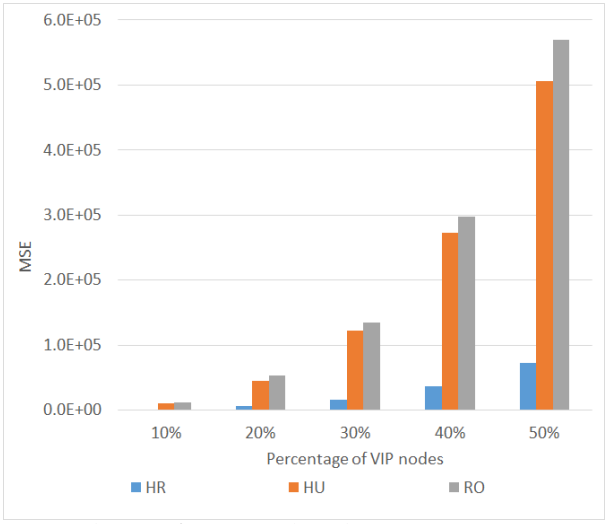


Figure 29: MSE of extrapolation for complete histogram queries under VIP/Standard partition.

The mean squared error of the extrapolated histogram query compared to the complete histogram query is shown in Figure 29. Naturally, the error depends on the ratio of the number of VIP nodes to the total number of vertices in the graph.

SUBCHAPTER 3.3.4

Blowfish Privacy for subgraphs protection in dynamic graphs

3.4 .1 Introduction

In this subpart,we present our second privacy-preserving technique for graphs dedicated for dynamic ones. Our technique consists of sequentially releasing anonymized versions of these graphs under Blowfish Privacy. To do so, we introduce a graph model that is augmented with a time dimension and sampled at discrete time steps. We have shown in the Introduction section that the direct application of state-of-the-art privacy-preserving Differential Private techniques is weak against background knowledge attacker models. We have presented different scenarios where randomizing separate releases independently is vulnerable to correlation attacks. Our method is inspired by Differential Privacy (DP) and its extension Blowfish Privacy (BP). To validate it, we show its effectiveness as well as its utility by experimental simulations.

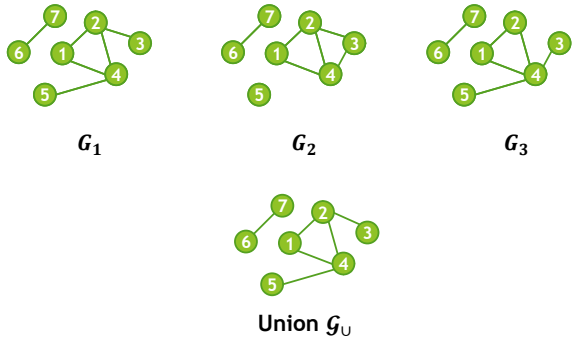
3.4 .2 Preliminary Definitions

Definition 1. *A simple graph G is undirected and defined in a given time frame by a set of vertices and a set of edges:* $G(V, E, [t_i, t_{i+1}])$ *where:*

- V *is the set of vertices* $V \subset \mathcal{V}$ *representing all the users in the time frame* $[t_i, t_{i+1}]$ *(While* $\mathcal{V}$ *represents all the vertices in all the versions).*
- E *is the set of edges* $E \subset V \times V$ *representing interactions between the users in V during the time frame* $[t_i, t_{i+1}]$*. We consider only two states: interaction and no interaction. An edge is assigned to two vertices (representing two individuals) if they are interacting during the* i^{th} *time frame.*
- $[t_i, t_{i+1}]$ *is the* i^{th} *time interval* $(i = 0, 1, ..., N)$

The definition can be used for the anonymized graph $G^*(V^*, E^*, [t_i, t_{i+1}])$. ■

For example, the first graph in Figure 30a is a simple undirected graph with a set of seven vertices, a set of six edges, and a time interval defined as Day 1.



(a) Three simple undirected graphs (G_1, G_2, G_3) and their respective union G_U.

Sampled subgraphs	G_1	G_2	G_3
	1	1	0
	1	1	1
	0	1	1
	1	0	1

(b) A logical matrix $\mathcal{M}$.

Figure 30: An example of the Union graph $\mathcal{G}_\cup$ and logical matrix.

Definition 2. *Let $\mathcal{G}$ be a set of graphs to be released in distinct time frames, $\mathcal{G} = \{G_1, ..., G_n\}$. Let SG be a subgraph of $\mathcal{G}$ where $\exists G \in \mathcal{G}$ such that $SG \subseteq G$. We define the logical matrix $\mathcal{M}$ whose row and column indices indicate the elements of SG and $\mathcal{G}$. $\mathcal{M}[SG_i][G_j] = \mathcal{M}_{ij}$ represents the status of subgraph SG_i in the graph G_j, where*

$$\mathcal{M}_{ij} = \begin{cases} 0 & \text{if } SG_i \nsubseteq G_j \\ 1 & \text{if } SG_i \subseteq G_j \end{cases} \tag{1}$$

In what follows, we will use $\mathcal{M}_{ij}$ to denote $\mathcal{M}[SG_i][G_j]$. ■

Several subgraphs were sampled from the Union graph in Figure 30a and were represented as the rows of the matrix in Figure 30b. The columns of the matrix represent the graphs of the three days, and each element represents the existence of the subgraph in the graph.

The notations in this paper are summarized in Table 7.

Symbol	Description	Symbol	Description
$\mathcal{T}$	Domain of all possible tuples	G_S	Discriminative secret graph
$\mathcal{G}_\cup$	Union graph of the set of graphs $\mathcal{G}$	G	Undirected simple graph
V	Set of nodes in G	E	Set of edges in G
$[t_i, t_{i+1}]$	i^{th} time interval	G^*	Anonymized version of G
$\mathcal{G}$	Set of graphs to be released	$\mathcal{M}^{n\times m}$	Logical matrix of size $n \times m$
$\mathcal{M}^*$	Anonymized version of $\mathcal{M}$	SG	Sampled subgraph from $\mathcal{G}$
f	Operation on logical matrix $\mathcal{M}$	S	Range of outputs
p, q	Probabilities	g	Operation on a set of graphs $\mathcal{G}$
δ'	Rate of subgraph in $\mathcal{G}$ that does not reflect their values in $\mathcal{M}^*$	$\mathcal{I}_Q$	Set of databases that are possible under the constraints Q

Table 7: Notations and descriptions

1	1	0		1	1	0		1	1	0		1	1	0
1	1	1		1	1	1		0	1	1		1	1	0
0	1	0		0	1	0		0	0	1		0	1	0
1	0	1		1	1	1		1	1	1		1	0	1
M				M'				M''				M'''		

Figure 31: Matrix $\mathcal{M}$ and three of its possible neighbors.

3.4 .3 BP Mechanism For Sequentially Releasing Graph Datasets

DP provides reliable privacy in a single graph [35, 77]. However, it has severe limitations when graphs are sequentially released [81] in different time frames. The problem appears when the Background Knowledge can be linked to correlated or sensitive subgraphs in $\mathcal{G}$. Hence, the adversary will tie his/her background knowledge to a complete subgraph instead of separated edges or vertices in $\mathcal{G}$. Here, we propose a new layer of privacy-preserving approach. It is neither edge-DP nor node-DP, considering that we are not protecting separated edges or vertices but a combination of them in the form of subgraphs. We propose a robust privacy-preserving operation on the logical matrix $\mathcal{M}$ represented by the graphs in $\mathcal{G}$ and the correlated (or sensitive) sampled subgraphs. In this way, we can determine the noise as many subgraphs are suppressed or added from/to the graphs in $\mathcal{G}$.

Before we elaborate more about our privacy-preserving mechanism, we define first how two logical matrices $\mathcal{M}$ and $\mathcal{M}'$ can be considered as neighbors.

Definition 3 (Neighboring Matrices)**.** *We say that two matrices $\mathcal{M}$ and $\mathcal{M}'$ are neighbors if:*

1. *$\mathcal{M}, \mathcal{M}' \in \mathcal{I}_Q$, which means that both respect the constraints mentioned in Subsection 2.2.*
2. *$\mathcal{M}, \mathcal{M}'$ differ by just one binary element: $\exists! i \in \{0, ..., n\}, j \in \{0, ..., m\} \mid \mathcal{M}_{ij} \neq \mathcal{M}'_{ij}$ where n is the number of graphs in the dataset and m is the number of sampled subgraphs.*■

For example, in Figure 31, we check the neighboring between $\mathcal{M}$ and the three other matrices. If we have the following constraint Q that "No column in any published matrix could be formed by just 1s." $\mathcal{M}$ and $\mathcal{M}'$ could not be neighbors because the second column of $\mathcal{M}'$ violates the constraint, then $\mathcal{M}' \notin \mathcal{I}_Q$. $\mathcal{M}$ and $\mathcal{M}''$ are not neighbors because they differ by more than one element, while $\mathcal{M}$ and $\mathcal{M}'''$ are neighbors because both of them respects the constraint and they differ by just one element.

Figure 32 shows the input and the output of the privacy-preserving operation applied to the logical matrix $\mathcal{M}$.

Lemma 1. *Let f be an operation on the logical matrix $\mathcal{M}$, we say that $f(M)$ is Differential Private (or f is ϵ-DP) if it respects the inequation:*

$$Pr[f(\mathcal{M}) = S] \leq e^{\epsilon} \times Pr[f(\mathcal{M}') = S]$$

Sampled subgraphs	G_1	G_2	G_3		G_1^*	G_2^*	G_3^*
	1	1	0		1	1	1
	1	1	1		1	1	1
	0	1	1		0	0	1
	1	0	1		1	1	0

Figure 32: An anonymization operation applied on a logical matrix results a noisy logical matrix.

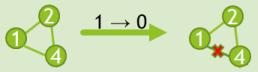
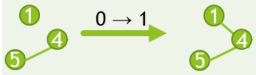
Flips	Final Results	Flips Statuses
		Cancelled
		Kept

Figure 33: Two flips performed by g where the first flip was cancelled by the second one.

where $\mathcal{M}$ and $\mathcal{M}'$ are two neighbors matrices, S is one possible output of the operation f, ϵ is a privacy parameter.
The output of $f(\mathcal{M})$ is a matrix $\mathcal{M}^*$ related to $\mathcal{M}$ by the equation

$$f(\mathcal{M}_{ij}) = \mathcal{M}^*_{ij} = \begin{cases} \overline{\mathcal{M}_{ij}} & \text{with probability } q = \frac{1}{e^\epsilon+1} \\ \mathcal{M}_{ij} & \text{with probability } p = 1-q = \frac{e^\epsilon}{e^\epsilon+1} \end{cases} \tag{1}$$

where $\overline{\mathcal{M}_{ij}}$ is the opposed binary value of $\mathcal{M}_{ij}$ and $p > q$.

Proof. Let's say that we have two neighboring matrices $\mathcal{M}$ and $\mathcal{M}'$ that differ by only one element $\mathcal{M}_{ij} \neq \mathcal{M}'_{ij}$. Let $\mathcal{S}$ be an output matrix of the operation f.

$$\begin{aligned} \frac{Pr[f(\mathcal{M}) = \mathcal{S}]}{Pr[f(\mathcal{M}') = \mathcal{S}]} &= \frac{Pr[\mathcal{M}_{11} \to \mathcal{S}_{11}] \times ... \times Pr[\mathcal{M}_{nm} \to \mathcal{S}_{nm}]}{Pr[\mathcal{M}'_{11} \to \mathcal{S}_{11}] \times ... \times Pr[\mathcal{M}'_{nm} \to \mathcal{S}_{nm}]} \\ &= \frac{Pr[\mathcal{M}_{ij} \to \mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij} \to \mathcal{S}_{ij}]} \end{aligned} \tag{2}$$

Then, to prove that f is ϵ-DP using the inequation $\frac{Pr[f(\mathcal{M})=\mathcal{S}]}{Pr[f(\mathcal{M}')=\mathcal{S}]} \leq e^\epsilon$, we have to prove that $\frac{Pr[\mathcal{M}_{ij}\to\mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij}\to\mathcal{S}_{ij}]} \leq e^\epsilon$. Each of $\mathcal{M}_{ij}$, $\mathcal{M}'_{ij}$ and $\mathcal{S}_{ij}$ can have two values 0 and 1. In case, $\mathcal{M}_{ij} = \mathcal{M}'_{ij}$ then $\frac{Pr[\mathcal{M}_{ij}\to\mathcal{S}_{ij}]}{Pr[\mathcal{M}'_{ij}\to\mathcal{S}_{ij}]} = 1 \leq e^\epsilon$. The four other cases to compute are:

- $\frac{Pr[1\to1]}{Pr[0\to1]} = \frac{p}{q} = \frac{\frac{e^\epsilon}{1+e^\epsilon}}{\frac{1}{1+e^\epsilon}} = e^\epsilon$
- $\frac{Pr[0\to1]}{Pr[1\to1]} = \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[1\to0]}{Pr[0\to0]} = \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[0\to0]}{Pr[1\to0]} = \frac{p}{q} = e^\epsilon$

In the four cases, the operation respects the inequation, which proves that f is ϵ-DP. □

3.4 .3.1 Achieving the mechanism *f* in $\mathcal{G}$

This section discusses how to populate the binary flips of the values in the logical matrix in their corresponding set of graphs $\mathcal{G}$. We also evaluate how these flips may affect the privacy of

our proposed mechanism. In the implementation, these flips are achieved by suppressing or adding edges in $\mathcal{G}$. We notice that this might affect the ability of the mechanism to achieve DP. The flips made by the operation f have two types:

- $0 \longrightarrow 1$: here, the element in $\mathcal{M}$ is 0 (which means that the subgraph does not exist in the corresponding original graph) but has been flipped to 1 in $\mathcal{M}^*$ (the subgraph exists in the correspondent anonymized graph). All the missed edges of the subgraph should be added to perform this modification in the anonymized graph.
- $1 \longrightarrow 0$: here, the element in $\mathcal{M}$ is 1 (which means that the subgraph exists in the corresponding original graph) but has been flipped to 0 in $\mathcal{M}^*$ (the subgraph does not exist in the correspondent anonymized graph). One or more edges should be deleted from the subgraph to perform this modification in the anonymized graph.

Let g be the operation that implements the flips made by $f(\mathcal{M})$ in $\mathcal{G}$. g cannot guarantee a full projection of the flips in $\mathcal{G}$. A possible scenario to explain this drawback in privacy is when the same edge should remain in a subgraph while it should be deleted in another subgraph in the same graph. Then, it is obvious that it is not possible to implement in the graphs all the modifications done on the matrix. In Figure 33 for example, to perform the first flip, the operation g deletes the edge $(1,4)$. For the second flip, the operation adds the two edges $(1,4)$ and $(1,5)$; thus, the second flip cancels the effect of the first flip, and then the final status of the subgraph in the second flip is similar to its status before the flip. In this way, the privacy guarantee provided by f could be reduced when applying g on the graphs. In other words, after releasing $\mathcal{G}$, an adversary may create a matrix $\mathcal{M}_{adv}$ that might represent the real status of the subgraphs more accurately than $\mathcal{M}^*$. That is the effect of g on the privacy guarantee. Thus, the implementation cannot provide the desired privacy guarantee provided by the operation of f. Still, it can offer a relaxed guarantee called (ϵ, δ)- BP, where ϵ is a privacy parameter provided by the data owner and δ is the relaxation parameter of the privacy definition.

In the next subsection, we will discuss in detail the privacy guarantee. However, this discussion cannot be reliable if we do not define the implementation steps first. As we have explained, the purpose of the implementation is to populate the flips in the graphs. We can list four types of these flips:

- $0 \longrightarrow 0$: an element 0 in the matrix $\mathcal{M}$ remains 0 in the noisy matrix $\mathcal{M}^*$.
- $0 \longrightarrow 1$: an element 0 in the matrix $\mathcal{M}$ is flipped to 1 in the noisy matrix $\mathcal{M}^*$.
- $1 \longrightarrow 0$: an element 1 in the matrix $\mathcal{M}$ is flipped to 0 in the noisy matrix $\mathcal{M}^*$.
- $1 \longrightarrow 1$: an element 1 in the matrix $\mathcal{M}$ remains 1 in the noisy matrix $\mathcal{M}^*$.

Because the implementation of a flip can cancel the effects of some previous implementations, we can assume that the ones performed initially have a higher probability of being canceled than those performed at the end. A random implementation of the flips will make impossible to compute the level of privacy provided by the implementation and to define the

relaxation parameter δ. Therefore, two possible orders can be applied. If we are interested in preserving the subgraphs that should exist in $\mathcal{G}$, the steps are as follows:

1. Remove at least one edge from each subgraph represented by 1 in $\mathcal{M}$ and 0 in $\mathcal{M}^*$.
2. Ensure that all the subgraphs represented by 1 in $\mathcal{M}$ and $\mathcal{M}^*$ remain in the graphs after Step (1). In other words, if any edge removed in Step (1) leads to the removal of a subgraph represented by 1 in the matrix, this edge should be re-added.
3. Add all the missing edges for the subgraphs represented by 0 in $\mathcal{M}$ and 1 in $\mathcal{M}^*$.

In this way, we guarantee that all the subgraphs represented by 1 in the matrix $\mathcal{M}^*$ will exist in the released version of $\mathcal{G}$. Also, we may find in the set some subgraphs that should not exist. Whilst, if we are interested in guaranteeing that all the subgraphs represented by 0 in $\mathcal{M}^*$ do not exist in the released version of $\mathcal{G}$, then the order need to be reversed as follows:

1. Add all the missing edges for the subgraphs represented by 0 in $\mathcal{M}$ and 1 in $\mathcal{M}^*$.
2. Ensure that all the subgraphs represented by 0 in $\mathcal{M}$ and $\mathcal{M}^*$ do not exist in the graphs after Step (1).
3. Remove one or more edges from each subgraph represented by 1 in $\mathcal{M}$ and 0 in $\mathcal{M}^*$.

By following this order, the set is clean from all undesirable subgraphs, but it may also miss some subgraphs that should appear in $\mathcal{G}$.

3.4 .3.2 Toward a BP mechanism

After explaining the operation f and its reliable privacy guarantee as well as the drawbacks of implementing g in the privacy domain, we will determine the implementation's privacy guarantee in this subsection.

Lemma 2. *Let δ' be the rate of subgraphs in $\mathcal{G}$ that does not reflect their values in $\mathcal{M}^*$. Let $\mathcal{G}$, $\mathcal{G}'$ and $\mathcal{S}$ be three sets of graphs where $\mathcal{G}$ and $\mathcal{G}'$ are two neighboring sets that differ in just one sampled subgraph and $\mathcal{S}$ is a possible output of g applied on $\mathcal{G}$ and $\mathcal{G}'$. g respects the inequation*

$$Pr[g(\mathcal{G}) = S] \leq e^{\epsilon}.Pr[g(\mathcal{G}') = S] + \delta \tag{3}$$

if

$$\delta' \leq \frac{\delta}{e^{\epsilon} - 1} \tag{4}$$

Proof. To prove the Lemma 2, we have to prove that:

$$\frac{Pr[g(\mathcal{G}) = \mathcal{S}] - \delta}{Pr[g(\mathcal{G}') = \mathcal{S}]} \leq e^{\epsilon} \tag{5}$$

The left-hand side part of the inequation can be written in the following way:

$$\frac{Pr[g(\mathcal{M}_{11}) = \mathcal{S}_{11}] \times ... \times Pr[g(\mathcal{M}_{nm}) = \mathcal{S}_{nm}] - \delta}{Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times ... \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}]}$$

Let SG_{dif} be the only subgraph that differs between $\mathcal{G}$ and $\mathcal{G}'$. The status of SG_{dif} in $\mathcal{G}, \mathcal{G}'$ and $\mathcal{S}$ can be indicated by the binary values $\mathcal{M}_{ij}, \mathcal{M}'_{ij}$ and $\mathcal{S}_{ij}$. Then, the above fraction can be reduced to:

$$\frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times ... \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}]}$$

All the probabilities are less than or equal to 1. Thus:

$$Pr[g(\mathcal{M}'_{11}) = \mathcal{S}_{11}] \times ... \times Pr[g(\mathcal{M}'_{nm}) = \mathcal{S}_{nm}] \leq Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]$$

Consequently, we can assume that

$$\frac{Pr[g(\mathcal{M}_{ij})=\mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij})=\mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{11})=\mathcal{S}_{11}]\times...\times Pr[g(\mathcal{M}'_{nm})=\mathcal{S}_{nm}]} \leq$$

$$\frac{Pr[g(\mathcal{M}_{ij})=\mathcal{S}_{ij}]}{Pr[g(\mathcal{M}'_{ij})=\mathcal{S}_{ij}]} - \frac{\delta}{Pr[g(\mathcal{M}'_{ij})=\mathcal{S}_{ij}]}$$

which leads to

$$\frac{Pr[g(\mathcal{G}) = \mathcal{S}] - \delta}{Pr[g(\mathcal{G}') = \mathcal{S}]} \leq \frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}] - \delta}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} \leq e^{\epsilon}$$

Therefore, to prove the Inequation 5, it is enough to prove that

$$\frac{Pr[g(\mathcal{M}_{ij}) = \mathcal{S}_{ij}] - \delta}{Pr[g(\mathcal{M}'_{ij}) = \mathcal{S}_{ij}]} \leq e^{\epsilon} \tag{6}$$

To continue our proof, we have to find all possible values of these probabilities. In this step of the proof, we can see the importance of the orders in the second stage. We will continue this proof based on the first order listed, but the second order will also lead to the same final result. We can list four possible values to the probabilities in Equation 6:

- Case 1: $Pr[1 \to 0]$ is the probability that f flips the binary value in the matrix. Nevertheless, because these flips are performed at first by the operation g, based on the first order, then there is a probability δ' that the following flip will cancel the current one, thus: $Pr[1 \to 0] = q - \delta'$.

- Case 2: $Pr[0 \to 0]$ is the probability that f doesn't flip the value. But some following flips performed by g may cause the appearance of the subgraph in $\mathcal{G}$, thus: $Pr[0 \to 0] = p - \delta'$.

- Case 3: $Pr[1 \to 1]$ is the probability that f doesn't flip the value. Based on the order of flips in g, all the upcoming flips will be performed by adding edges so that no flip can cause the drop of this subgraph from $\mathcal{G}$, thus: $Pr[1 \to 1] = p$.

- Case 4: $Pr[0 \to 1]$ is the probability that f flips the binary value in the matrix. In this case too, no further flip could drop this subgraph, thus: $Pr[0 \to 1] = q$.

In this way, we have computed the four possible values for the probability. The next step is to show that by using any two of these values in Equation 6, the result remains lesser than e^ϵ:

- $\frac{Pr[1\to1]-\delta}{Pr[0\to1]} = \frac{p-\delta}{q} \leq \frac{p}{q} = e^\epsilon$
- $\frac{Pr[0\to1]-\delta}{Pr[1\to1]} = \frac{q-\delta}{p} \leq \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)
- $\frac{Pr[1\to0]-\delta}{Pr[0\to0]} = \frac{q-\delta'-\delta}{p-\delta'} \leq \frac{q}{p}$ if:

$$\begin{aligned} p(q-\delta'-\delta) &\leq q(p-\delta') \\ p\delta' + p\delta &\geq q\delta' \\ \delta &\geq \delta' \times \frac{q-p}{p} \end{aligned} \tag{7}$$

  $q - p \leq 0$ and $\delta \geq 0$, then the Inequation 7 is always true, thus:
  $\frac{Pr[1\to0]-\delta}{Pr[0\to0]} \leq \frac{q}{p} = \frac{1}{e^\epsilon} \leq e^\epsilon$ (because $e^\epsilon \geq 1$)

- $\frac{Pr[0\to0]-\delta}{Pr[1\to0]} = \frac{p-\delta'-\delta}{q-\delta'} \leq \frac{p}{q}$ if:

$$\begin{aligned} q(p-\delta'-\delta) &\leq p(q-\delta') \\ q\delta' + q\delta &\geq p\delta' \\ \delta &\geq \delta' \times \frac{p-q}{q} \end{aligned} \tag{8}$$

  Then, the only inequation that should be verified to ensure that $\mathcal{G}$ can be released under (ϵ, δ)-BP is: $\delta' \leq \delta \times \frac{q}{p-q}$. By substituting p and q by their values, we get:

$$\delta' \leq \frac{\delta}{e^\epsilon - 1}$$

□

Theorem 3. *Let g be the operation that implements the flips made by $f(\mathcal{M})$ in $\mathcal{G}$, we say that g is (ϵ, δ)-BP if there exists a policy $P = (\mathcal{T}, G_S, \mathcal{I}_Q)$ that applies on $\mathcal{G}$ and composed of the Universe $\mathcal{T}$, the discriminative secret graph G_S and $\mathcal{I}_Q$ that denotes all the possible $\mathcal{G}$ under the constraints Q.*

Proof. $\mathcal{T}$ contains all the possible graphs from the anonymized version of the set $\mathcal{G}$. $G_S = (V_S, E_S)$ is a discriminative secret graph where $V_S = \mathcal{T}$ and $e_S \in E_S$ connects two vertices that should be indistinguishable for the public. In this work, e_S will connect any two vertices of G_S that represent two graphs that differ by just one subgraph sampled in our mechanism. Two graphs that differ by one or more edges but resemble in all the sampled subgraphs do not form a discriminative pair. They might be distinguishable, and the two vertices representing them in G_S are not connected by an edge e_S. Therefore, the discriminative secret graph G_S is not complete. In other words, E_S is the set of edges that connects two vertices representing a difference in the status of one or more sampled subgraphs. By proving that our mechanism's

discriminative secret graph is not complete, we have demonstrated that our mechanism does not respect the DP definition requirements.

We can assume that our mechanism is under the Attribute version of BP. The Attribute Definition in BP describes a mechanism that aims to hide any changes made to any attribute in a set of tuples, which is literally what we aim to do by protecting any modifications done to the element of the matrix $\mathcal{M}$ by manipulating the existence and non-existence of its corresponding subgraph in the released set of graphs.

Finally, the constraint Q is related to the Inequation 4: $\delta' \leq \frac{\delta}{e^{\epsilon}-1}$ beside any constraint provided by the data owner. Therefore, if $\mathcal{G}$ respects this inequation, then $\mathcal{G} \subset \mathcal{I}_Q$.■ □

By proving that g respects:

- the BP Inequation in Equation 3,
- a well defined Policy P,

we have proved that the mechanism is (ϵ, δ)-BP.

3.4 .4 Anonymization Algorithm

This section details our mechanism by applying the Blowfish Privacy for sequentially releasing graphs discussed in the previous section.

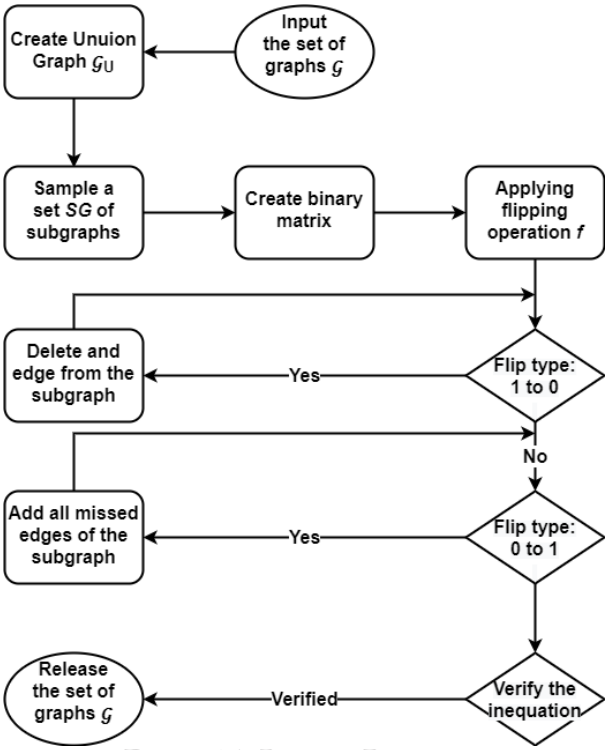


Figure 34: Process Diagram.

The process is depicted in Figure 34. To note that we should possess all the graphs before starting the anonymization process. To proceed, we sample a number of subgraphs from a Union Graph $\mathcal{G}_\cup$, based on some criteria. For example, in our implementations, we will focus

on the high correlated subgraphs. These are the subgraphs that exist in the most graphs in the set of graphs $\mathcal{G}$: a subgraph that exists in 90% of graphs has a much higher possibility of being sampled than a subgraph that exists in 50% of the graphs. Based on these sampled subgraphs, we create a matrix $\mathcal{M}$, as in Figure 30b, with binary attributes showing the status of each subgraph in each graph. Then, we apply the flipping probability to the values of the attributes. In this section, we will adapt the first order of implementation explained in Subsection 3.4 .3.1; for this reason, all the $1 \longrightarrow 0$ flips should be performed first by suppressing the edge that occurs the least in the sampled subgraphs to minimize the effect of this flip on other subgraphs. When all the $1 \longrightarrow 0$ flips are performed, the $0 \longrightarrow 1$ flips are applied by adding all the subgraph's missing edges. If all the required flips are completed, we can start to release the noisy graphs.

In summary, our proposed mechanism is composed of these steps:

1. Sampling a number of subgraphs of K vertices.

2. Creating a matrix that represents the existence of the sampled subgraphs in each graph of the set.

3. Applying the operation f by anonymizing the matrix under the requirements of Differential Privacy.

4. Applying the operation g by adding/suppressing edges in the set of graphs.

In the following subsection, we will propose an algorithm to apply the operation and the implementation.

3.4 .4.1 Applying The Algorithm

The list of sampled subgraphs is converted into a matrix $\mathcal{M}$, as explained in the Definition 2. $\mathcal{M}$ will be subject to the flipping operation f, as in Figure 32 where the binary output of the algorithm relies on the probabilities p and q shown in Lemma 1.

The main algorithm takes as input the original set of graphs $\mathcal{G}$, a number of subgraphs that we estimate representing the Background Knowledge or Public Knowledge of this set, and the probability p. The output is the noisy version of the set $\mathcal{G}^*$. The algorithm adapts the first order explained in Subsection 3.4 .3.1, which focuses on well representing the 1s of the matrix $\mathcal{M}^*$ in the released graphs. In step 1, we sample a number of subgraphs based on the data provided by the Union Graph $\mathcal{G}_\cup$ and the set of Background Knowledge subgraphs. Each edge in $\mathcal{G}_\cup$ contains data concerning the graphs where this edge exists. In the second step, the logical matrix $\mathcal{M}^{n\times m}$ is created where each row represents a subgraph, and each column represents a graph of $\mathcal{G}$ and each binary value $\mathcal{M}_{ij}$ represents the existence of subgraph SG_i in graph G_j.

In steps 3-7, the algorithm creates the noisy matrix $\mathcal{M}^*$ by applying the function Flip. This function takes a binary value and a probability of p as input; it generates a random number of *rnd* between 0 and 1; if *rnd* is higher than p, then the binary value is flipped.

Algorithm 1: Main Algorithm

Require: $\mathcal{G}$, $\{SG_{BK}\}$, p
Ensure: $\mathcal{G}^*$
$\{SG_1, ..., SG_n\} \leftarrow SampleSG(\mathcal{G}_\cup, \{SG_{BK}\})$
$\mathcal{M}^{n\times m} \leftarrow BuildLogicalMatrix(\mathcal{G}, \{SG\})$
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    $\mathcal{M}^*_{ij} \leftarrow Flip(\mathcal{M}_{ij}, p)$
  end for
end for
$\mathcal{G}^* = \mathcal{G}$
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    if $\mathcal{M}_{ij} == 1$ & $\mathcal{M}^*_{ij} == 0$ **then**
      $G^*_i \leftarrow DropSG(SG_i, G^*_j)$
    end if
  end for
end for
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    if $\mathcal{M}^*_{ij} == 1$ **then**
      $G^*_i \leftarrow AddSG(SG_i, G^*_j)$
    end if
  end for
end for

```
1:  function FLIP(val, p)
2:      rnd := random(0, 1)
3:      if rnd ≤ p then
4:          val* := val
5:      else
6:          if val :== 0 then
7:              val* := 1
8:          else
9:              val* := 0
10:         end if
11:     end if
          return val*
12: end function
```

In steps 9-15, the algorithm drops all the subgraphs that should be deleted based on the flipping process. While in steps 16- 22, the algorithm adds the corresponding subgraphs, even if these subgraphs are already in the original graph because steps 9-15 might have caused the drop of some of these subgraphs unintentionally.

AddSG searches for all the edges that should be in the subgraph SG_i and adds all of these edges that do not exist in the graph G^*_j. While the *DropSG* function sorts all the edges of SG_i, then deletes the one having the least weight in the Union Graph $\mathcal{G}_\cup$. The edges' weight is calculated while creating the Union Graph based on the required type of the subgraphs' sampling, for example, sampling the high correlated subgraphs or the least

correlated subgraphs.

In is important to note that before releasing, we have to compute $\delta' = \frac{FaultSg}{n \times m}$ where $FaultSg$ is the number of subgraphs in $\mathcal{G}$ that doesn't reflex their binary values in $\mathcal{M}^*$ and $n \times m$ is the number of elements in $\mathcal{M}^*$.

Finally, we have to verify the Inequation 4. In case the inequation is verified, then it is safe to release the graphs under the guarantee of (ϵ, δ)-BP. Otherwise, the mechanism must be re-executed, or the data owner may decide to increase the relaxation parameter δ.

3.4 .4.2 Discussion

In this subsection, we will discuss three issues. The first one is: Is it possible to have a different result by re-executing the mechanism without changing any of the inputs ϵ, δ, and K? Actually, yes, it is possible. Any BP or DP algorithms are called a randomized algorithm which means. They employ a degree of randomness as part of their logic. In our case, the two probabilities p and q are concerned with this randomness. The value of these probabilities will remain the same as the ϵ is not changed. However, even if the probabilities and the original matrix did not change, each value in this matrix is subjected to a flip with a probability of q. The group of subgraphs that should be flipped and the rate of subgraphs in this group that will not be flipped practically because of other flips, as explained in Subsection 3.4 .3.1 and especially in Figure 33, cannot be controlled by the mechanism. Therefore, if δ' is not respecting the Inequation 4, then maybe a re-execution will fix the problem. However, if many re-executions did not help, this is an indicator that the relaxation parameter δ provided by the data owner is very small for this dataset and should be increased.

The second issue is about how n the number of sampled subgraphs may impact the result. Any increase in the number of sampled subgraphs means more manipulation in the edges. That is because the number of subgraphs that should be added or removed will increase, leading to a big increase in the added edges and a smaller increase in the removed edges, especially if we are removing just one edge for each subgraph. Overall, the rise in the value of n will lead to more privacy and less utility.

The third issue is why we have chosen to remove just one edge from each subgraph that should be dropped. In Figure 33, we have explained how deleting one edge may effect a subgraph that should be added but does not exist in the final result. When deleting two or more edges from each subgraph, the possibility of this impact increases, so the rate δ'. Hence, the possibility that the Inequation 4 is not respected becomes higher, forcing the data owner to re-execute the mechanism or increase the relaxation parameter δ.

function ADDSG(SG_i, G_j^*)
  for each edge e in SG_i **do**
    if e not in G_j^* **then**
      $add(e)$ to G_j^*
    end if
  end for
  return G_j^*
end function

function DROPSG(SG_i, G_j^*)
  Sort all edges of SG_i based on their weights in $\mathcal{G}_\cup$
  $\#\mathcal{G}_\cup = G_1 \cup G_2 \cup ... \cup G_n$
  e_{min} := edge of SG_i with minimum weight
  $drop(e_{min})$ from G_j^*
  return G_j^*
end function

3.4 .5 Experiments

This section will present our experiments' results to evaluate our approach both in terms of utility and privacy. We implement our algorithm in Python on the "Autonomous Systems AS-733" dataset [90, 91] containing 733 releases, 6474 vertices, and 13895 edges in the largest release, and we conducted experiments on an Intel Core i7 2.4 GHz PC with 8GB RAM.

As well as our algorithm, we will apply an edge-DP algorithm called TmF [125] that adds noise to each graph without considering the correlations. We will compare the results of both algorithms based on many privacy and utility measures.

3.4 .5.1 Trade-off Between Privacy and Utility

When applying DP or BP mechanisms, one of the data owners' issues is choosing the privacy parameters to set an acceptable trade-off between privacy and utility. This subsection proposes a method for the data owners applying our mechanism and the TmF mechanism to choose ϵ.

The values we have used for the privacy parameter ϵ of our algorithm are: 0.1, 0.2, 0.5 and 1. The TmF mechanism has two privacy parameters ϵ_1 and ϵ_2. In [125], they fix the value of $\epsilon_2 = 0.1$, while ϵ_1 is related to the number of nodes $\mid V \mid$: $\epsilon_1 = coef \times ln(\mid V \mid)$, where $coef = 1, 2$ and 3. In our work, we add also the value 0.5 to the values of $coef$.

In this subsection, we use the confusion matrix as a method for the data owners to choose ϵ for our BP mechanism and ϵ_1 for the TmF mechanism. We extract a number of subgraphs from each original graph and project them on original and private versions of the graph to find isomorphic subgraphs. We compare the number of subgraphs found in all the versions and compute the ratio of:

- **True Positive:** Subgraphs that appear in both original and released versions.
- **False Positive:** Subgraphs appear in the released version but do not exist in the original one.
- **True Negative:** Subgraphs that exist neither in the original graph nor in the released one.
- **False Negative:** Subgraphs that exist in the original graph but do not appear in the released one.

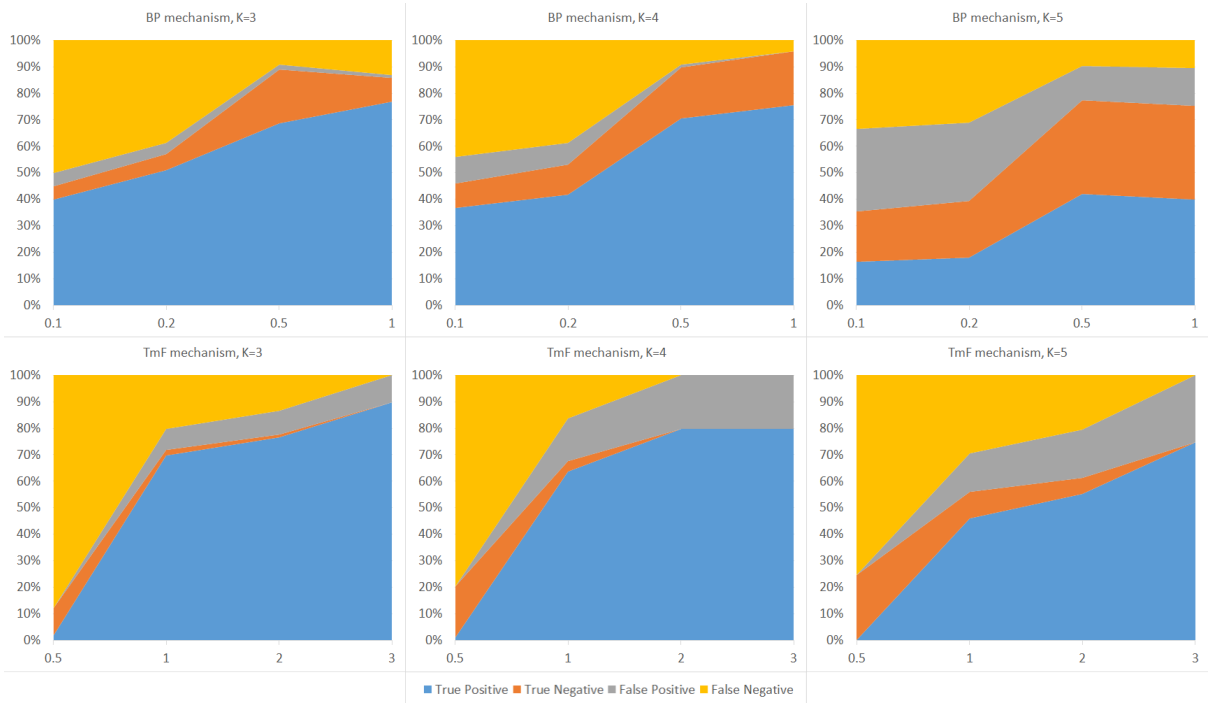


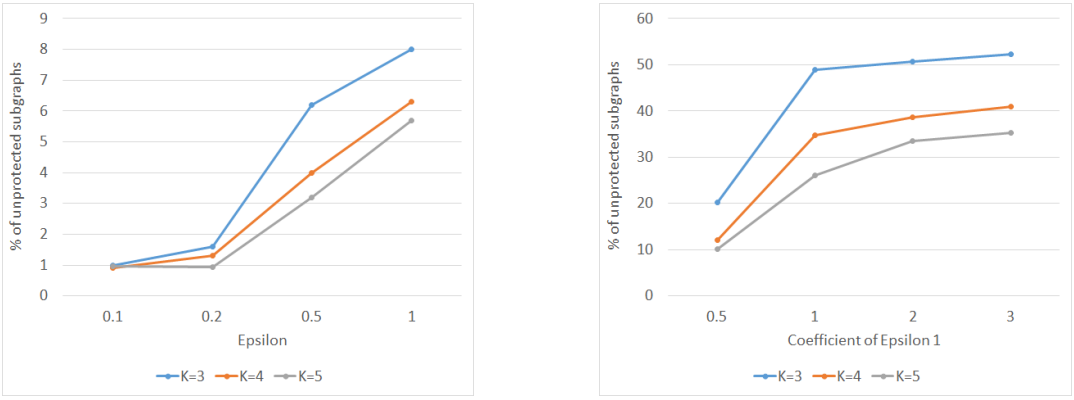
Figure 35: Confusion matrix for sampled subgraphs in every graph and its noisy versions based on K (number of vertices for each subgraph) and privacy parameter.

In Figure 45, we consider the percentage of True Positive and True Negative as the ratio of utility, while the rate of False Positive and False Negative isis the ratio of privacy. It is up to the data owners to choose the privacy parameter based on how much privacy they want in their released graphs. For the BP mechanism, increasing ϵ means more utility and less privacy. Increasing $coef$ for ϵ_1 in the TmF mechanism also leads to less privacy, but this drop in privacy is quicker than the decline in the BP mechanism. We can then say that our mechanism maintains a more balanced trade-off between privacy and utility than TmF while changing each mechanism's privacy parameters.

3.4 .5.2 Measuring Privacy

In these experiments, we are focusing on protecting the high correlated subgraphs. By high correlated, we mean the subgraphs that frequently appear in the original graphs. Therefore, we have sampled 1000 high correlated subgraphs. We consider the scenario that an adversary has generated the Intersection graph from our released graphs. Then we compare the percentage of these 1000 subgraphs that appear in the Intersection graphs generated from the released graphs under our mechanism and TmF mechanism. Higher numbers mean more high correlated subgraphs are not protected.

Figure 36 shows that the percentages of high correlated subgraphs that our mechanism failed to protect in the Intersection of the released graphs are much smaller than these of the TmF mechanism. It also shows that increasing the number of nodes in the sampled subgraphs from 3 to 5 leads to a smaller portion. The reason is that a higher number of nodes in a subgraph means a higher number of edges, which increases the possibility that one of these edges is removed in the Intersection, which is enough to consider this subgraph as protected.



(a) Our BP mechanism (b) TmF mechanism

Figure 36: Percentage of sampled high correlated subgraphs unprotected in Intersection graphs.

Also, in Figure 36, we can notice that a higher ϵ increases the percentage of failure for both algorithms and the three K values. That is reasonable because having a higher ϵ means that the value of probability p has increased. In contrast, the value of q has decreased, which leads to a decline in the number of flips. Therefore an rise in the number of unprotected subgraphs can be predictable.

3.4 .5.3 Kullback–Leibler Divergence for Union and Intersection Graphs

In this subsection, we use Kullback–Leibler (KL) Divergence to compare the distributions of degrees of nodes between the Union of the original graphs and the Union of the released graphs under our mechanism and TmF mechanism. We generate the Union of the original graphs then the Union of the released graphs. Each edge in these Unions is weighted by the number of times it appears in the graphs. For example, an edge that appears in 35 graphs, its weight in the Union is 35. We create two lists of edges $W_\cup$ containing all the edges and their weights in the original Union and $W'_\cup$ for the edges of the Union of released graphs.

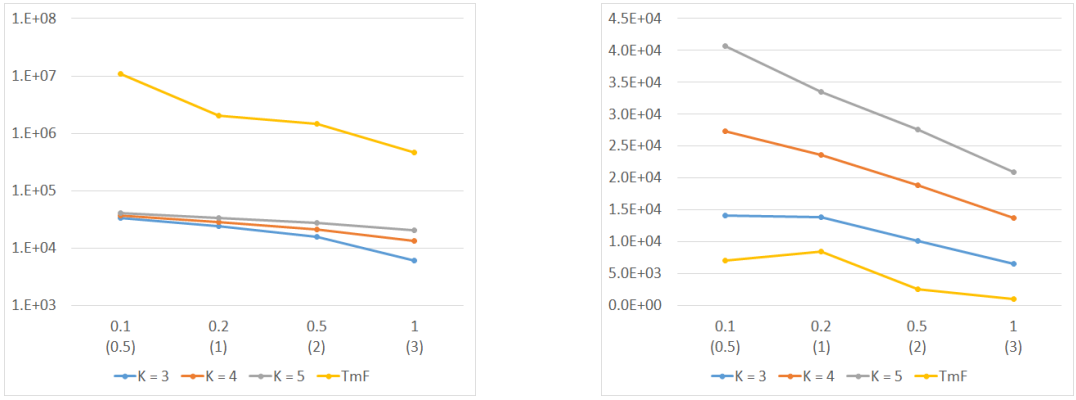
$$D_{KL}(W'_\cup \,||\, W_\cup) = \mid \sum_{e \in E_{W'_\cup}} W'_\cup[e] \times log \frac{W'_\cup[e]}{W_\cup[e]} \mid \tag{1}$$

where $E_{W'_\cup}$ are all the edges listed in list $W'_\cup$ and $W'_\cup[e]$ is the weight of edge e.

For the second KL Divergence, we generate the weighted Intersections of the original and the released graphs. Then we create two lists $W_\cap$ and $W'_\cap$ containing the edges that appear in at least one of the two Intersections. The weight of each edge in $W_\cap$ is the number of times it appears in the original graphs, while its weight in $W'_\cup$ is the number of times it appears in the released graphs.

$$D_{KL}(W'_\cap \,||\, W_\cap) = \mid \sum_{e \in E_{W'_\cap}} W'_\cap[e] \times log \frac{W'_\cap[e]}{W_\cap[e]} \mid \tag{2}$$

In Figure 37a, KL Divergence of Union graphs under the TmF mechanism is much higher than those under the BP mechanism for $K = 3, 4$, and 5. That is because TmF adds and removes a large number of edges all over each graph, and all these manipulations affect the Union graphs,



(a) Union graphs (b) Intersection graphs

Figure 37: KL Divergence for Union and Intersection graphs. (The above values in horizontal axis are the values of ϵ for our algorithm. The bottom values represent $coef$ for TmF mechanism.)

leading to this high KL Divergence for TmF. On the other hand, our mechanism focuses on manipulating the edges that create high correlated subgraphs. All other edges are not affected by our algorithm; hence, the KL Divergence under our mechanism is much smaller than the KL divergence of TmF.

In Figure 37b, we see that KL Divergence for TmF is smaller than KL Divergence for our algorithm. This chart proves the motivation for this work. As we have explained, using a DP mechanism on sequentially released graphs will cause the injection of much noise, as shown in Figure 37a. Still, most of the noise disappears when generating the released graphs' Intersection, which may lead to a serious privacy breach.

We also notice from the two charts in Figure 37 that a higher K leads to a higher KL Divergence, which means more privacy and less utility. It can be explained as a result of adding a high number of edges for each sampled subgraph that should be added to $K = 5$ nodes, for example, compared to $K = 3$.

3.4 .5.4 Centrality Measures

The majority of the studies on social networks focus on extracting the most important individuals (vertices) in the network. We will prove that the effect of our mechanism on the list of most important individuals is negligible.

To find the most important individuals, we study the centrality of the vertices, and according to the purpose of each study, we choose the type of centrality measures. Linton C. Freeman said in 1978 [59]: "There is certainly no unanimity on exactly what centrality is or on its conceptual foundations, and there is little agreement on the proper procedure for its measurement," which is still true today. However, we can define it as a measurement of the extent to which an individual interacts with other individuals in the network. We compare the 100 most important nodes of the original graphs, the BP version, and the TmF version based on four centralities:

- Degree: It is the number of vertices at a distance one for each vertex. In our case, where

the graph is a simple undirected graph, we can define the degree simply as the number of edges connected to each vertex. The degree centrality [27] of a vertex v is defined as $C_D = deg(v)$. In many social studies, people with the most connections are the most important individuals in the network.

- Farness: The farness of a node v is the sum of all shortest-paths of v to all other nodes. The closeness [128] is the reciprocal of the farness : $C(v) = \frac{1}{\sum_y d(y,v)}$ where $d(y,v)$ is the distance between vertices v and y. In this case, the most important vertices are the closest ones to all other nodes. Thus, we can rely on these individuals to spread information to all other nodes sequentially.

- Betweenness[6]: This centrality quantifies the number of times a vertex occurs on a geodesic; in other words, it is the number of times a vertex appears in the shortest path between two other nodes.

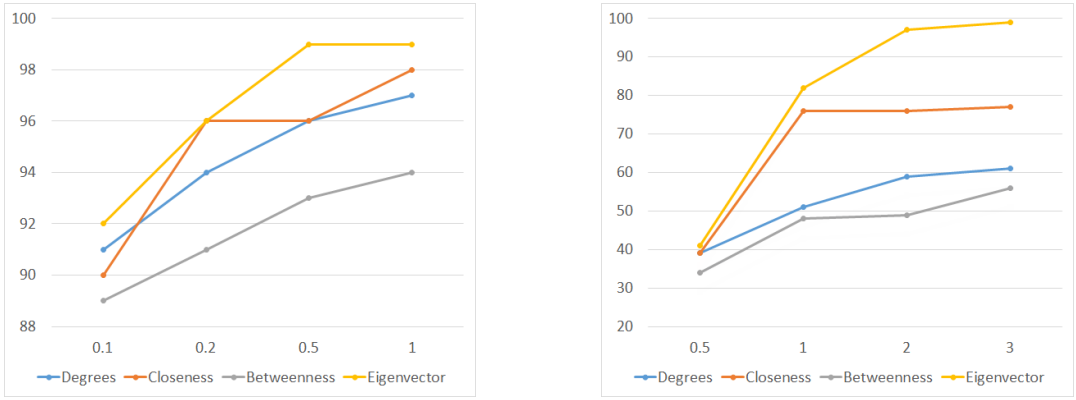
  This centrality was proposed by Freeman [58], and his idea was that actors who exist between other individuals might control the interactions between these individuals. Then, this centrality quantifies the control of a vertex on the communication of other vertices. Therefore, in this case, the most important individuals have a high probability of occurring in the shortest path of two vertices randomly chosen.

  The betweenness centrality of vertex v is defined as $C_B(v) = \sum_{i<j} \frac{sp_{ij}(v)}{sp_{ij}}$ where sp_{ij} is the number of shortest paths between vertices i and j, and $sp_{ij}(v)$ is the number of shortest paths between vertices i and j that pass through v.

- Eigenvector [26]: This centrality measures the influence of the vertex in the graph. It depends on the number and the quality of the connections. Therefore, a vertex v_1, with number of connections less than a vertex v_2, may outrank v_2 if the quality of its connections is higher. The centrality score of vertex v can be defined as: $x_v = \sum_{i \in N(v)} x_i = \sum_i A_{vi} x_i$ where $N(v)$ is a set of the neighboring nodes of v, A is the non-negative adjacency matrix of the graph.

The main idea of releasing multiple graphs sequentially instead of releasing just one graph is to improve the utility of the released data. For this reason, we compare the 100 most important nodes between each graph and its released versions to measure each graph's utility instead of just the Union and the Intersection graphs. Then we count the common nodes in these 100 nodes to determine the number of most important nodes unaffected by the anonymization mechanism.

In Figure 38, we compute the mean of the common nodes counts for all the graphs in each release. We consider this mean as a measure of utility for the set of released graphs. The number of common important nodes of our algorithm for $K = 3, 4$, and 5 are incredibly close; then, we choose to use the number of common important nodes for just $K = 4$. Figure 38a shows that our algorithm highly preserves the most important nodes. Approximately more than 90% of the most important nodes are preserved in our algorithm's released graphs. While in Figure



(a) BP algorithm (b) TmF algorithm

Figure 38: Number of common nodes for the 100 most important nodes based on four centralities.

38b, we see that the algorithm does not preserve most of the important nodes, especially for $coef = 0.5$. That changes when $coef$ increases, but in most cases, our utility is still much higher than the utility provided by TmF. We can explain that difference in utility due to focusing just on manipulating the edges related to particular subgraphs (in our experiments, the high correlated subgraphs) we aim to protect, while the TmF mechanism is manipulating edges all over the graph.

SUBCHAPTER 3.3.5

Differential Privacy for Active Attacks on Dynamic Graphs

3.5 .1 Introduction

Graphs are data that abstract a number of nodes (individuals, servers, routers,...) and the relations between them called edges. Dynamic graphs, also called multi-released or sequentially released graphs are even more valuable than static ones. A static graph shows the relation between many individuals in a community as several edges presenting the fact that, for example, at least one call has taken place between two of the individuals in a month. Instead, in a dynamic graph, we show this relation daily for a month. In other words, in this case, the dynamic graph is presenting 30 static graphs. Dynamic graphs represent more accurate data than statics', but with each progress, the challenges grow.

Static graphs already have many privacy issues that should be taken into consideration before sharing the graphs. These issues rise and become more complex in the dynamic graphs. Actually, in our previous subpart, we have treated one of these threats in dynamic graphs.

In that subpart, we have discussed the scenario of an adversary taking advantage of the multiple graphs released by creating the Intersection and the Union graph of these graphs to expose some relations between individuals that should remain confidential. We have focused on protecting the subgraphs with high occurrences or very low occurrences in the set of released graphs. We have proved that edge-Differential Privacy, a technique based on adding and removing some edges from the original graph, cannot provide the desired protection to these subgraphs.

We have proposed a three steps solution based on Blowfish Privacy, an extension of Differential Privacy. The first step required sampling of the subgraphs that we aim to protect in the set of graphs. In the second step, we create a table, every row represents one of the sampled subgraphs, and every column represents one of the graphs in the set. Each cell is filled by 1, if the subgraph of that row exists in the graph of that column, or 0 otherwise. We proposed a probability under Blowfish Privacy to flip some of these cells from 0 to 1 or vice versa. The third step consists of applying in the graphs the changes that occurred in the second step. This step is done by adding the missed edges for the flips from 0 to 1. Then to remove some edges for the

subgraphs that have flipped from 1 to 0.

Dealing with active attacks becomes more complicated in online dynamic graphs than offline ones and much more challenging than in static graphs. This subpart proposes a technique to protect the individuals in online dynamic graphs exposed to active attacks. Our approach is based on the idea of Hypergraph, where a Hypernode called Detention Node will hide the nodes suspected of being Sybil by applying a catch and release mechanism based on Differential Privacy.

3.5 .2 Preliminaries

A dynamic or sequentially released graph G is composed of a set of static graphs $\{G_1, G_2, ..., G_i\}$ where each static graph $G_i = \{V_i, E_i\}$ is composed of a number of nodes V_i and edges E_i representing the connections between the nodes. G_i is dedicated to the connections that take place in the time interval $[t_{i-1}, t_i]$, for this reason, it doesn't show any node that didn't exist before the timestamp t_i. It's also important to mention that each node has a unique id that doesn't change between G_i and G_{i+1}.

3.5 .3 Active Attack on Sequentially Released Graphs

Passive attacks wait for the release of the anonymized graph and benefit from two types of knowledge to breach the privacy of individuals in the dataset. The first type is the auxiliary or the background knowledge. It's based on information known by the adversary about one or more individuals in the graph. For example, if we know that a traveler arrived yesterday to the town and on a dynamic published graph representing the town residents and their communications, one new node was added. We are sure that it represents this traveler. The second type is foreground knowledge, where the adversary performs some data science techniques to extract some information to breach the privacy of individuals by linking them to their representative nodes and find more about their connection with other individuals. For example, we notice that a high percentage of nodes that appear for a short period in the graph then disappear have created relationships with a specific node. We can analyze that these nodes are tourists and the particular node is Alex, the touristic guide in the town. Now that we have found him in the graph, we focus more and more on background and foreground knowledge about him to reveal more about his personal and professional life.

Meanwhile, the first step of the active attacks is performed even before the data owner generates the original graph representing the community. The adversary creates fake accounts or identities in the community to be presented in the graph. Then, they benefit from these nodes, called Sybil nodes, to connect to some victims and explore their nodes in the graph to find out more about their relations with other individuals.

Therefore, they create their own foreground knowledge in the original dataset, try to retrieve it from the published dataset, and then use it to reveal the real identity of the victims.

In this subpart, we focus on defending the individuals against the active attack on dynamic graphs formed of three steps which were highly inspired from [36] with some modifications. The first step is creating Sybil subgraphs where the attacker inserts a small number of nodes in several releases. By that process, a tempo-structural pattern is created relating each Sybil node to a specific release. $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ is the set of Sybil nodes forming one of the subgraphs injected by the adversary. Each S_i has two important characteristics: the first release $r(S_i) = G_j$ in which S_i was injected and a set of degrees of S_i representing its degree in each graph from its first appearance till the current graph: $Deg(S_i) = \{deg_j(S_i), deg_{j+1}(S_i), ..., deg_c(S_i)\}$ where j is the index of the graph where S_i was included for the first time and c is the index of the current graph. These two characteristics are crucial to start the second step of the attack.

To make the second step easier, it's also important to have a mechanism to set the edges between the nodes in the Sybil subgraph called inter-Sybil connection. In [17], they suggest to include an edge between S_i and S_{i+1} then to include with probability 0.5 between S_i and all other nodes in the Sybil subgraph. The standard results in random graph theory [25] imply that this form of subgraphs, with high probability, has no non-trivial automorphism. In other words, each Sybil node in this subgraph has a different characteristic from the other Sybil nodes, and it won't be confused with other nodes in the subgraph.

Sybil subgraph retrieval is the second step where the adversary tries to identify the Sybil subgraph in the released versions of the graphs. The adversary profits from their knowledge about the characteristics of the Sybil nodes and the structure of the subgraphs to retrieve them. First, a Sybil node S_i is chosen, then the adversary collects all the possible candidate nodes in the published graph that could be the actual S_i. We assume that a node x in G_c is a candidate to be the Sybil node S_i if:

- $r(x) \geq r(S_i)$. In [36], the two first appearances should be in the same graph. Still, we are considering the possibility that the data owner could apply a privacy-preserving mechanism that might hide the new node for some releases before allowing its appearance to make it harder for the adversary to assume the original graph where this node was included for the first time.

- the set of degrees $Deg(x)$ should be close to $Deg(S_i)$ where, for example, for the current graph, $deg_c(S_i) - \theta \leq deg_c(x) \leq deg_c(S_i) + \theta$, where θ is a tolerance threshold. And the same condition should be true for all other degrees existing in $Deg(x)$.

x and all other nodes that fit these two conditions form a set of candidates $\mathcal{X}_{S_i} = \{x^1_{S_i}, ..., x^m_{S_i}\}$.

The second part of this step extends each candidate from a single node to a subgraph until just one candidate subgraph resembles the Sybil one. Therefore, we can assume that this candidate is the real Sybil subgraph.

Using a Breadth-First-Search or Depth-First-Search techniques, we can list in Syb_{list} all the nodes in the Sybil graph starting from S_i (The Sybil node chosen in the first part of this step). Let S^2_i be the second Sybil node in Syb_{list}. Each candidate x in $\mathcal{X}_{S_i}$ should have at least one node

in its ego network that fulfills the two conditions listed in the first part when compared to S_i^2. In this case, x and its neighbor form a candidate subgraph in $\mathcal{X}_{S_i}$. If more than one neighbor fulfills the conditions, then x will form a candidate subgraph with each of these neighbors. If none of these neighbors fulfill the conditions, then x is removed from $\mathcal{X}_{S_i^2}$. For each other Sybil node in Syb_{list}, each subgraph candidate is extended or removed in the same way, but with one more condition. For S_i^4, for example, the extended subgraph should form an isomorphic subgraph to the one formed by S_i^1, S_i^2, S_i^3 and S_i^4. After the last node in Syb_{list}, the candidate graphs have the same number of nodes as the Sybil graph. If just one candidate graph remains in Syb_{list}, then this is considered as the Sybil graph. If more than one remains, more Sybil nodes are required to be injected in the coming graphs, and the same process is applied until one candidate remains. The third and last possibility is when all the subgraph candidates failed to resemble the Sybil subgraph. In this case, the adversary should increase the value of θ. In this way, more subgraphs will be considered candidates, increasing the possibility of finding the Sybil one.

When the Sybil subgraph is retrieved, the adversary could initiate the third step. This step aims to link individuals that the adversary knows they exist in the dataset to their actual nodes. It uses the Sybil subgraph to attack several individuals by contacting them using the fake accounts that form the Sybil subgraphs. One or more Sybil nodes reach each victim, and a set of candidate victim nodes is created. After each release, this set is updated until the adversary has complete confidence that a specific node represents the individual victim.

3.5 .4 Node-Detention Differential Privacy

This section represents our solution for protecting the graph from Sybil nodes using the same example of the traveler in the pandemic era. Thus, the first question is when the traveler can get into the plane to travel to our country? Approaches as SybilGuard, SybilLimit, and some similar ones, even if his PCR result is positive or he is from a high-risk country, he is allowed to get on the plane and enter the country. Still, they provide a platform for their citizens listing the new arrivals with positive results from high-risk areas and urge their citizens to stay far from them. Edge-DP also lets the traveler control his connections by deciding who can meet and who is forbidden to meet. Anyway, for the two previous approaches, the infected traveler has entered society and presents a health risk.

For this reason, node-DP has suggested a much more strict solution. Any person from a high-risk area or who has positive results will be forbidden to enter the country. This approach represents high protection for society but has many drawbacks. What if the traveler is a citizen and wants to return to his country from his country of residence? What if he's a doctor or a nurse that will help to fight Covid? What if he's a businessman that will help the economy of the country in this pandemic? This last solution provides a high level of protection but in many cases also presents a significant loss for the economic, health, or touristic sectors and a legal challenge for the authorities by banning a fellow citizen from entering his own country.

The solution for us is simple. We prepare quarantine facilities that accommodate any

traveler with a positive PCR or coming from a high-risk area. His exit from these facilities is decided based on his new PCR results and the number of days he has been quarantined. Getting out of these facilities doesn't mean he will never come back. Whenever the health authorities have some suspicions about him because someone from his surroundings was infected or was in a place at the same time with a covid patient, he will be required to do a new PCR test.

If it's positive, all his relatives, close friends, and co-workers must do the test. All the infected are quarantined in the facilities. The others must stay at home for a while till we are entirely sure they are Covid free.

When we are confident that this quarantined group gets rid of Covid, they are free to leave the facilities and their home.

That's very similar, with tiny differences, to our idea of dealing with new nodes in each graph that are suspicious of being Sybil nodes. We create a hyper node that we call the detention node v_{DET} where nodes could be hidden where some of their characteristics are suspicious. A probability P_{catch} computed under DP is used to decide if the new node v_i will be hidden or not in a published graph G_j^*. If P_{catch} is higher than a threshold θ_{catch} than v_i is hidden inside v_{DET}.

If v_i has been hidden in graph G_j^*, then for G_{j+1}^*, a probability $P_{release}$ is computed based on three parameters:

- the suspicions about the characteristics v_i in graph G_{j+1},
- the quality of the connections: having more connections to suspects give v_i increase the suspicions about v_i,
- the ¨criminal record¨ of v_i.

In the next section, while proposing a mechanism to apply our approach, we give a possible definition of suspicions and ¨criminal record¨ based on many features.

Let's say v_i was released, but new suspicions have been raised in a graph G_t. If $P_{catch-gang}$ is higher than $\theta_{catch-gang}$, v_i should be hidden in G_t^*. Let's say v_i is a Sybil node, and the adversary makes an extreme move via $v - i$ as connecting to a very high number of nodes to identify that v_i is their Sybil node. If v_i was the only node that appears in G_{t-1}^* and is hidden in G_t^*, therefore the adversary can be sure that v_i is a Sybil node.

For this reason, when $P_{catch-gang}(v_i) > \theta_{catch-gang}$, we check if v_i is related to a gang of k nodes than all the k nodes are hidden. If not, the mechanism should choose $k-1$ suspicious nodes unrelated to any other gang and group with v_i. The gang members should be hidden or on probation, hiding their ids but letting them be visible in the published graph G_t^*.

In future graphs, if $P_{release-gang}$ of each hidden member of the gang is higher than $\theta_{release-gang}$, then the whole gang is released.

The crucial difference between our technique and node-DP is that, in the latter, when a node is deleted, all its edges are also deleted. Thus, the ego network and the degree of all its neighbor nodes are affected. In our approach, the only edges that are hidden are those linking the nodes inside v_{DET}. All the edges connecting the hidden node v_i with a free node v_f appear

as linking v_f with the detention node v_{DET} without the possibility for the public of specifying the corresponding node inside v_{DET}.

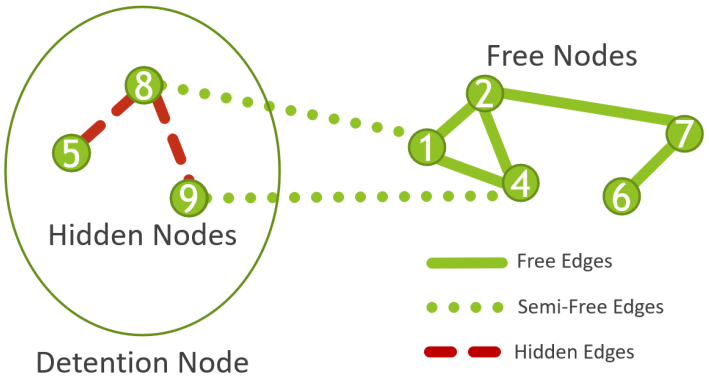


Figure 39: The types of nodes and edges

From here, we can list four types of nodes:

- **Detention node** v_{DET} is a special type of hyper node that could hold many nodes inside of it. The id, the number, and the edges connecting these nodes are hidden. We have just one detention node in every released graph.
- **Hidden node** is a node hidden inside the v_{DET} so the public cannot see it.
- **Free node** is the normal node that the public can see.
- **On-probation node** is the same as a free node, but its id is hidden from the public.

The types of nodes implies that we can also list three types of edges:

- **Free edge** is an edge between two free nodes: $e(v_i, v_j) \in E_f$ if $v_i, v_j \notin v_{DET}$ where E_f is the set of free edges.
- **Semi-free edge** is an edge between a free and a hidden node. The public can see that the edge is linking a free node to the detention node but cannot know the Id of the hidden node: $e(v_i, v_j) \in E_sf$ if just one of v_i and v_j is in v_{DET} where E_sf is the set of semi-free edges.
- **Hidden edge** is an edge hidden from the public linking two hidden nodes: $e(v_i, v_j) \in E_h$ if $v_i, v_j \in v_{DET}$ where E_h is the set of hidden edges.

Our approach aims to ensure a high level of uncertainty for the adversary when trying to re-identify the Sybil nodes in the published graphs. When adding a new node to a graph G_j, it's uncertain in which published graph it will appear for the first time. Even after releasing the new nodes, they could be hidden again whenever they are suspicious so that the adversary couldn't have more than $1/k$ confidence about which node is the Sybil one.

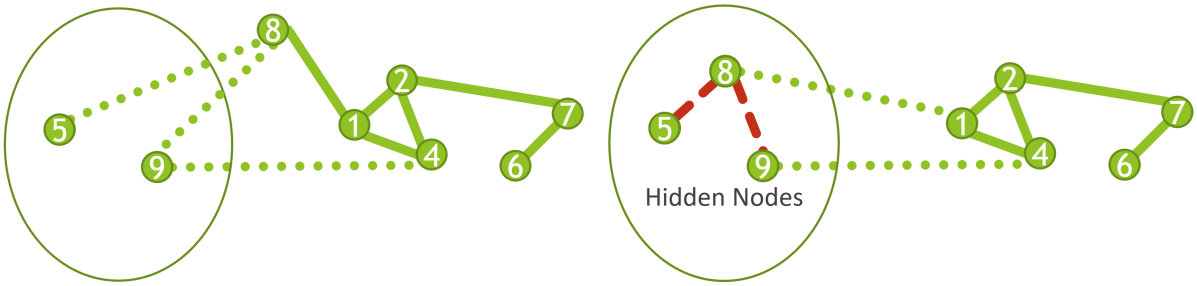


Figure 40: Two neighbor graphs G_1 and G_2 differing by just Node 8.

3.5 .4.1 Neighbor Graphs in Node-Detention Differential Privacy

As we can see in Figure 40, we consider that two graphs are neighbors if one node differs by being free in one of the graphs and hidden in the other one. Let G_1 be the graph where Node 8 is free, and G_2 be the graph where Node 8 is hidden.

Definition 4. *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs where $V_1 = V_2$ and $E_1 = E_2$. Let $V_1 = V_{h1} \cup V_{f1}$ and $V_2 = V_{h2} \cup V_{f2}$ where V_{h1} and V_{f1} (V_{h2} and V_{f2}) are the set of the hidden nodes and free nodes respectively in G_1 (in G_2). And $V_{h1} \cap V_{f1} = V_{h2} \cap V_{f2} = \emptyset$. The two graphs are neighbors if and only if*

$$V_{f1} \Delta V_{f2} = V_{h1} \Delta V_{h2} = \{v_d\}$$

In other words, in the two neighboring graphs, one and only one node differs between the sets of hidden nodes between the first and the second graph, and it's the same node that differs between the set of free nodes also.

3.5 .4.2 Global Sensitivity of Node-Detention Differential Privacy

The most critical advantage NDDP has over node-DP is the relatively low global sensitivity.

The global sensitivity computes the maximum number of edges that could be affected by adding or removing one element from the graph. In this way, we have an indicator of the noise needed to cover the affected edges.

In edge-DP, the concerned element is an edge. Then it's evident that the number of affected edges is just one. While for the node-DP, the concerned element is a node. By adding or removing a node, the number of affected edges is related to the degree of this node. For example, if we add a node with a degree of 15, then 15 edges were added to the graph. In the worst-case scenario, we add a node with the highest possible degree, which is the number of nodes in the graph beside the new node. The global sensitivity for Node-DP is $\Delta f_{node-DP} = \mid V \mid -1$.

In NDDP, the concerned element, the same as node-DP, but instead of adding or removing the node and affecting all its edges, we just hide it, and the only edges that are completely hidden with the node are thus relating it to other hidden nodes. In this way, the maximum number of edges affected is related to the number of hidden nodes in the graph instead of the total number of nodes.

In Figure 40, the only edges that were hidden from the public by hiding Node 8 in G_2 are those linking Node 8 to hidden nodes. Then the maximal difference in the number of hidden

edges between the two neighbors is the number of hidden nodes in the graph where Node 8 is free. In our example, the global sensitivity of NDDP is 2, while global sensitivity of node-DP would be the highest possible degree of Node 8, which is 7.

In general, global sensitivity of NDDP could be computed by considering two neighbor graphs G_1 and G_2. The node that differ between these two graph is v_{dif}, which is free in G_1 and hidden in G_2. In a worst-case scenario, v_{dif} is connected to all the hidden nodes in G_1 and their number is $\mid V_{h1} \mid$.

In G_2, v_{dif} is hidden, thus, $\mid V_{h2} \mid = \mid V_{h1} \mid +1$. And all its edges connected to other hidden nodes are also hiddens. In the worst-case scenario, the number of these edges is $\mid V_{h1} \mid$ and that's the global sensitivity.

Therefore, to compute the global sensitivity, having two neighbor graphs G_1 and G_2, we focus on the graph where v_{dif} is free; it's the graph with the lowest number of hidden nodes. The global sensitivity is the maximum possible number of edges between v_{dif} and the hidden nodes; in other words, it's the number of hidden nodes in that graph.

For this reason, the global sensitivity of NDDP is

$$\Delta f_{NDDP} = min(\mid V_{h1} \mid, \mid V_{h2} \mid)$$

while that of node-DP is $\mid V \mid -1$ which is, usually, much higher than $min(\mid V_{h1} \mid, \mid V_{h2} \mid)$. On the other side, the global sensitivity of edge-DP is indeed very low. Still, we have explained in the Related Work section how edge-DP could not prevent Sybil nodes from being injected into the graph and forming a threat to the privacy of the individuals.

3.5 .5 Discussion

3.5 .5.1 Adding Edge-Differential Privacy

As we have explained in the Subsection 3.2.5.3 and Table 2, edge-DP cannot by itself protect the individuals in an online dynamic graph. It's because a Sybil node will be allowed to appear in the published graph whenever the adversary injected it, and then adding some noise to its original degree might not be enough to prevent the adversary from retrieving it with high confidence.

While our approach is detected to this type of protection, we notice that the only edges affected by our algorithm are those connecting two hidden nodes, and the edges connecting a free node to a hidden node are partially affected. Therefore, the adversary is entirely sure that an edge relating to two free nodes in the published version is a real edge. In addition, the degree of any free node is also the real degree of that node. When a node has all its neighbors free, the adversary fully exposes this neighbor list.

For this reason, we recommend using our approach with any edge-DP mechanism that adopts a non-interactive approach (a mechanism that returns a noisy graph instead of a noisy query result).

The composite theorem [55] gives us the possibility of dividing the privacy parameter ϵ into ϵ_1 and ϵ_2. ϵ_1 for the mechanism intending in manipulating the edges and ϵ_2 for our mechanism. In that way, the anonymization procedure is considered (ϵ_1, ϵ_2)-NNDP.

3.5 .6 A Mechanism of Node-Detention Differential Privacy

This section proposes a mechanism to apply the NDDP, relying mainly on the exponential mechanism. When a graph has one or more new nodes, the mechanism should decide which nodes will be published and which will be hidden in the detention node.

The exponential mechanism, as we have explained in the Background section, relies on three parameters: the privacy parameter ϵ provided by the data owner, the quality score q that we will compute for each step of our mechanism in this section, and finally the global sensitivity GS based on the quality scores.

Before computing these two parameters, we should notice that the adversary doesn't know the characteristics of the graph release that they are trying to insert the Sybil nodes in. But they see the structure of the previously published graphs. The adversary always aims to insert the Sybil nodes with unique characteristics to retrieve them in the second stage of the attack easily.

In our mechanism, we will focus on the degree of the nodes, but any other feature could be considered instead of the degree in other mechanisms.

The adversary has two ways to choose the degree of the nodes. The first one is arbitrary, and hoping that the degree is not very common in the graph. The second one is based on the previously published graphs. These graphs are indeed noisy, but the noise should not affect the degrees severely because the utility of the graphs would be highly affected in this case. Therefore, computing the average degree of the previous releases and the number of times each degree occurs in the earlier graphs might give the adversary a good idea of what degrees to choose and what to eliminate. For example, is it common to have isolated nodes or to have nodes with a maximum degree? Is it rare to have nodes with more than 1000 degrees or less than 5 degrees? Answering these questions would help the adversary in choosing the degrees of their Sybil nodes.

This mechanism is based on the idea of anomaly detection. We put ourselves in the adversary's shoes, and for each node, we ask: is it a degree that, as adversaries, would we use it for our Sybil node or not? The more likely the answer would be yes, the more the quality we give to the node is higher, leading to a higher chance of being hidden in the published version.

The answer to the previous question is based on an analysis that we think the adversary has done on the previously published graphs. But the other way is that they have chosen the degrees arbitrarily without any study on the published data. Our advantage here is that we have the original graph before starting our privacy mechanism. When the adversary makes decisions about the node degrees, they know nothing about the original graph. For this reason, when computing the quality of the nodes, we consider the two possibilities. A part of this score will be linked to our analysis of previous graphs, and the other part is related to analyzing the current graph. In this way, even if the Sybil degrees were arbitrary, we forbade the adversary

from retrieving them in the published version of the current graph.

3.5 .6.1 Quality Components

In the coming subsections, we explain how we propose to apply the four stages of our mechanism: the First Capture, the First Release, the Gang Capture, and the Gang Release. In each of these stages, we use the exponential mechanism to decide the status of each node in the published versions. Each exponential mechanism needs a quality score as an input. The quality scores have four components that recur in each score.

We will explain these four components based on each node's characteristics and its relations with other nodes.

3.5 .6.1.1 Clustering based on degrees

When a Sybil node has a unique characteristic, it will help the adversary retrieve it from the published graphs. For this reason, we apply the Kernel Density Estimation (KDE) to cluster the nodes based on their degrees. The clustering is based on 1-Dimensional data; thus, we can consider the clusters as integrals.

KDE creates several minima and maxima. Each minima is considered as a border between two integrals, and each integral has a maxima. Each node is inside an integral based on its degree. The absolute difference between the degree of a node $deg(v_i)$ and maxima of its integral $maxima(v_i)$ is a component in the quality score. The higher difference gives an indicator that the number of nodes having the same degree is low. Then, if it's a Sybil node, it's easier for the adversary to retrieve it.

Then, the first component is:

$$q_c(v)_{G_j} =\mid deg(v) - maxima(v) \mid$$

where $maxima(v)$ is the maxima of the integral where v exists based on its degree in graph G_j.

It's important here to clarify for the reader that this quality is not about uniqueness, where if a node has a unique degree, then it's suspicious. But it's closeness; more a node has a degree (even if unique) close to the degrees of numerous other nodes, that makes this node less suspicious.

For example, in an interval between 20 and 30, and the maxima is 23. Having a unique node of degree 24 doesn't make it suspicious more than two nodes of degree 28. The unique node is close to the maxima; then many nodes have the degrees 22, 23, and 25. Thus the degree 24 is not an anomaly. While two nodes of degree 28 might be considered suspicious because they are far away from the main gathering in this interval, thus it's easier for the adversary to find them if they are Sybil nodes. Even if their degrees are anonymized, the adversary could discover them with higher confidence than the unique node of degree 24.

It's always important to remember that the goal of privacy-preserving mechanisms is not exclusively privacy, but it's about a balanced trade-off between privacy and utility. Therefore, the adversary would expect that by applying edge-DP, the original degree of its Sybil node

has changed by, for example, ± 2 most probably. An actual value of an anonymized published degree of 28 is most likely between 26 and 30. As those degrees are far from the maxima, then the number of nodes in this area is low, and it's easier for the adversary to find if these nodes are their malicious nodes or not. While the unique node of anonymized degree 24 most probably, has an original degree between 22 and 26. It's a crowded area where it's not easy to find if this is the Sybil node or not. For this reason, a unique node could be more trusted than multiple nodes far from the maxima.

3.5 .6.1.2 Profile Quality

Each node has a profile helping us to know its history. Precisely as a person's medical history is vital to deal with their possible Covid infections, their history of vaccination or diseases determines if quarantine is required.

We apply the same concept to the nodes. We create a vector $Q(v)$ containing the qualities of the node v in the previous graphs. Thus, after the release of the published version G^*_{j-1}, we could update the vector by adding the quality of v in that graph.

$$Q(v) =< q(v)_{G_f}, ..., q(v)_{G_{j-1}} >$$

where G_f is the first graph where v appears.

The quality profile $q_p(v)_{G_j}$ is the second component of v for the graph G_j:

$$q_p(v)_{G_j} = \frac{1}{\mid Q(v) \mid} \sum_{i=1}^{|Q(v)|} \frac{q(v)_{G_i}}{\mid Q(v) \mid -(i-1)}$$

This component has two functions: the first is computing the quality of v in G_j, and the second is computing the quality of any neighbor node of v in G_j (the fourth component).

3.5 .6.1.3 Connection Quality

The connection score is the third component of the quality score. This component examines the relations of the node. Going back into our example about the Covid pandemic, let's say that the records show that most of the traveler's family members that come back from the same country some days ago were infected. In that case, there should be a higher possibility of imposing a quarantine on the traveler than if his family members were not infected.

The connection score component has the same goal of investigating the relations of new and old members of the graph. In our mechanism, we use three statuses and a score for each of them, as we can see in Table 8. The data owner might have statuses or other scores to use, but the scores should always be positive.

Neighbour Status	Score
Hidden	2
New	1
Free	0.5

Table 8: Sensitive data associated to the nodes.

Let's say that v has ten neighbors in G_i, where 5 of them were free in G^*_{i-1}, three were hidden, and two new nodes. Then,

$$Conn_{score}(v) = \{score : N_{neighbours}\} = \{2:3, 1:2, 0.5:5\}$$

Therefore, the connection quality is

$$q_{conn}(v)_{G_j} = \prod_{l=0}^{|Conn_{score}(v)|-1} (score^{N_{neighbours}})[l] = 2^3 \times 1^2 \times 0.5^5 = 0.25$$

3.5 .6.1.4 Neighboring Quality

The fourth component has the same concept as the connection quality in investigating the connections of v_i. But in this component, we focus on the qualities of the neighbors, not their status in the last published graph. It's essential to check the statuses of the neighbors of v_i, but it's also important to compute their qualities.

Thus, we compute the mean of the qualities of v's neighbor:

$$q_n(v)_{G_j} = \frac{\sum_{i=1}^{|N_o(v)|} q_p(N_o(v)[i])_{g_{j-1}}}{\mid N_o(v) \mid}$$

where $N_o(v)$ is the set of neighbors that are old nodes.

These four qualities listed and explained are engineered features to compute an anomaly score for a given node. Now, we can start by proposing the four stages of our mechanism.

3.5 .6.2 Anonymization of First Graph G_1

In the first graph, we cannot categorize the nodes as new and old. All of the nodes are new, and their quality vector Q is still empty.

Therefore, the only component available is q_c based on the difference between the degree of the node and its integral maxima.

$$q(v)_{G_1} = q_c(v)_{G_1}$$

The set of neighboring graph $N(G_1)$ of G_1 is the set of graphs that differ by just one node of G_1. Then, it's similar to the node-DP in the first graph as we don't yet have detention node and hidden nodes.

The global sensitivity of $q(v)_{G_1}$ is the maximum difference in the distance between the degree of v and its interval's maxima in G_1 and any graph of $N(G_1)$. This difference is based on two possibilities. When a node is added or deleted in a neighbor graph G'_1, the value of $maxima(v)$ will change, or even v will change its interval. Then, the global sensitivity is:

$$GS(q(v)_{G_1}) = \max_{G'_1 \in N(G_1)} \mid q(v)_{G_1} - q(v)_{G'_1} \mid$$

The rest of the anonymization mechanism for G_1 is similar to what we will explain in Subsection 3.5 .6.5.

3.5 .6.3 First Capture

The First Capture is a step in the mechanism applied on the new nodes in the current graph G_j where $j > 1$. It will provide a list of new nodes that should be hidden. In this subsection, we compute our suggested quality score and global sensitivity for this mechanism. The quality of each new node v_{new} is composed of three components already explained. The component missed in this quality is the profile quality q_p as the quality vector of this node is still empty.

$$q^1(v_{new})_{G_j} = q_c(v_{new})_{G_j} \times q_{conn}(v_{new})_{G_j} \times q_n(v_{new})_{G_j}$$

To compute the global sensitivity $GS(q^1(v_{new})_{G_j})$, we create a set of published graphs $\mathcal{G}'^* = \{G_1'^*, ..., G_{j-1}'^*\}$ that differ from the real set of published graphs $\mathcal{G}^* = \{G_1^*, ..., G_{j-1}^*\}$ by just one node v_{diff} that is hidden in one set and free in the other set. v_{diff} differs in the last graphs of the set G_{j-1}^* and $G_{j-1}'^* \in N(G_{j-1}^*)$ where $N(G_{j-1}^*)$ is the set of graphs that differ by just v_{diff} from G_{j-1}^*, where v_{diff} is free in one graph and hidden in the other.

By this difference, two qualities are affected. First, q_{conn} could change by $\max(\frac{score_{hidden}}{score_{free}}, \frac{score_{free}}{score_{hidden}})$. The second component is q_n that computed the mean of the qualities of the neighbours of v_{new}. q_n changes based on the modifications that happens on the quality of the node v_{diff} that differs between G_{j-1} and its neighbour graph, and also the qualities of the nodes that are in the same time neighbours of v_{diff} in G_{j-1} and neighbours of v_{new} in G_j. Therefore,

$$GS(q^1(v_{new})_{G_j}) = q_c(v_{new})_{G_j} \times \max(\frac{score_{hidden}}{score_{free}}, \frac{score_{free}}{score_{hidden}}) \times \max_{G_{j-1}'^* \in N(G_{j-1}^*)} \mid q_n(v_{new})_{G_j} - q_n(v_{new})_{G_j'} \mid$$

where G_j' and G_j are the same but they differ by the last published graph $G_{j-1}^{*'}$ and G_{j-1}^*.

Finally, the probability that v_{new} is hidden in the published version of G_j is

$$P(exp(G_j, V_{new}^{G_j}, q^1, \epsilon) = v_{new}) = \frac{exp(\frac{\epsilon q(G_j, v_{new})^1}{2GS(q^1)})}{\sum_{v'_{new} \in V_{new}^{G_j}} exp(\frac{\epsilon q(G_j, v'_{new})^1}{2GS(q^1)})}$$

where $V_{new}^{G_j}$ is the set of new nodes in graph G_j. v_{new} is hidden if $P(exp(G_j, V_{new}^{G_j}, q_1, \epsilon) = v_{new})_1 > \theta_{first-capture}$ where $\theta_{first-capture}$ is a threshold set by the data owner. Higher threshold gives more protection to the graph from sybil nodes but less utility by hidden a greater number of new nodes.

3.5 .6.4 First Release

In this subsection, we are focusing on hidden nodes in G_{j-1} that have never been free in any prior releases. After introducing the fake account in a previous release, the adversary is unsure if the node representing this account is still hidden or a free node. For this reason, they still try to give a remarkable degree to the node in the same way as the first creation to identify the Sybil node in case it is already free. Here, the quality contains the four components explained previously with an additional element related to the number of releases $hr(v_h)$ where v_h was hidden.

$$q(v_h)^2_{G_j} = \frac{hr(v_h)}{q_c(v_h)_{G_j} \times q_p(v_h)_{G_j} \times q_{conn}(v_h)_{G_j} \times q_n(v_h)_{G_j}}$$

The global sensitivity of q^2 for two neighboring sets of published graphs similar to the previous subsection:

$$GS(q^2, v_h) = \max_{G'^*_{j-1} \in N(G^*_{j-1})} \mid q(v_h)^2_{G_j} - q(v_h)^2_{G'_j} \mid$$

The exponential probability is:

$$P(exp(G_j, V_h^{G_j}, q^2, \epsilon) = v_h)_2 = \frac{exp(\frac{\epsilon q(G_j, v_h)^2}{2GS(q^2)})}{\sum_{v'_h \in V_h^{G_j}} exp(\frac{\epsilon q(G_j, v'_h)^2}{2GS(q^2)})}$$

where $V_h^{G_j}$ is the set of nodes hidden in the detention node in G^*_{j-1} and waiting their First Release.

v_h is released from the detention node and will be a free node in G^*_j if $P(exp(G_j, V_{new}^{G_j}, q^2, \epsilon) = v_{new})_2 > \theta_{first-release}$ where $\theta_{first-release}$ is a threshold set by the data owner.

3.5 .6.5 Gang Capture

In every release, some nodes might be suspicious without being new nodes. Then, for each release, a privacy mechanism for old nodes should be applied.

Identical quality scores and global sensitivities as those in the First Capture are also used in the Gang Capture. If the probability is higher than $\theta_{Gang-Capture}$, then the node should be hidden. But we assume this is not enough to prevent the adversary from identifying their Sybil nodes.

While the adversary is still unsure about the Sybil node they have implemented in a previous graph, they might try to perform an extreme move and then benefit from the list of nodes that were free in the last graph and hidden in the currently published one. Suppose the list contains a small number of nodes. In that case, the adversary might recognize their Sybil nodes, especially if the other newly hidden ones don't have close characteristics to the Sybil nodes. In this way, the adversary can retrieve the Sybil nodes and being one step closer to find their Sybil subgraphs.

For this reason, we add another layer of protection based on the idea of node gangs. When the mechanism returns that an old node v_s is a suspect and should be hidden in this graph, we check if it is already linked to a gang or not. If the answer is yes, all the members that are also suspects are hidden while the remaining members are put on probation. A node is on probation when it's not in the detention node, but its id is hidden.

If v_s isn't associated with any gang, we choose $k-1$ nodes unassociated to a gang and having the closest quality scores to v_s. In this way, we hide the suspected nodes of the gangs and keep the highest possible utility by leaving the other members outside the Detention. At the same time, the protection of the hidden nodes' ids of this gang was not affected because the

ids of the free members are also hidden. Thus, the adversary cannot differentiate between the gang members to find their Sybil node.

3.5 .6.6 Gang Release

The exponential probability for the Gang Release is similar to that of the First Release. But to release a gang, all its hidden members should have a probability greater than $\theta_{gang-release}$. Sometimes, it's difficult for all the hidden members to get an enough high probability to be freed if the threshold has a high value. Then, it's recommended to have a $\theta_{gang-release}$ relatively low.

Another issue that we can face in this stage is the scenario where the user (adversary or honest) of an account has stopped using it for good then the degree of the node v_{d0} is 0. It might be the reason to keep the other gang members on probation for a long time.

Our solution is based on the idea of identity swap. Let's take the scenario where a gang is hidden in a previously published graph mainly because of a set of nodes V_{d0} having 0 degrees and the gang still in the detention node. First, we perform the exponential probabilities in the current graph to check if the gang is eligible to be freed. If not, we review the nodes responsible for that result. If all of them have a 0-degree in this graph and the last z original graphs, the identity swap occurs.

We pick the node v_{is} with the highest exponential probability P_1 when it was hidden and perform an identity swap with $v_{d0} \in V_{d0}$. Thus, now the node having 0 degrees has the id v_{is}, and the node eligible to be freed has the id v_{d0}. The gang is freed in the currently published graph with its secretly new members, while the public doesn't know that some members are new with swapped identities.

This scenario is not always necessary. In any exponential mechanism, we have some randomness conditioned on the quality scores. Then, there is some possibility that the probability for the 0-degree node is high enough not to experience the Gang Capture or to perform a Gang Release, especially that we have recommended that $\theta_{gang-release}$ should not be high.

3.5 .6.7 State machine diagram

After listing and explaining all the parts of the mechanism, we will recap them by drawing a state machine diagram showing all the possible statuses of a node and how they could shift from one status to another:

- We start with a new node. In this status, the node has two options; it will be hidden in the published version of the graph if the First Capture step outputs this decision; otherwise, it will be a free node.

- A hidden node in the last published graph has three options in the current graph:

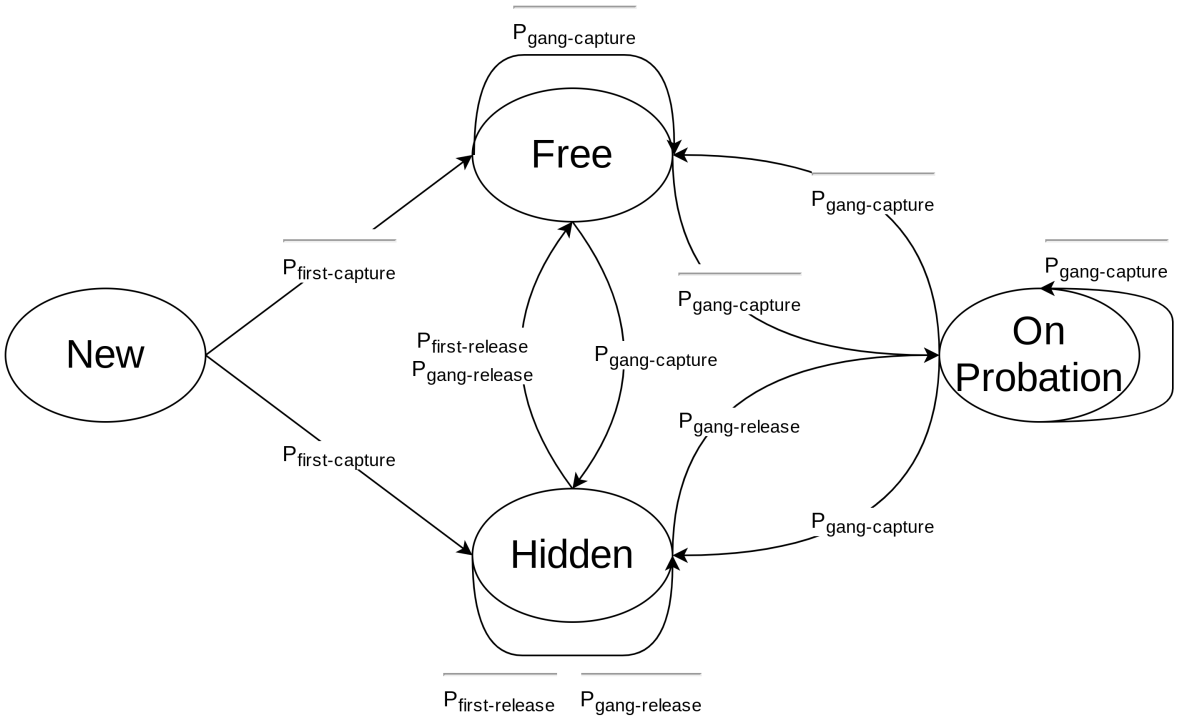


Figure 41: A state machine diagram for all the possible statuses of the node.

  - It could become a free node if the First Release algorithm (applied if the node was not free in any previously published graph) returns this decision, or the Gang Release algorithm (applied if the node was free previously in at least one published graph and now, it's a member of a gang) returns this decision, and none of the other gang members is suspicious.
  - In case one member at least is suspicious, the freed nodes of this gang are put on probation.
  - If the First Release or the Gang Release algorithm decides that the nodes are still suspicious, the user will stay hidden in the detention node.

- A free node in the last published graph also has three options in the current one:
  - If the Gang Capture algorithm detects an anomaly in the current characteristics of the node, then the node is hidden in the detention.
  - Otherwise, the node will remain free if it's not related to any gang or all the fellow members in the gang are not suspicious.
  - If at least one gang member is suspicious; then our innocent node is put on probation.

- The fourth status is on probation. This status means that, in the last published graph, the node is not suspicious, but it's a member in a gang where at least one member is suspicious. As the node was not suspicious, then, as any free node, the Gang Capture algorithm will be applied to it. The options for the current graph are as follow:
  - The algorithm returns that the node has become suspicious and should be hidden.

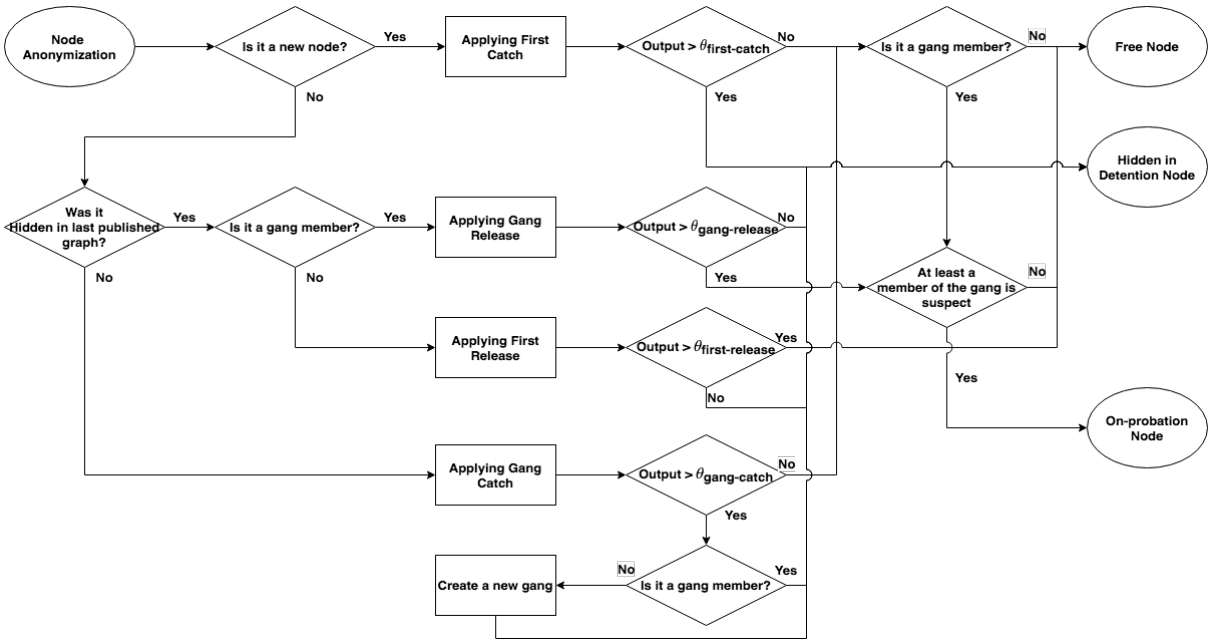


Figure 42: Process Diagram of the Detention Differential Privacy.

– The algorithm returns that the node is still not suspicious, but other gang members will remain hidden. Thus, the node will still be on probation.

– The algorithm returns that all the gang members are not suspicious anymore. The node and all his fellow gang members are free.

3.5 .7 Process Diagram

The diagram in Figure 42 shows the decision-making that the mechanism passes through for a node v_i to decide its status in the published version. The first question is if it's a new node or not. If the answer is yes, then the First Capture mechanism is applied. If the output is greater than the threshold $\theta_{first-catch}$, thus, the mechanism should hide the node. If not, the mechanism waits until all the new gangs are formed for this graph and checks if v_i is chosen as a gang member. Its status is on probation if it's a member; otherwise, it's a free node. If it's an old node, the mechanism checks its status in the last published graph. The Gang Capture would be applied regardless of v_i being free or on probation if it weren't hidden. If the output is higher than $\theta_{gang-catch}$, we check if v_i is a gang member. If it's not, we create a new gang, but v_i will be hidden in both cases. If the output is less than $\theta_{gang-catch}$, the mechanism checks if it's a gang member to decide if it should be free or on probation.

In that way, we have covered the possibilities of v_i being new or old and free or on probation in the last published graph. We still have the case of being hidden. In this scenario, we check if it's a gang member. Then we decide which part of the mechanism should be applied between the First Release and the Gang Release. If the output is less than the required threshold, the node will stay in Detention in both cases. The difference is when the output is higher than the threshold. In the First Release case, the node is directly freed, while in the Gang Release case, we check if all the members are free, then v_i is free also; otherwise, v_i is on probation.

3.5 .8 Experiments

In this section, we present the results of our experiments to evaluate the NNDP on dynamic graphs both in terms of utility and privacy. We implement our algorithm in Python on the "Reddit Hyperlink Network" dataset [83] containing daily releases between January 2014 and April 2017. The dataset contains 55863 subreddits (communities on Reddit.com) as nodes and 858490 edges representing hyperlinks connecting the subreddits. We conducted experiments on an Intel Core i7 2.4 GHz PC with 8GB RAM.

Besides our algorithm, we will also apply an edge-DP algorithm called TmF [125] which adds noise to each graph independently from other graphs by adding and removing edges before releasing an anonymized graph (non-interactive approach). For the node-DP, as we have mentioned in the Related Work section, none of the node-DP mechanisms adapts to the non-interactive mechanism. For this reason, we created two baseline node-DP algorithms to compare their results to the NNDP results.

3.5 .8.1 The baseline node-DP algorithms

The naive algorithm is based on randomly deleting nodes from the original graph. The second algorithm is based on an exponential mechanism where the quality is based on the uniqueness of the degree. A node with a unique degree in the graph has a higher probability of being removed than a node that shares the same degree with other nodes. A higher number of nodes sharing the same degree leads to a lower probability of one of them being removed.

The second algorithm is based on the exponential mechanism, which is a well-known DP mechanism. But the first one is based on a random mechanism; thus, we have to prove it respects the node-DP requirements. For each node in the original graph, we randomly generate a number between 0 and 1. If this number is smaller than a threshold θ, the node is removed from the anonymized version.

Let's consider two neighbor original graphs G_1 and $G_2 = G_1 \cup \{v_{diff}\}$ that differ by just one node. Let S be a subgraph of G_2. First, we apply the random mechanism $\mathcal{R}$ on the two original graphs. Then, we compute the probability that the output graph has the same nodes as S. As the nodes are similar in G_1 and G_2 except for v_{diff}, we can conclude that by dividing the two probabilities, we are dividing the probabilities that the status of v_{diff} in the anonymized version of G_1 (or G_2) is still the same in S:

$$\frac{P[\mathcal{R}(G_1) \longrightarrow S]}{P[\mathcal{R}(G_2) \longrightarrow S]} = \frac{P[\mathcal{R}(G_1)[v_{diff}] \longrightarrow S[v_{diff}]]}{P[\mathcal{R}(G_2)[v_{diff}] \longrightarrow S[v_{diff}]]}$$

where $\mathcal{R}(G_1)[v_{diff}$ ($\mathcal{R}(G_2)[v_{diff}$) represents the status of the node v_{diff} as exists or doesn't exist in the anonymized graph $\mathcal{R}(G_1)$ ($\mathcal{R}(G_2)$ respectively).

That's because any other node v in G_1 and G_2 has the same status, thus, has the same probability:

$$P[\mathcal{R}(G_1)[v] \longrightarrow S[v]] = P[\mathcal{R}(G_2)[v] \longrightarrow S[v]] = 1$$

.

To prove that a mechanism is node-DP, we should demonstrate that the division of those two probabilities has an upper bound that we can compute and that we call it e^{ϵ}.

In the case of this random algorithm, we can list two scenarios. The first is that v_{diff} exists in S, and the second is that the node doesn't exist. In both scenarios, v_{diff} exists in G_2 and doesn't exist in G_1.

In the first scenario:

$$\frac{P[\mathcal{R}(G_1)[v_{diff}] \longrightarrow S[v_{diff}]]}{P[\mathcal{R}(G_2)[v_{diff}] \longrightarrow S[v_{diff}]]} = \frac{0}{\theta} = 0$$

As we are just removing nodes, a node that doesn't appear in the original graph has zero chance of existence in the anonymized version. Then in this first scenario, v_{diff} exists in S, there is no possibility that $\mathcal{R}(G_1)$ has the same nodes as S.

In the second scenario,

$$\frac{P[\mathcal{R}(G_2)[v_{diff}] \longrightarrow S[v_{diff}]]}{P[\mathcal{R}(G_1)[v_{diff}] \longrightarrow S[v_{diff}]]} = \frac{\theta}{1} = \theta$$

Then the highest upper bound in the two scenarios is $e^{\epsilon} = \theta \leq 1$. Therefore, ϵ should be positive (which is always the case) to consider that the algorithm satisfies the node-DP requirements.

3.5 .8.2 Privacy parameters

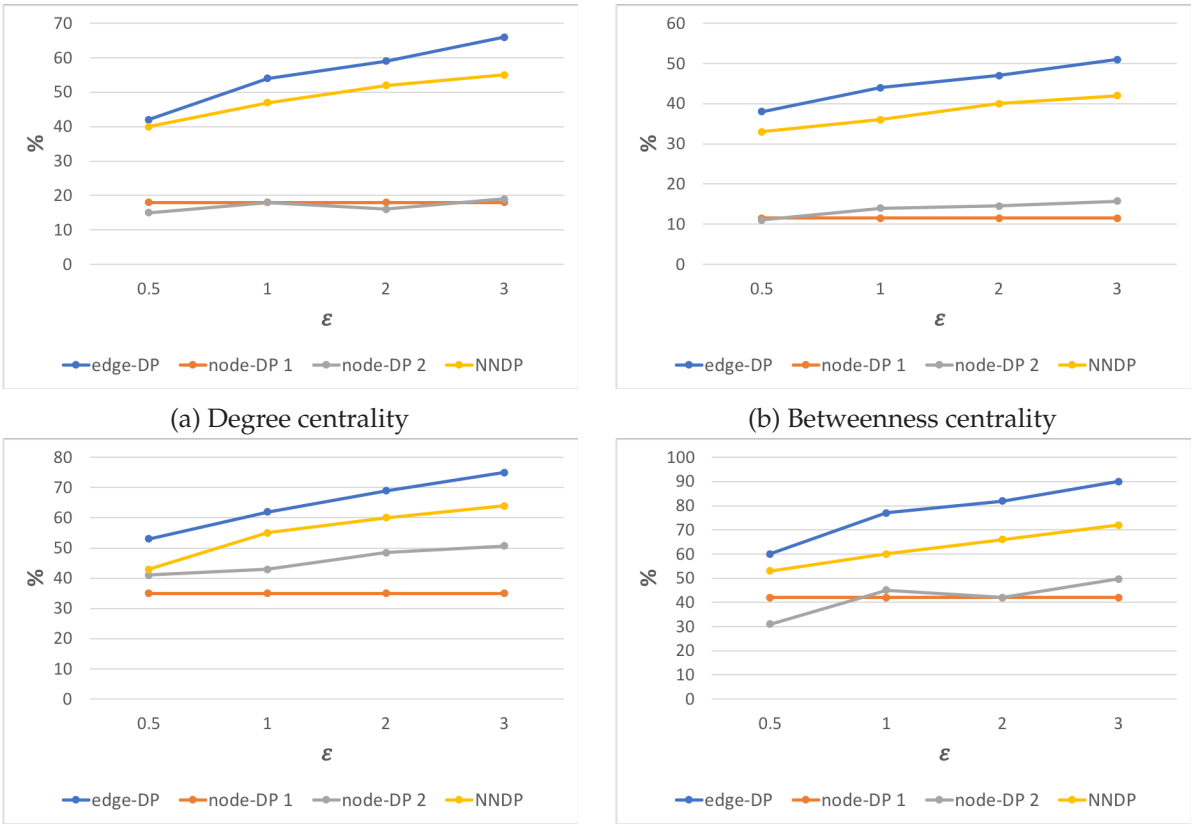
The TmF mechanism has two privacy parameters ϵ_1 and ϵ_2. In [125], they fix the value of $\epsilon_2 = 0.1$, thus we did the same in our experiments. ϵ_1 is related to the number of nodes $\mid V \mid$: $\epsilon_1 = coef \times ln(\mid V \mid)$. The values under the horizontal (x) axis represent the values of $coef$, and at the same time, they represent the second privacy parameter in NNDP as explained in Subsection 3.5 .5.1. The first baseline node-DP is random and doesn't rely on a privacy parameter; thus, its results will be presented as a horizontal line in the charts.

3.5 .8.3 Utility metrics

One of the most known methods to measure the utility of an anonymized graph is to compare the original graph with the anonymized one on the base of the most important nodes in the graph. The more these nodes are still the same in the two graph versions, the more we can assume that the privacy-preserving algorithm provides high utility to the published data.

In our previous work [3], we have explained and used four centralities to choose the most important nodes: degree, farness, betweenness, and eigenvector. In these experiments, we use the same centralities.

It is not possible to compute and compare in a precise way the centralities between two graphs that differ in the number of nodes, especially for betweenness and closeness centralities. Therefore, comparing the original graphs and their published version isn't practical for the node-DP and NNDP mechanisms.



(a) Degree centrality

(b) Betweenness centrality

(c) Closeness centrality

(d) Eignevector centrality

Figure 43: Number of common nodes for the 100 most important nodes based on four centralities.

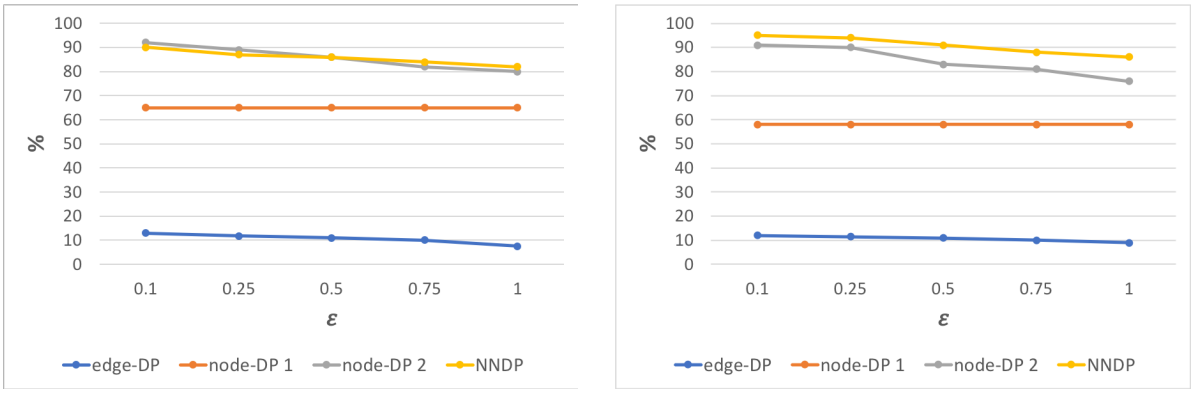
For this reason, we have compared the published graphs (the free nodes and their edges in the case of NNDP) with their corespondent subgraph in the original graph. Then, the goal of experiments in this subsection is not to find out the number or the rate of important nodes hidden or removed by the anonymized methods. But the objective is to study the impact of these privacy-preserving techniques on the nodes that appear in the published graph (nodes that are not hidden or removed). We apply this study by comparing the 100 most important nodes in both the published graphs and their counterparts subgraphs in the original graphs.

The charts in Figure 43 show that the two baseline node-DP mechanisms have obvious less utility in the scope of most important nodes in the original graph and published graph. As expected, the edge-DP surpasses the NNDP mechanism, but this out-performance is small compared to the difference between the node-DP and the edge-DP.

3.5 .8.4 Encountering the active attack

In this subsection, we simulate an active attack on the four anonymized versions of the published graph (anonymized by TmF, the two node-DP algorithms, and the NNDP). We will choose some new nodes as Sybil nodes; we will try to retrieve them from the anonymized graphs. The rate of successful retrieve indicates how much vulnerable the privacy-preserving algorithm is against an active attack.

We perform two types of active attacks:



(a) Attacks based on random degrees (b) Planned attacks

Figure 44: The percentage of attacks prevented by each algorithm.

- the random: we choose the Sybil nodes arbitrarily,
- the planned: we first check the previous releases to find the rarest degrees. These degrees should not be very far from the other degrees so they don't form apparent suspicions. We choose the nodes having these degrees in the current graph as Sybil nodes.

The results of the simulation attacks shown in Figure 44 confirms that an edge-DP algorithm doesn't have the potential to prevent the retrieval of the Sybil subgraphs from the published graphs.

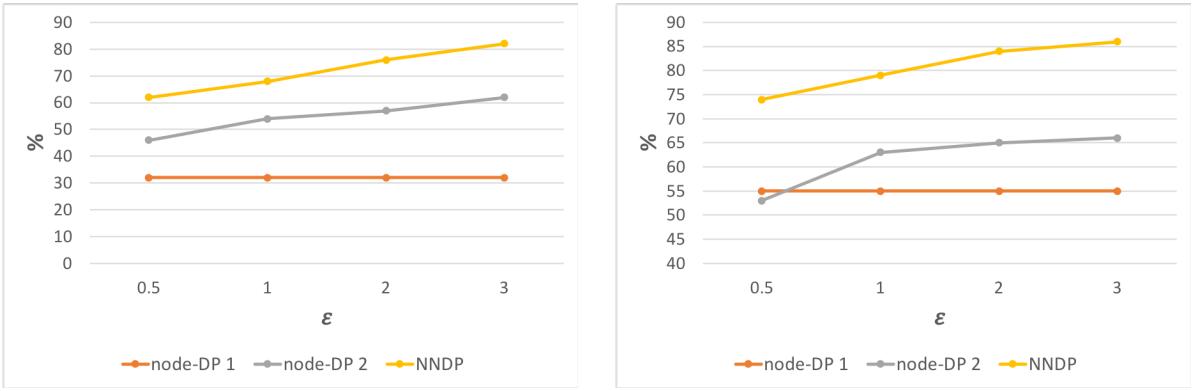
By node-DP 1, we mean the baseline privacy-preserving mechanism based on random removing of nodes and their edges, and node-DP 2 is the second baseline mechanism based on exponential mechanism. The latter surpasses the former significantly in preventing the success of the second step of the attack. NNDP and node-DP have very close performance facing the attacks based on random degrees, while our mechanism outperforms the node-DP in the case of planned attacks. We think this out-performance is because we are studying the previously published graphs similar to the attacker, which gives us a higher potential to detect suspicious nodes than node-DP 2, relying on just the uniqueness of the degree.

3.5 .8.5 Confusion Matrix

In this subsection, we use a confusion matrix to measure the efficiency of the two node-DP and the NNDP mechanisms. We compute the percentage of four categories:

- **True positive:** The nodes are appearing in the published graphs and are innocent nodes.
- **False positive:** The nodes are appearing in the published graphs and are Sybil nodes.
- **True negative:** The nodes removed or hidden and are Sybil nodes.
- **False negative:** The nodes removed or hidden and are innocent nodes.

The edge-DP is not a part of this experiment as it doesn't remove or hide any node. For the other mechanisms, we compute the mean of the percentages of the Sybil nodes hidden or removed in each release (true positive) and the mean of the percentages of innocent nodes



(a) Percentage of hidden or removed Sybil nodes. (b) Percentage of innocent published nodes.

Figure 45: Confusion matrix for the two baseline node-DP and NNDP mechanisms.

published in the releases (true negative). We don't have to compute the two other categories as they are just the complements of these two: the percentage of false-positive is equal to 100 - the percentage of true positive, and the same concept for the false and true negative.

The two charts in Figure 45 show these two means based on the mechanism and the privacy parameter ϵ. As we can notice, the NNDP outperforms the two baseline mechanisms in both categories, which ensures a high utility for the NNDP outputs.

CHAPTER 4

Blowfish Privacy for Transactional Dataset

4.1 Introduction

The emergence of data outsourcing promoted by the services provided by cloud computing, has led to the promulgation of privacy regulations [30] to protect the confidentiality of personal information. These regulations direct data providers to abide by strict rules of anonymization before the release of data containing Personally Identifiable Information (PII). Data providers should anonymize not only the identifying values but also the associations that link individuals to their sensitive values as an adversary may be able to combine his/her background knowledge [143, 151] to information in the released dataset to breach privacy.

A particularly challenging problem for data anonymization is dealing with transactional data. Most anonymization methods assume homogeneous, independent and identically distributed (i.i.d.) data; ¨flattening¨ transactional data to satisfy this model results in wide, sparse data that does not anonymize well with traditional techniques. While there have been some approaches for generalization-based anonymization, bucketization techniques (e.g., Anatomy) pose new challenges. In particular, bucketization provides the opportunity to learn correlations between data items, but also a risk of identifying individuals because of dependencies inferred from such correlations. We present a method that balances these issues, retaining the ability to discover correlations in the data, while hiding dependencies that would enable correlations to be used to link specific values to individuals. We introduce a *correlation anonymization* constraint that ensures correlations do not allow data to be linked to a specific individual, and an *elastic safe grouping* algorithm that meets this constraint while preserving data correlations. We evaluate the utility loss on a transactional rental dataset.

4.2 Related Work

4.2.1 Correlation in literature

The correlation problem in data anonymization has been a topic of interest for some time. The works described in [80, 98, 159] are good examples of attacks that identify correlation problems in anonymization techniques allowing attackers to use their background knowledge or their extracted foreground knowledge to breach privacy.

In [159], the authors consider correlation as foreground knowledge that can be mined from anonymized data. They use the possible worlds model to compute the probability of associating an individual with a sensitive value based on a global distribution. In [80], a Naïve Bayesian model is used to compute association probability. They used exchangeability [12] and DeFinetti's theorem [138] to model and to compute patterns from the anonymized data. In [98], the authors deal with background knowledge that can be mined from the data. In their paper, they focus mainly on what we consider as negative correlations, which limits the ability to handle positive and exposed correlations.

In [154, 11], the authors provide solutions to cope with the correlation problem by either bounding it to the privacy constraint as in [11] or hiding highly correlated values, outliers in

the dataset [154]. Unlike these techniques that are specifically defined for correlations between quasi-identifying attributes and sensitive values in non-transactional datasets, we consider that those correlations, where multiple tuples are related to the same individual, result in possible linking of a particular sensitive values to a particular individual when correlated in the same group or across several groups. Wang et al. propose in [155] a bucketization technique that creates flexible QI-groups based on the importance of the sensitive value. By doing so, the authors reduce the impact of correlations that spread across groups but fail to consider the intra group correlations; correlations between the values in the same group. In a similar work to ours [65], Gong et al. propose a (k, l)-diversity privacy constraint to anonymize a dataset that contains multiple tuples for the same individual. Their technique ensures that there are at least k individuals for an identifying set of sensitive attribute values in addition to l well-represented identifying sets of sensitive attribute values. This is similar to our assumptions, having k distinct individuals and at the same time maintaining l-diverse QI-groups, the only difference is that this technique uses generalization to ensure that these constraints are met.

In [57], the authors propose a knowledge-based sequential anonymization algorithm (KSAA) for privacy preservation when different anonymized views of the same original dataset are published. They present a bottom-up anonymization algorithm, KSAA, that uses generalization to protect against background knowledge attacks. KSAA clusters tuples and generates QI-groups satisfying the privacy model in the current view. It checks, in a second step, if the privacy constraint is satisfied when several views are joined together. In our work and unlike KSAA, we do not hide the correlations we expose them instead and make sure that they cannot be used to link an individual to his/her identifying and sensitive values. Differential privacy [53] approaches release privatized data by adding noise to the released data or query results. The noise is calibrated so that the impact of any individual on the outcome is small relative to the noise added, thus effectively hiding any individual in the noise. Since differentially private mechanisms add independent and identically distributed noise, they are vulnerable against correlations where an adversary is able to filter out the noise While this is a strong privacy definition, for many uses this approach is impractical, as the noise, while typically small, is potentially unbounded. We opt for bucketization, which keeps values intact but provides protection against linking sensitive data to individuals. We extend our previous work [10] to address correlations that explicitly violates the targeted privacy goal while minimizing the loss in data utility.

4.2.2 Privacy approaches and algorithms

Two attack models can be classified: Linkage and probabilistic attacks. As its name indicates, the linkage attacks model occurs when the adversary is trying to relate an individual victim to a record in the published dataset (record linkage), to a sensitive attribute (attribute linkage), or the whole table (table linkage) in the case of Existential Uncertainty.

The second model, called probabilistic attack, aims at providing the adversary with vital information to ensure a significant variation in his/her prior and the posterior of beliefs.

	Record	Attribute	Table	Probalistic
k-anonymity	✓			
(ϵ, m)-anonymity		✓		
l-diversity	✓	✓		
t-closeness	✓			✓
δ-presence			✓	
ESG	✓	✓		✓

Table 9: Privacy techniques and attack models

k-anonymity [151] requires that every tuple in the dataset is indistinguishable in their quasi-identifiers from $k-1$ other tuples by generalizing these attributes to ensure this similarity. This definition can deal with record linking.

Machanavajjhala et al. [111] propose the l-diversity definition relying on distributing the sensitive attributes in a way that is called "well-represented", three definitions of this term were suggested. The simplest one is that in each group containing all the tuples with the same generalized quasi-identifiers, these tuples have at least l different sensitive attributes. In this way, l-diversity deals with attribute linking by avoiding the lack of diversity of sensitive attributes for an equivalence class or a QI-group.

Proximity breach occurs when the adversary may not learn the exact numeric value of the sensitive attribute linked to an individual. Nevertheless, suppose the victim is linked to an equivalence class with all (or the majority) of the sensitive values are in a small interval. In that case, the adversary could estimate a proximate sensitive value for this individual with high confidence. (ϵ, m)-anonymity citejiexing2008 requires that for every sensitive value x in the QI-group, at most $1/m$ other values are in the interval $[x-\epsilon, x+\epsilon]$.

The previous approaches and their extensions focus on the local distribution of each equivalence class, neglecting the importance of global distribution. Thus, t-closeness [95] is proposed to deal with two problems that l-diversity fails to consider. The first one is the semantic closeness of the values. A class of 3 tuples having the same generalized quasi-identifiers is considered 3-diverse if it contains three different locations, at least for the sensitive attribute. If we link Bob to this class and all three locations are in the same city, we have unveiled that Bob was in that city.

Another problem is the distribution of the sensitive values in the whole table versus in one class. For example, if a location appears in 10% of sensitive attributes in the entire dataset and appears two times (50%) in a class of 4 tuples. Then, linking an individual to this class will increase the probability of being in this location from 10% to 50%.

By focusing on the global distribution, t-closeness can confront the challenges of the probabilistic attack. But one objective to publish the table is to analyze the correlation between some QI attributes and the sensitive attribute. Since t-closeness restricts that each QI-group has nearly the same distribution as the distribution of the entire table, the desired goal cannot be achieved. Hiding the safe corrlations is also the disadvantage of the safe grouping appraoch that motivates us to propose the Elastic Safe Grouping.

These privacy approaches are based on the idea that the adversary knows that the victim exists in the dataset, tries to identify him/her, and linking its id to the correct sensitive value. Another type of privacy violations could be when the adversary is uncertain and tries to disclose the existence or absence of an individual in the private dataset. The basic idea of δ-Presence [124] is that the probability for an adversary to identify any individual as being in the anonymized table should be in a range $\delta = (\delta_{min}, \delta_{max})$ that defines the level of trade-off between the utility and privacy.

Our approach, called Elastic Safe Grouping (ESG), relies on Anatomy and l-diverstiy to encounter record and attribute linkages, respectively. Concerning the probabilistic attack, ESG ensures that the global distribution exposes just safe correlations for better utility with the same privacy guarantees of the Safe Grouping. Thus, ESG encounters probabilistic attacks also as we have mentioned in Table 9.

Numerous algorithms and tools were proposed to realize these privacy approaches. For record linkage, Incognito [87] presents several optimal bottom-up algorithms based on Full-domain generalization and Record suppression, and the size of the QI-group is computed according to the rollup property [60]. μ-argus [74] achieves k-anonymity by checking the low-dimensional combinations of identifying values and applying subtree generalization and cell suppression.

For attribute linkage, InfoGain Mondrian [88] is based on Multidimensional Generalization to create a k-anonymous and l-diverse dataset with the consideration of some data mining tasks. Top-Down disclosure.

For table linkage, SPALM and MPALM [124] lean on Full-domain and Multidimensional Generalizations, respectively, and benefit from the anti-monotonicity property of the δ-presence approach to hide the presence of individuals in datasets.

For Probabilistic attack, all the algorithms lies on adding noise as ϵ-Differential Privacy [51], Cross-Training Round Sanitization [33], and $\alpha\beta$ Algorithm [137]. In Section 4.7, we prove that our algorithm satisfies an extension of Differential Privacy called Blowfish privacy, which confirms that our work also can deal with probabilistic attacks.

Also, many tools were proposed to apply the privacy approaches. TIAMAT [42] is a tool for interactive analysis of anonymization techniques, mainly k-anonymity. It allows data publishers to analyze the accuracy and runtime performance of various k-anonymization techniques and find suitable parameter settings for anonymization. ARX Data Anonymization [130] Tool implements a set of techniques that can handle a broad spectrum of data anonymization tasks. It supports many privacy and risk models, methods for transforming data, and methods for analyzing the usefulness of output data.

4.3 Preliminary Definitions

Table 10 summarizes the set of notation used in this chapter. Given a table T with a set of attributes $\{A_1, ..., A_b\}$, $t[A_i]$ refers to the value of attribute A_i for the tuple t. Let U be the set of individuals of a specific population, $\forall u \in U$ we use T_u to denote the set of tuples in T related to

T	a table containing individuals related tuples
t_i	a tuple of T
u	an individual described in T
T_u	a set of tuples related to individual u
A	an attribute of T
A^{id}	an identifying attribute of T
A^s	a sensitive attribute of T
QI_j	a quasi-identifier group
T^*	Anonymized version of table T

Table 10: Notations

the individual u. Attributes of a table T that we deal with in this paper are divided as follows;

Identifier (A^{id}) represents an attribute that can be used (possibly with external information available to the adversary) to identify the individual associated with a tuple in a table. We distinguish two types of identifiers; sensitive and non-sensitive. For instance, the attribute *Security Number* is a *sensitive identifier*; as such it must be suppressed (encrypted). *Non-sensitive identifiers* are viewed as public information and include both direct identifiers such as *User ID* in Figure 6a, and quasi-identifiers that in combination may identify an individual (such as <*Gender, Birthdate, Zipcode*>, which uniquely identifies many individuals.)

Sensitive attribute (A^s) contains sensitive information that must not be linkable to an individual but does not inherently identify an individual. In our example (Figure 6a), the attribute *Location* is considered sensitive and should not be linked to an individual.

4.4 Attack Model

The adversary relies on two types of Knowlege: Foreground and Background. The Foreground Knowledge is based on the correlations that can be found in the published dataset.

An adversary can obtain additional information from the published dataset in the form of global distribution, which can lead to individual privacy breach.

The table in Figure 6b satisfies 3-diversity. Let's say we have 10 Qi-groups containing tuples for *Carl_U*3, in 9 of them we don't have 19.60;35.40 as a sensitive value. They co-exist just in QI_2. The adversary can predict with conficdence of 90% that 19.60;35.40 in QI_2 is not linked to *Carl_U*3. Then the probability of linking this sensitive value to one of the two remaining identities is $PR(v_{19.60;35.40} \mid v_{U1}) = PR(v_{19.60;35.40} \mid v_{U4}) = 0.9 \times 0.5 = 0.45$ which is higher then $1/l$ where $l = 3$ which is considered a privacy breach. With more Background or Foreground Knowledge linked to $U1$ or $U4$, one of these two probabilities may increase to allow the adversary to link one of the two v_{id} with that sensitive value with high certainty.

The same learning could be done on the relation of a QI attribute with sensitive attribute in an attempt to link the id of the tuple containing the QI attribute with one of the sensitive attributes. In our example this linkage attack cannot be done, but if we take a dataset of patients

and their diseases, the adversaries can easily link a quasi-identifier or a non-sensitive value to a sensitive value using Background or Foreground Knowledge. For example, breast cancer is related with females in 0.99% of the cases. Heart attacks is related to men with 70% of cases. The flu is very common in US in February and in Australia in August. The osteoporosis is related with women in 80% of cases. 98% of people diagnosed with Parkinson are above 40 years old. The adversary can get this knowlege by doing some research on some medical websites or can get close numbers from doing some data mining from the published dataset. Thus they will be able to exclude the link between a tuple containing the age, the sex and the country of an individual with some sensitive values and increase the probability of this link between that tuple and some other sensitive values.

From this example, the knowledge can be modelled in two formats. First, as a set of tuples $< u, v_s, conf(u, v_s) >$ where u is the identification of an individual, v_s is a sensitive attribute and $conf(u, v_s)$ is a confidence score that links u to v_s. And the second format is $< v_{ns}, v_s, conf(v_{ns}, v_s) >$ where v_{ns} is a non-sensitive attribute. For example, $< U1, 19.60; 35.40, 0.45 >$ means that the adversary can link $Roan_U1$ and the location 19.60; 35.40 with a 45% confidence. While $< Male, HearAttack, 0,7 >$ means that the adversary can link the non-sensitive attribute 'Male' with the sensitive attribute 'Heart Attack' with 70% confidence.

From This example also we can see how the adversary take advantages from *inter-group dependencies* to form his/her Foreground Knowledge. The problem is that the majority of the privacy techniques focus on the local distribution which means the distribution of QI and sensitive attributes in the QI-groups to ensure that no *intra-group dependencies* will let an adversary get important Foreground Knowledge from the group. But it's rare to find a technique dealing with *inter-group dependencies* in the global distribution.

Safe grouping hides the correlations to forbid the adversary from gaining Foreground Knowledge. Our motivation in this work is to propose an approach that allows the adversary to expose the correlations and prevent him/her from taking advantage of these correlations. The idea is that if some individuals co-exist in one QI-group, then, in any other QI-group, all of them will also co-exist, or none of them will exist. In that way, it is possible to gain Foreground Knowledge from the correlations between the non-sensitive and sensitive attributes. But, at the same time, it's hiding the correlations between the individuals and the sensitive attribute. Suppose we take the same example discussed in this section. Let's say that the three ids $U1$, $U3$, and $U4$ co-exist in 10 QI-groups where the location 19.60;35.40 is mentioned in 9 of them. This foreground Knowledge can't help in linking the location with one of the ids. These rules are not applied to the non-sensitive attributes, than the correlations between them and the sensitive attributes could be exposed for a better utility.

4.5 Strength and Weaknesses of Safe Grouping

Safe grouping ensures that an individual's tuples are grouped in one and only one QI-group that is at the same time ℓ-diverse, respects a minimum diversity for identifying attribute values, and all subsets T_u for an individual u in QI_j have an equal number of tuples. The formal definition

of safe grouping is given as follows.

Definition 5 (Safe Grouping). *Given a table T, safe grouping is satisfied on T iff*

1. *$\forall u \in U$, the subset T_u of T is contained in one and only one quasi-identifier group QI_j ($1 \leq j \leq m$) such that QI_j respects ℓ-diversity and contains at least k subsets $T_{u_1}, ..., T_{u_k}$ where $u_1, ..., u_k$ are k distinct individuals of the population and,*

2. *$Pr(u_{i_1}|QI_j) = Pr(u_{i_2}|QI_j) \leq 1/\ell$ where*
   $u_{i_1}, u_{i_2}, i_1 \neq i_2$ are two distinct individuals in QI_j with ($1 \leq i \leq k$) and $Pr(u_i|QI_j)$ is the probability of u_i in QI_j.

This definition gives Safe Grouping the ability to hide the correlations that could link an identifying value to a specific sensitive value. For example, the adversary has Background Knowledge that Roan has rented ten cars last year, 9 of them were from the same location 20.09,45.11. If the User ID $U1$ exists in 10 Qi-group wherein 9 of them the location 20.09,45.11 coexists with $U1$, the adversary could conclude that $U1$ is the User ID of Roan with high probability. Safe Grouping suggests grouping all the tuples of Roan with the tuples of Lisa and Bob, for example, in the same QI-group. In that way, the adversary could not benefit from the Background Knowledge because they could not find an inter-group correlation between $U1$ and 20.09,45.11, leading to link $U1$ to Roan. In a case where 20.09,45.11 is very rare in other Qi-groups, the adversary may predict that the tuples of Roan exist in this specific QI-group. But they cannot link Roan to $U1$ with a probability higher than $1/3$ because they could not distinguish it from the User IDs or Lisa and Bob.

We now study the strength and weaknesses of safe grouping under two scenarios. In the first, we assume that the original data is anonymized and shared with others for analysis purposes. An adversary will try to infer individuals' sensitive values from the anonymized dataset. We assume that the adversary knows that a particular significant correlation exists in the dataset and wants to use it to link the individual to his/her identifying and sensitive values in the anonymized dataset to gain greater confidence in an individual's value that is inherent in simply knowing the correlation.

In the second, we assume that the adversary is able to learn significant correlations and use them to link an identifying value to a particular individual. We use β_u to denote the background knowledge of an adversary related to the correlation of u's tuples in the dataset. For instance, $Pr(u, v_{id}|T^*, \beta_u)$ represents the probability of linking u to his/her identifying value v_{id} given, the anonymized dataset T^*, and the adversary's knowledge of the correlation of u's related tuples in T^*.

4.5.1 Assumption 1: adversary knows the correlations

The adversary knows that the identifying value of an individual u correlates with some other values in the dataset; e.g., individual u is using the renting service frequently, or more seriously when the adversary knows the correlation between the identifying value of u and a particular

sensitive value; e.g., an individual u is continuously renting from location v_s. In both cases, a privacy breach might occur if the adversary can link u to his/her corresponding identifying value v_{id} in the dataset. This could happen when the same identifying and sensitive values correlate across QI-groups.

We demonstrate, in the following, that safe grouping protects against such disclosure.

Lemma 3 (Membership Disclosure)**.** *Given an anonymized dataset T^* that satisfies safe grouping, an adversary cannot link an individual u, $\forall u \in U$, to his/her identifying value v_{id} in T^* with a probability $Pr(u, v_{id}|T^*, \beta_u)$ greater than $1/k$.*

Proof. Since safe grouping is used, the identifying value v_{id} of individual u exists in one and only one QI-group, QI. The probability of linking u to v_{id}, $Pr(u, v_{id}|T^*, \beta_u)$ is no more than $Pr(u, v_{id}|QI, \beta_u)$, which is equal to $\frac{|T_u|}{|QI|}$, $Pr(u, v_{id}|QI, \beta_u) = \frac{|T_u|}{|QI|}$ where $\forall t \in T_u, t[A^{id}] = v_{id}$. In addition, there are at least $k-1$ individuals grouped in QI such that $|QI| \geq \sum_{i=1}^{k} |T_{u_i}| = k \times |T_u|$ where, according to safe grouping, $\forall u_1, u_2$, two distinct individuals in QI, the number of tuples for u_1, $|T_{u_1}|$ is equal to the number of tuples for u_2, $|T_{u_2}|$, thus the probability of linking u_1 or u_2 to their identifying values $v_{1_{id}}$ and $v_{2_{id}}$ respectively is at most equal to $1/k$. □

Alternatively, let us now assume a stronger adversary; an adversary capable of linking an individual u to his/her identifying value v_{id} in T^*. If T^* satisfies safe grouping, the probability of linking u to his/her sensitive value v_s is at best equal to $1/\ell$.

Lemma 4 (Attribute Disclosure)**.** *Given an anonymized table T^* that satisfies safe grouping, an adversary with the ability to link an individual u to his/her identifying value v_{id} in T^*, cannot link u to his/her sensitive value with a probability $Pr(u, v_s|T^*, \beta_u)$ greater than $1/\ell$.*

Proof. $Pr(u, v_s|T^*, \beta_u)$ can be written as $Pr(u, v_{id}|T^*, \beta_u) \times Pr(v_{id}, v_s|T^*, \beta_u)$ where, $Pr(u, v_{id}|T^*, \beta_u)$ is the probability of linking an individual u to the identifying value v_{id} assuming knowledge of the correlation β_u and $Pr(v_{id}, v_s|T^*, \beta_u)$ is the probability of linking the identifying value v_{id} to the sensitive value v_s. $Pr(u, v_{id}|T^*, \beta_u)$ is equal to 1 since the adversary is able to link u to his/her identifying value. Safe grouping, however, groups all of the individual's tuples in one and only one QI-group, and thus $Pr(u, v_s|T^*, \beta_u)$ is re-written as $Pr(v_{id}, v_s|QI_j) = \frac{c_j(v_{id}, v_s)}{c_j(v_{id})}$. Now, given that safe grouping separates the dataset, in a similar way to anatomy, into two tables T_{QIT} and T_{SNT} where only the link between the tuples in the T_{QIT} and T_{SNT} is anonymized, $Pr(v_{id}, v_s|QI_j)$ is computed as $\frac{c_j(v_{id}) \times Pr(t, v_s|QI_j)}{c_j(v_{id})}$ where $Pr(t, v_s|QI_j)$ is the probability of linking the tuple t for which $t[A^{id}] = v_{id}$, to the sensitive value v_s. This leads to a probability $Pr(t, v_s|QI_j) = \frac{c_j(v_s)}{|QI_j|}$ that is less than or equal to $1/\ell$.

□

4.5.2 Assumption 2: adversary learns the correlations

The adversary learns the correlations from the anonymized table; e.g., an individual's identifying value v_{id} correlates with location v_s. For instance, this would happen in an anatomized table when both, the identifying value and the sensitive value correlate across

several QI-groups (e.g., in Figure 6b, the identifying value U_1 correlates with location 20.09; 45.11 in QI_1, QI_2, and QI_3). A privacy breach might occur if the adversary can use the learned knowledge to link the identifying value v_{id} to an individual u.

Works described in [159, 80, 98] provide an in-depth study on how to estimate the strength of the association between sensitive and non-sensitive values in post-anonymization based on their correlations. They demonstrated that some *significant correlations* in the data could be measured (i.e., estimating the probability of associating a quasi-identifying value with a sensitive value in an anonymized table) as such the resulting score would exceed the allowed privacy threshold.

In a dataset that satisfies safe grouping, none of these significant correlations can be learned from the dataset where $Pr(v_{id}, v_s) = Pr(v_{id}, v_s|QI_j) = 1/\ell$. While this is a good indicator from a privacy perspective, it is, however, expensive in terms of utility. Safe grouping and other proposed techniques [143, 151, 95, 53], unfortunately, sacrifice some of the data utility by losing these correlations for the sake of privacy.

In the next section, we show how sparing some of these correlations; keeping them exposed, can be achieved without risking data privacy.

4.6 Exposing Safe Correlations

4.6.1 Safe correlations

A significant correlation is considered safe if an adversary cannot use his/her knowledge of this correlation (or his/her ability to learn it from the anonymized table) to link an individual u to his/her identifying value. Formally:

Definition 6 (Safe Significant Correlation)**.** *A significant correlation between an identifying value v_{id} and a sensitive value v_s is safe if $\forall u \in U$, $Pr(u, v_{id}|T^*) \leq 1/k$ and $Pr(v_{id}, v_s|T^*) \leq 1/\ell$.*

In other words, a correlation between an identifying and sensitive value is safe if both the probability of a user having that particular identifying value is low, and the correlation between the identifying and sensitive values is below $1/\ell$.

While exposing significant correlations results in better utility for aggregate analysis, it tends to be difficult to achieve without threatening individuals' privacy in a rigid bucketization technique. In fact, the use of a bucketization technique such as anatomization or even safe grouping, creates QI-groups that are either unsafe; exposing significant correlations to the adversary, or of large sizes reducing the utility of the dataset.

In the following, we define a privacy constraint that allows us to expose significant correlations and ensure their safety by creating flexible QI-groups in a way that preserves both privacy and utility. The basic idea is that if an identifying value in a particular QI-group, appears multiple times in several QI-groups in T^*, then there are $k-1$ identifying values that must appear the same number of times. This ensures that the correlation known to the adversary

does not "single out" a specific identifying value. Our Correlation Anonymization constraint is formally defined as follows:

[Correlation Anonymization] Let T^* be a anonymized table, and $\mathcal{D}(A^{id})$ be the domain of values of A^{id}, $\forall QI_j \in T^*$ for $(1 \leq j \leq m)$, we say that a significant correlation between an identifying value v_{id} and a sensitive value v_s is safely anonymized in T^* iff

1. $\forall v_{id} \in \mathcal{D}(A^{id})$, if $v_{id} \in \pi_{A^{id}} QI_1 \cap ... \cap \pi_{A^{id}} QI_j$, $\exists V_{id} \subseteq \mathcal{D}(A^{id})$, a set of identifying values, such that $V_{id} = \{v_{id}, v_{id_1}, ..., v_{id_{k-1}}\}$ and $V_{id} = \pi_{A^{id}} QI_1 \cap ... \cap \pi_{A^{id}} QI_j$. In other words,

$$\pi_{A^{id}} QI_1 \cap ... \cap \pi_{A^{id}} QI_j = \begin{cases} V_{id} & \text{if } \exists v_{id} \in \pi_{A^{id}} QI_1 \\ & \cap ... \cap \pi_{A^{id}} QI_j \\ \emptyset & \text{otherwise} \end{cases} \quad (1)$$

2. $\forall v_{id_1}, v_{id_2}, Pr(v_{id_1}|QI_j) = Pr(v_{id_2}|QI_j)$ where $Pr(v_{id_i}|QI_j)$ is the probability of having v_{id_i} in QI_j, and

3. $\forall v_{id} \in \mathcal{D}(A^{id}), Pr(v_{id}, v_s|T^*) \leq 1/l$.

User ID	Vin Number	GID
Bob_U5	05d7f4f9496*	1
Elyse_U2	0038da44c64*	1
*	0036153c476*	1
*	0038da44c64*	1
Roan_U1	0061d4a8248*	2
Lisa_U4	0e352814d34*	2
Carl_U3	0038da44c64*	2
Roan_U1	05d7f4f9496*	3
Lisa_U4	000cf44c9b3*	3
Carl_U3	0061d4a8248*	3
Roan_U1	0036153c476*	4
Lisa_U4	000cf44c9b3*	4
Carl_U3	0e352814d34*	4

GID	Location
1	19.10;38.13
1	19.72;33.96
1	19.29;36.15
1	19.72;33.96
2	20.09;45.11
2	19.60;35.40
2	19.72;33.96
3	20.09;45.11
3	19.47;43.71
3	19.57;44.70
4	20.09;45.11
4	19.60;35.40
4	19.57;44.70

Figure 46: Correlation anonymization example

Correlation anonymization ensures that the intersection of any two or more QI-groups on their identifying attribute A^{id}, yields either a set of k identifying values or an empty set. This is achieved while preserving the uniform distribution of individuals' related tuples in the QI-groups to cope with the inter-dependencies in each of the QI-groups (as in safe grouping) and diversifying the sensitive values in the QI-group to prevent linking non-sensitive to sensitive information.

Lemma 5. *Correlation anonymization privacy constraint exposes only safe correlations in an anonymized table T^*.*

Proof. $\forall u \in U$, $\forall v_{id}, v_s$, an identifying value and a sensitive value in T^*, if a significant correlation exists between v_{id} and v_s in T^*, then v_{id} and v_s either correlate in the same QI-group or across QI-groups. Given that the correlation anonymization constraint is satisfied on T^*, if v_{id}, and v_s correlate in the same QI-group, there exists $k-1$ other identifying values that correlate with v_s since each and every QI-group in T^* contains at least identifying values for k individuals. Therefore $Pr(u, v_{id}|T^*) \leq 1/k$ and $Pr(v_{id}, v_s|T^*) \leq 1/l$ due to bucketization. Alternatively, if v_{id} and v_s correlate across QI-groups, e.g., $\forall QI_i, QI_j$, QI-groups in T^*, correlation anonymization guarantees that $\forall v_{id} \in \pi_{A^{id}} QI_j \cap \pi_{A^{id}} QI_j$, $\exists V_{id}$ such that, $v_{id} \in V_{id}$, where V_{id} is a set containing $k-1$ identifying values that correlate with v_{id} across the QI-groups in T^*, and thus $Pr(u, v_{id}|T^*) \leq 1/k$ and $Pr(v_{id}, v_s|T^*) \leq 1/l$ due to bucketization.

□

Figure 46 is an example of an anonymized table T^* that respects the correlation anonymization privacy constraint. Let us assume that an adversary knows that the individual *Roan* rents frequently from location 20.09; 45.11. We can notice that such a correlation is exposed; location 20.09; 45.11 is correlated across QI-groups QI_2, QI_3, and QI_4 in T_{SNT}. The same applies for *Roan_U*1 that is correlated in QI-groups QI_2, QI_3, and QI_4 in T_{QIT}. However, since the privacy constraint is satisfied, these same correlations apply equally to two other identifying values, *Lisa_U*4 and *Carl_U*3 making $Pr(Roan, Roan\_U1|T^*, \beta_{Roan}) \leq 1/3$;

the only reason the adversary can attribute this to Roan is prior knowledge.

4.6.2 Elastic safe grouping (ESG)

We present in this section an elastic safe grouping algorithm (ESG) that achieves correlation anonymization. As its name implies, the algorithm simulates an elastic behavior where individuals' tuples are grouped, spread over several *sub-groups*, and then merged again to create ℓ-diverse QI-groups. Technically, the algorithm is based on a divide-and-conquer strategy; dividing buckets composed of at least k identifying values into sub-groups to expose safe correlations, and merging (if necessary) these sub-groups to create ℓ-diverse QI-groups.

The algorithm breaks down into three main stages. In the first stage, the table is divided into buckets of $\delta \geq k$ identifying values (i.e., in a similar manner to the safe grouping algorithm described in [9, 10]). By doing so, we group tuples of each of the identifying values in one and only one bucket along with tuples of at least $k-1$ other identifying values. In the second stage, each bucket is divided into sub-groups based on the identifying value with the minimum number of tuples in the bucket. This reduces the size of the QI-groups, exposing safe correlations since each identifying value correlates with at least $k-1$ other identifying values. In the final stage, ℓ-diverse QI-groups are created. Sub-groups that are not ℓ-diverse are merged together to create ℓ-diverse QI-groups.

Algorithm 2: Elastic safe grouping algorithm

Require: T, k, ℓ
Ensure: $T^* = \{T_{QIT}, T_{SNT}\}$
1: Hash the tuples in T by their A^{id} values, $\{T_u\}$
2: Sort the set of $\{T_u\}$ based on their number of tuples in an ascending order # $\{T_u | \forall T_{u_i}, T_{u_j}, i \leq j, |T_{u_i}| \leq |T_{u_j}|\}$
3: $\delta := max(k, l)$
4: $z := \frac{|U|}{\delta}$
5: *QI-groups*[] := {}
6: *sub-groups*[] := {}
7: **for** $i := 0$ **to** z **do**
8: Create a Bucket B_i of δ A^{id} values from $\{T_u\}$
Remaining tuples are kept in the last bucket
9: *QI-groups* $\leftarrow Divide(B_i)$
10: **end for**
11: **for** each QI_j **in** *QI-groups* **do**
12: **for** each tuple $t \in QI_j$ **do**
13: insert tuple $(t[A_1], ..., t[A], ...t[A_m], j)$ into T_{QIT}
14: **end for**
15: **for** each random value v_s of $A^s \in QI_j$ **do**
16: insert tuple (j, v_s) into T_{SNT}
17: **end for**
18: **end for**

1: **function** DIVIDE(Bucket B) # takes a bucket B of δ identifying values and returns a set of *QI-groups*
2: *QI-groups*[] := {}
3: *tmp*[] := {}
4: $min := |T_{u_0}|$ # T_{u_0} since the elements in the bucket B are sorted
5: $c := 0$
6: **while** $c < min - 1$ **do**
7: *sub-group*[] := {}
8: **for** each T_u **in** B **do**
9: remove a random tuple t_u from T_u
10: *sub-group* $\leftarrow t_u$ # add t_u to the *sub-group*
11: **end for**
12: MergeOrCreate(*sub-group*)
13: $c := c + 1$
14: **end while**
15: Ensure uniform distribution of identifying values in B
16: MergeOrCreate(B)
17: Suppress remaining tuples in *tmp*
return *QI-groups*
18: **end function**

Algorithm 2 describes the core of ESG. It takes a table T, two privacy constraints, k (number of identifying values) and ℓ, and returns an anonymized table T^*. In the first two steps, the algorithm hashes the table based on the identifying values and sorts it in an ascending order starting from the value that has the minimum number of tuples. In Step 7 to 11, the algorithm creates z buckets, divides each bucket into *sub-groups*, and creates from these *sub-groups*, QI-groups that respect the ℓ-diversity privacy constraint. Once QI-groups are identified, the algorithm separates in Step 11 to 18, the table T into two distinct tables T_{QIT} and T_{SNT} and inserts tuples of each of the created QI-groups into T_{QIT} and T_{SNT} accordingly while randomly shuffling the sensitive values.

4.6.2.1 Dividing buckets

The Divide function divides a bucket B, in Step 6 to 14, into sub-groups such that the total number of created sub-groups depends on *min*, which is the number of tuples of the first identifying value in the bucket $|T_{u_0}|$. T_{u_0} is the set that contains the least number of tuples since individuals' tuples in each of the buckets are sorted in an ascending order. A sub-group is formed in Step 8 to 11, by randomly removing a tuple t_u from each individual's related tuples T_u in the bucket B and adding it to *sub-group*.

The Divide function ensures in Step 14, that the remaining identifying values in the bucket B are uniformly distributed. That is hiding, encrypting or suppressing some identifying values to ensure that the probability of linking an individual u to his/her identifying value is equal to the probability of linking u to any other identifying value in the QI-group. The Divide function creates after that the last *sub-group*. For example, let us assume that a bucket B contains identifying values of three distinct individuals, u_0, u_1, and u_2 having $|T_{u_0}| = 2$, $|T_{u_1}| = 3$, and $|T_{u_2}| = 3$. $B = \{T_{u_0} = \{t_{u_0}, t_{u_0}\}, T_{u_1} = \{t_{u_1}, t_{u_1}, t_{u_1}\}, T_{u_2} = \{t_{u_2}, t_{u_2}, t_{u_2}\}\}$. The function divides the bucket into two sub-groups, $sub\text{-}group_1$ and $sub\text{-}group_2$ since $|T_{u_0}| = 2$. It creates first $sub\text{-}group_1 = \{t_{u_0}, t_{u_1}, t_{u_2}\}$, anonymizes in Step 15 some of the identifying values remaining in the bucket B to ensure their uniform distribution, and creates sub-group$_2 = \{t_{u_0}, t_{u_1}, t^*_{u_1}, t_{u_2}, t^*_{u_2}\}$. t^*_u is used to denote a tuple in which the identifying value $v_{id} = t_u[A^{id}]$ is anonymized.

```
1:  procedure MERGEORCREATE(sub-group)
2:      if sub-group is ℓ-diverse then
3:          QI-groups ← sub-group
4:      else
5:          tmp := tmp ⋃ sub-group
6:          if tmp is ℓ-diverse then
7:              QI-groups ← tmp
8:              tmp[] := {}
9:          end if
10:     end if
11: end procedure
```

4.6.2.2 Creating QI-groups

QI-groups are created using the procedure MergeOrCreate, which takes a *sub-group* and verifies its ℓ-diversity. In Step 5 in the MergeOrCreate procedure, *sub-groups* that do not meet the ℓ-diversity constraint are stored in a temporary buffer tmp. This process is repeated sequentially for the *sub-groups* in the bucket until tmp satisfies ℓ-diversity. Before moving to the next bucket, if ℓ-diversity is still not met in tmp, all the tuples in the buffer are then suppressed. This process increases the size of the QI-group for the sake of privacy but also retains as many tuples as possible.

Let n be the total number of tuples in T, ESG produces m QI-groups such that, $m \leq \frac{n}{\delta}$, where $\delta = max(k, l)$.

Proof. The Divide function in ESG produces at most $\sum_{i=1}^{z} min(B_i)$ QI-groups (assuming that all

QI-groups are ℓ-diverse) where, $min(B_i)$ is a function that returns the minimum number of tuples of an individual in Bucket B_i. Now, if $\sum_{i=1}^{z} min(B_i) \leq \frac{n}{\delta}$, then

$$\begin{aligned}
\sum_{i=1}^{z} min(B_i) &\leq \frac{n}{|U|} \times \frac{|U|}{\delta} \\
\sum_{i=1}^{z} min(B_i) &\leq \frac{n}{|U|} \times z \\
\frac{1}{z} \times \sum_{i=1}^{z} min(B_i) &\leq \frac{n}{|U|} \\
\frac{1}{z} \times \sum_{i=1}^{z} min(B_i) &\leq \frac{1}{|U|} \times \sum_{y=1}^{|U|} |T_{u_y}|
\end{aligned}$$

The last statement in our equation compares two values in a cumulative moving average since $\{min(B_i)\}$, $(\forall i, 1 \leq i \leq z)$ is a subset of $\{|T_{u_y}|\}$, $\forall y, 1 \leq y \leq |U|$, $\{min(B)\} \subseteq \{|T_u|\}$ and $\sum_{i=1}^{z} min(B_i) \leq \sum_{y=1}^{|U|} |T_{u_y}|$. Thus, for $|U| \geq z$, the value $\frac{1}{|U|} \times \sum_{y=1}^{|U|} |T_{u_y}|$ is continuously greater than $\frac{1}{z} \times \sum_{i=1}^{z} min(B_i)$. □

4.6.3 The anonymization process

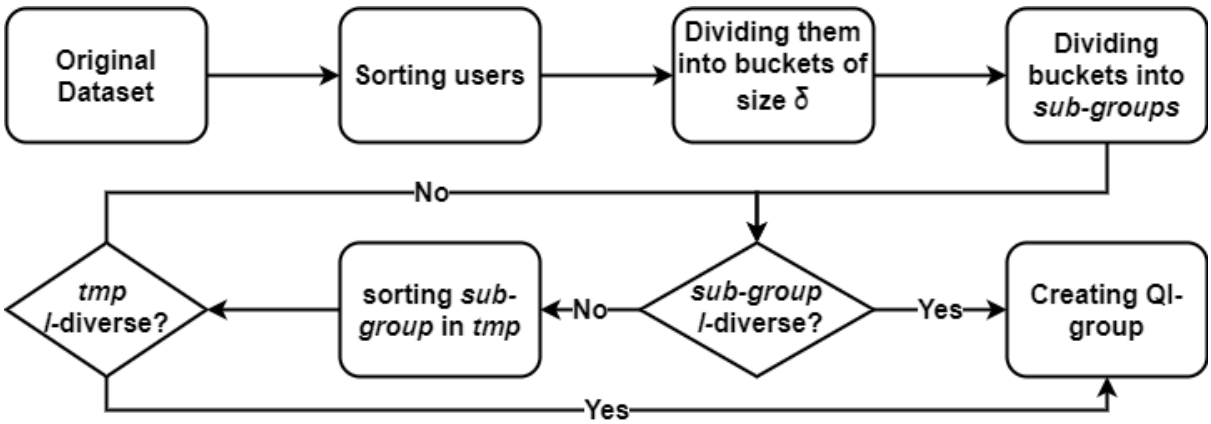


Figure 47: Process Diagram of the elastic safe grouping

In Figure 47, we explain the process of our algorithm. We can divide it to three stages. In the first one; the users are sorted in descending order based on the number of tuples then divided into buckets of δ users each, where $\delta = max(K, l)$, for example in Table 48a, $\delta = 3$ as we can see in Figure 48a.

In the second stage, each bucket is divided into *sub-groups* based on the identifying value with the minimum number of tuples in the bucket. For example, in Figure 48c, 3 *sub-groups* of size 3 where each *sub-group* contains one tuple from u_1, u_3 and u_4.

In the third stage, we check if the *sub-group* is l-diverse. If it's not, then the *sub-group* is sorted in *tmp* sequentially till *tmp* is l-diverse. If all the *sub-groups* related to a bucket are treated and the *tmp* is not l-diverse than all its tuples are suppressed. Finally, each l-diverse QI-group is created wether directly from *sub-groups* or from the buffer *tmp*.

4.6.4 Analysis of elastic safe grouping algorithm

Given n as the total number of tuples in the table, the number of distinct individuals $|U|$ in T, $|T_u|$, which is the number of tuples per individual u, the following estimates the complexity of ESG.

The cost of the sorting step (Step 2 in Algorithm 1) is $n \times log(n)$. In the Divide function, we estimate the value of min (Step 6 in the Divide function) to be the average number of tuples per individual denoted by $\frac{n}{|U|}$ (see property 4.6.2.2). The cost of MergeOrCreate procedure is in a worst case, equal to δ, which is the cost of merging a QI-group of size δ with $temp$ (Step 5 in the MergeOrCreate procedure). So now, to sum up, the cost of ESG is as follows:

$$\begin{aligned}\mathcal{T}(n) &= n \times \log n + z \times (\tfrac{n}{|U|} \times 2\delta) \\ &= n \times \log n + \tfrac{|U|}{\delta} \times (\tfrac{n}{|U|} \times 2\delta) \\ &= n \times \log n + 2n\end{aligned}$$

4.7 Achieving Blowfish Privacy

In this section, we prove that our mechanism achieves Blowfish Privacy [71], an extension of Differential Privacy [55]. Differential and Blowfish Privacy are widely used privacy techniques in many domains as data mining[**?**], deep learning [**?**], image classification [**?**], location [**?**], graphs, social networks and Call Details Records anonymization [**?** 3]. These two approaches can confront probabilistic attacks. We benefit from this proof as a confirmation that our algorithm has the same ability.

The key difference between the two definitions is that the set of neighbors in Blowfish depends on a policy P that determines the set of discriminative pairs as well as the constraints known about the database.

Blowfish Privacy required that a privacy-preserving mechanism has the potential to hide the existence or non-existence of a row in a dataset. To prove this ability:

- we define a range of outputs S,
- we list all the possible scenarios of adding (or deleting) one row to the dataset,
- we compute the probability $Pr[ESG(D) \rightarrow S]$ that the output of our mechanism applied on the original dataset is in the range S, then the same probability but applied on D',
- we prove the existence of an upper bound ϵ for the distance of these two probabilities and prove that this upper bound holds up for all the scenarios of the row addition (or deletion).

In the coming subsection, we list all the possible scenarios of adding a tuple to the dataset. These scenarios focus on the status of the bucket containing the tuples of the same individual related to the added tuple. We compute an upper bound ϵ and prove it holds up for all the cases.

4.7.1 Determining the privacy paramater ϵ

In this subsection, we show that our mechanism respects the Blowfish Privacy inequation shown in Equation 1. For this reason, we have to determine an upper bound to the possible effect on the output that takes place when adding or removing one tuple from the original dataset.

Let's say we have two datasets $D = \{t_1, ..., t_n\}$, $D' = \{t_1, ..., t_{n+1}\}$ where t_i is a tuple containing quasi-identifier and sensitive attributes, t_i is connected to one of the users in the Universe and t_{n+1} is connected to one of those appearing in D.

To assume that our mechanism ESG respects the differential private inequation, we have to compute the privacy parameter ϵ in the following inequation:

$$\frac{Pr[ESG(D) \rightarrow S]}{Pr[ESG(D') \rightarrow S]} \leq e^{\epsilon} \tag{1}$$

It means that if the QI-group QI_j of $ESG(D)$ is formed of 4 tuples: 2 belonging to one user and the other 2 belonging to 2 different users, then the QI-group QI_j of any output in the range S should have the same structure even if the tuples don't belong to the same users as in QI_j of $ESG(D)$. The structure doesn't take into consideration the excluded identifying values in these groups (Excluded identifying values are those switched by *).

Therefore, the range of S is based on the output of ESG(D), thus, it's obvious that $Pr[ESG(D) \rightarrow S] = 1$. Then, to achieve the upper bound e^{ϵ}, we should compute the minimal value of $Pr[ESG(D') \rightarrow S]$.

First, we assume that we are adding a tuple $\{t_{n+1}\}$ to D'. This tuple is definitely added to one of the buckets as shown in Figure 48. The process of dividing the buckets into *sub-groups* is based on the identifying value with the minimum number of tuples $\min_{u \in B_i}(\mid T_u \mid)$ for all users in the bucket B_i. Then $\min_{u \in B_i}(\mid T_u \mid)$ is the factor that controls the number of *sub-groups* created from each bucket, for example in Table 48a, B_1 and B_2 will be divided into 3 and 1 *sub-groups* respectively.

The first step in computing $Pr[ESG(D') \rightarrow S]$ and its lower bound is to calculate the probability that adding t_{n+1} to D' doesn't affect $\min_{u \in B_i}(\mid T_u \mid)$ of any bucket.

In Table 48a, 3 *sub-groups* will be created from B_1 and 1 *sub-group* from B_2. While the two remaining tuples from U_5 will be excluded ($A^{id} \rightarrow u^*$) as we can see in Table 48b .

In Table 48c, B_1 will create also 3 *sub-groups*. B_1 has more than one individual (in this example, all of them) with the minimum number of tuples $\min_{u \in B_1}(\mid T_u \mid) = 3$. In this case, regardless of the individual related to t_{n+1}, $\min_{u \in B_1}(\mid T_u \mid)$ will remain 3, because at least one individual still has just 3 tuples. Thus, as we can see in 48d, the only change is in the number of u^* which doesn't affect the possibility of the output to be in range S.

In Table 48e, the new tuple is related to U_5 which requires to sort the individuals again based on the number of their tuples. After sorting, u_5, u_1 and u_4 are in B_1 while u_4 and u_2 are in B_2. Neither $\min_{u \in B_1}(\mid T_u \mid)$ nor $\min_{u \in B_2}(\mid T_u \mid)$ were affected by adding t_{n+1} to u_5. For this reason, as Table 48f shows us, the structure of the output is similar to that of $ESG(D)$ even if the identity

Users	u_1	u_3	u_4	u_5	u_2
|Tuples|	3	3	3	3	1
Buckets	B_1			B_2	

(a) Buckets of dataset D.

u_1	u_3	u_4	u_5	u_2	u^*
3	3	3	1	1	2

(b) Number of tuples for each individual in S.

Users	u_1	u_3	u_4	u_5	u_2
|Tuples|	**4**	3	3	3	1
Buckets	B_1			B_2	

(c) First case of D'.

u_1	u_3	u_4	u_5	u_2	u^*
3	3	3	1	1	3

(d) Output of the first case of D'.

Users	u_5	u_1	u_3	u_4	u_2
|Tuples|	**4**	3	3	3	1
Buckets	B_1			B_2	

(e) Second case of D'.

u_5	u_1	u_3	u_4	u_2	u^*
3	3	3	1	1	3

(f) Output of the second case of D'.

Users	u_1	u_3	u_4	u_5	u_2
|Tuples|	3	3	3	3	**2**
Buckets	B_1			B_2	

(g) Third case of D'.

u_1	u_3	u_4	u_5	u_2	u^*
3	3	3	**2**	**2**	1

(h) Output of the third case of D'.

Figure 48: Four Scenarios of adding a new tuple belonging to one of the users.

of the individuals has changed. Thus the output is in S.

In Table 48g, t_{n+1} is related to u_2 which increments $\min_{u \in B_2}(\mid T_u \mid)$ from 1 to 2. Thus, the structure of the output will be affected which put it outside the range S.

In conclusion, based on the third, we should compute the probability Pr_{min} that t_{n+1} doesn't belong to an individual having $\min_{u \in B_i}(\mid T_u \mid)$ tuples in any bucket unless another individual have the same number of tuples in the same bucket.

For each bucket B_i, the probability that t_{n+1} doesn't affect its $\min_{u \in B_i}(\mid T_u \mid)$ is:

$$pr_i = \begin{cases} 1 & \text{if } \mid T_{U_a} \mid = \mid T_{U_b} \mid = \min_{u \in B_i}(\mid T_u \mid) \\ \frac{|U|_i - 1}{|U|_i} & \text{Otherwise} \end{cases}$$

where $\mid U \mid_i$ is the number of individuals in B_i. The first condition presents the case where more than one individuals have the minimum number of tuples $\min_{u \in B_i}(\mid T_u \mid)$; otherwise, the second condition is met if t_{n+1} is related to any individuals in B_i except the last one.

Then, we can compute Pr_{mnt} as:

$$Pr_{mnt} = \sum_{i=1}^{|B|} pr_i$$

where $\mid B \mid$ is the number of buckets.

The second step in the algorithm is to make sure that the *sub-group* is l-diverse to create a QI-group. Otherwise, the *sub-group* might merge with another *sub-group* to create a QI-group. Thus, we have to compute the probability Pr_{div} that the *sub-group* sg containing t_{n+1} in D' will create the same QI-group that it has created in D. This event could be sub-divided into 4 events:

- Pr'_{div}: the case were sg isn't l-diverse in D, and still not l-diverse in D'. The minimal value of this probability could be reach when, for example in D, sg merge with another

sub-group sg_1 to become l-diverse and create a QI-group while in D' it doesn't merge or merge with a *sub-group* different than sg and creates a different QI-group. In this case $Pr'_{div_{min}} = 0$.

- Pr''_{div}: the case were sg is l-diverse in D but not in D'. $Pr''_{div} = 0$ because sg in D' will not merge and don't create a QI-group or merge with another *sub-group* and create a different QI-group.
- $Pr^{(3)}_{div}$: the case were sg is not l-diverse in D but it is in D'. $Pr'''_{div} = 0$ for the same logic mentioned for Pr''_{div}.
- $Pr^{(4)}_{div}$: the case were sg is $l - diverse$ in D and D'. The minimum probability of this case is the probability of choosing l distinct sensitives values in sg in D and in D':

$$Pr^{(4)}_{div_{min}} = \frac{\mid A^s_D \mid!}{l! \times \mid A^s_D \mid^l} \times \frac{\mid A^s_{D'} \mid!}{l! \times \mid A^s_{D'} \mid^l}$$

where $\mid A^s_D \mid$ ($\mid A^s_{D'} \mid$) is the number of sensitive attributes in sg in D (D' respectively).

Therefore, the minimal value of Pr_{div} is:

$$Pr_{div_{min}} = 0 + 0 + 0 + \frac{\mid A^s_D \mid!}{l! \times \mid A^s_D \mid^l} \times \frac{\mid A^s_{D'} \mid!}{l! \times \mid A^s_{D'} \mid^l}$$

Finally, we can compute the minimal value of $Pr[ESG(D') \rightarrow S]$ as:

$$Pr[ESG(D') \rightarrow S]_{min} = Pr_{not-new} \times Pr_{mnt} \times Pr_{div_{min}}$$

Then:

$$\frac{Pr[ESG(D) \rightarrow S]}{Pr[ESG(D') \rightarrow S]} \leq \frac{l!^2 \times \mid A^s_D \mid^l \times \mid A^s_{D'} \mid^l}{\mid A^s_D \mid! \times \mid A^s_{D'} \mid! \times \sum_{i=1}^{|B|} pr_i}$$

Therefore:

$$\epsilon = ln \left(\frac{l!^2 \times \mid A^s_D \mid^l \times \mid A^s_{D'} \mid^l}{\mid A^s_D \mid! \times \mid A^s_{D'} \mid! \times \sum_{i=1}^{|B|} pr_i} \right)$$

4.7.2 Determining the policy

Blowfish Privacy is based on a privacy policy $P = (\mathcal{T}, G, \mathcal{I}_Q)$, where $\mathcal{T}$ denotes the domain of all possible datasets created from a universe of tuples, $G = (V, E)$ is a discriminative secret graph with $V \subseteq \mathcal{T}$, an edge in graph G connects two nodes that represent two datasets from $\mathcal{T}$ that differ by one and just one tuple and $\mathcal{I}_Q$ denotes the set of databases that are possible under the constraints Q that are known about the database.

4.7.2.1 Determining the constraints

It's not uncommon to have publicly known constraints about our original dataset. Therefore, logically, the anonymized data should also respect the constraints. But Differential Privacy doesn't take into consideration any constraint. This drawback is one of the reasons that

motivated He et al. [71] to propose the Blowfish Privacy as a generalization of Differential Privacy.

In our case, we can assume that the public knows that our released dataset should respect the ℓ-diversity.

Therefore, for each tuple $< u, v_s, conf(u, v_s) >$, the confidence score after any release is as follow: $conf'(u, v_s) \leq max(\frac{1}{l}, conf(u, v_s))$ where $conf'(u, v_s)$ is the updated $conf(u, v_s)$ after the release.

In other words, the release does not help the adversary to increase any of his confidence scores $conf(u, v_s)$ in any tuple to exceed an upper bound $max(\frac{1}{l}, conf(u, v_s))$.

For more explanation, what is meant by the upper bound of $conf'(u, v_s)$ is: if $conf(u, v_s)$ is already greater than $\frac{1}{l}$ then the release will not increase the confidence score, then $conf'(u, v_s)$ will be equal to $conf(u, v_s)$. If $conf'(u, v_s)$ is less then $\frac{1}{l}$, then the release will not allow to the adversary to get a $conf'(u, v_s)$ that exceeds $\frac{1}{l}$.

Thus, this upper bound can be considered as the constraint of the data when applying our mechanism.

Therefore, by proving that our mechanism respects the inequation of Blowfish Privacy and by defining the constraint, we have shown that Elastic Safe Grouping is $ln\left(\frac{l!^2 \times |A^s_D|^l \times |A^s_{D'}|^l \times |U|_{(Universe)}}{|A^s_D|! \times |A^s_{D'}|! \times |U|_{(D)} \times \sum_{i=1}^{|B|} pr_i}\right)$-Blowfish Private.

4.8 Experiments

We now present a set of experiments to evaluate the efficiency of our approach, both in terms of computation as well as loss of data utility. We implemented our algorithms in Java based on the Anonymization Toolbox [76], and conducted experiments with an Intel XEON 2.4GHz PC with 2GB RAM.

4.8.1 Dataset evaluation

# of distinct tuples	109760
# of distinct individuals	2374
# of distinct sensitive values	1306

Table 11: Dataset properties

We performed our experiments on a sample test dataset provided here [1]. The generated dataset simulates rental transactions, containing the attributes *UserID*, *VinNumber* and *Location*. The attribute *UserID* is the identifying attribute whereas *Location* is sensitive. The dataset is of size 109760 tuples with 2374 distinct individuals. Table 11 shows the properties of the dataset.

[1] https://github.com/ElieChicha/ESC/blob/main/sourcedata.xlsx

4.8.2 Utility metrics and measures

We use the following metrics and measures to evaluate the efficiency of our technique in preserving the data's utility.

4.8.2.1 Relative confidence error

The relative confidence error (RCE) measures the relative difference in the confidence of the correlation between an identifying value v_{id} and a sensitive value v_s between the original and anonymized tables.

This captures the impact of privacy protection on the ability to learn interesting and potentially useful correlations. RCE is defined in the following equation:

$$RCE(v_{id}, v_s) = \frac{|conf(v_{id}, v_s|T) - conf(v_{id}, v_s|T^*)|}{conf(v_{id}, v_s|T)}$$

where,

- $conf(v_{id}, v_s|T)$ is the confidence of v_{id} and v_s in table T,
- $conf(v_{id}, v_s|T^*)$ is the observed confidence that shows how often the association between v_{id} and v_s is true across the anonymized table T^*.

We assume that a *frequent* association between v_{id} and v_s in table T can still be identified in anonymized T^* (i.e., v_{id} is more likely to be associated with v_s in the QI-groups to which it belongs). The observed confidence is formally defined as follows:

$$conf(v_{id}, v_s|T^*) = \frac{\sum_{j=1}^{m} f_j(v_{id}, v_s)}{\sum_{j=1}^{m} g_j(v_{id})}$$

where

$$f_j(v_{id}, v_s) = \begin{cases} 1 & \text{if } v_{id} \text{ and } v_s \text{ are associated in } QI_j \\ 0 & otherwise \end{cases}$$

and,

$$g_j(v_{id}) = \begin{cases} 1 & \text{if } v_{id} \text{ exists in } QI_j \\ 0 & \text{otherwise} \end{cases}$$

4.8.2.2 Mean squared group size

The Mean Squared Group Size ($MSGS$) investigates how likely an identifying value remains associated with its corresponding sensitive value in post-anonymization [155]. The $MSGS$ is formally computed as follows:

$$MSGS(T^*) = \frac{\sum_{j=1}^{m} (|QI_j| - 1)^2}{|T| - 1}$$

Intuitively, the smaller the size of the QI-groups, the smaller the value returned by the $MSGS$, which indicates that an identifying value is more likely to be associated with its corresponding sensitive value in the same QI-group.

4.8.2.3 Kullback-Leibler divergence

In information theory, Kullback-Leibler (KL) divergence calculates the difference between two probability distributions. In this paper, we use it to compare the distributions of identifying attribute A^{id} values before and after anonymization; before and after anonymizing/suppressing some of the identifying values and tuples to preserve the privacy constraint. This measure originates from the observation that, whenever the distribution of identifying values in the anonymized table diverges from the original distribution, the utility of the data in the anonymized table will decrease. We denote by $P_{A^{id},T}$ and P_{A^{id},T^*}, the distributions of identifying attribute A^{id} values in T and T^* respectively. KL divergence on T and T^* is computed as follows:

$$D_{KL}(P_{A^{id},T}||P_{A^{id},T^*}) = \sum_{i=1}^{|\mathcal{D}(A^{id})|} Pr(v_{id_i}|T) \times \log \frac{Pr(v_{id_i}|T)}{Pr(v_{id_i}|T^*)}$$

where $|\mathcal{D}(A^{id})|$ is the number of distinct identifying values in A^{id} , $Pr(v_{id}|T)$ and $Pr(v_{id}|T^*)$ are the probabilities of the observed value v_{id} in table T and T^* respectively.

We use a non-linear transformation of KL divergence denoted by $D_N = 1 - e^{-D_{KL}}$, to bound the score of D_{KL} between 0 and 1 and compare it in several anonymization techniques. The smaller D_N value, the closer the distribution P_{A^{id},T^*} is to $P_{A^{id},T}$.

4.8.2.4 Comparing with the 1:M mechanism

This subsection compares our mechanism with the 1:M mechanism by evaluating each anonymization's Information Loss.

In [162], Xiao et al. compare their Anatomy algorithm to a Generalization algorithm. They use a set of 10 000 count queries applied on a generalized dataset and an anatomized dataset.

The generalization algorithm generalizes the QI attributes into groups. The sensitive attributes are not manipulated, but it's impossible to relate the sensitive attribute with its real corresponding attribute by more than $\frac{1}{k}$.

In the conditions of each query, they indicate at least one QI attribute beside the sensitive attribute. Then they compute the probability p that a tuple in the QI-groups qualifies the range predicates of the query. Finally, the query result is $p \times s$, where s the number of sensitive queries having the same value as the one indicated in the query.

This technique is not feasible for the 1:M algorithm because the sensitive attributes are also generalized in this algorithm. For this reason, we can use two probabilities and multiply their results by each other to answer the query. The first probability is that a tuple in a QI-group qualifies the query's range predicates. The second is that a sensitive attribute in the same QI-group qualifies the query's predicates for the sensitive groups. To compute these probabilities, we can apply the NCP used in the experiments in [65].

Let's the query be: Select count(*) from *Table* where $col_{QI1} = q_1$ and $col_{QI2} = q_2$ ... and $col_{QIm} = q_m$ and $col_{sens} = q_{sens}$. In this query we are counting all the tuples containing $q_1, q_2, ..., q_m$ and q_{sens}. We can apply the query twice and change the value of *Table* from Rental_Table_Original to Rental_Table_1M. Normaly, computing the difference between the two results gotten from the

query, then dividing the difference by the result of the query applied to Rental_Table_Original will give us the Information Loss.

However, this method is not achievable in the 1:M algorithm because all the table values are generalized. For this reason, we will compute the Information Loss in another way; we will rely on the Equation 1 to get the same result expected by using the method of the difference of two queries' results:

$$IL_{1M} =\mid Q_{sens} \mid \times NCP(q_{sens}) \times \sum_{i=1}^{m} QID - NCP(q_i) \tag{1}$$

where Q_{sens} is the generalization of q_{sens} and

$$NCP(q_{sens}) = \begin{cases} 0 & \mid q_{sens} \mid = 1 \\ \mid q_{sens} \mid / \mid A \mid & otherwise \end{cases}$$

and

$$QID - NCP(q_i) = \frac{\sum_{j=1}^{d} NCP(q_i)}{d}$$

where d is the number of all possible values for col_i.

After getting the information loss of the 1:M algorithm, we should get the loss of our algorithm to compare the results. The method that we apply to compute the loss is based on the following equation:

$$IL_{ESG} = \frac{\mid \sum_{i=0}^{g} (n_i^S(q_{sens}) \times \frac{n_i^{QI}(q_1,...,q_m)}{n_i^{QI}}) - Org\_Res \mid}{Org\_Res} \tag{2}$$

where g is the numbers of groups in the Table, $n_i^S(q_{sens})$ is the number of sensitive attributes in the i^{th} group of the Sensitive table where the value of the attributes is q_{sens}, $n_i^{QI}(q_1, ..., q_m)$ is the number of the tuples in the i^{th} group of the Quasi-Identifier table where the value of the attributes are $(q_1, ..., q_m)$, n_i^{QI} is the number of tuples in the i^{th} group in the Quasi-Identifier table and *Org_Res* is the result of the query when applied on the original table.

Finally, we compute the mean of all IL_{1M} computed for a set of queries and the mean of IL_{ESG} for the same set, and then we compare the two means to find out which method has a higher information loss.

We compare the two algorithms by applying IL_{ESG} and IL_{1M} on approximately 1000 combinations of attributes. Figure 49 shows that 1:M generalization has a rate of information loss more than ESG by 1.5 to 2 times generally.

4.8.3 Evaluation results

We elaborated a set of experiments to evaluate the efficiency of our elastic safe grouping and compared the results with safe grouping and anatomy. These experiments can be summarized as follows:

- Evaluating the size and the growth of QI-groups in a dataset anonymized using elastic safe grouping, safe grouping, and anatomy.

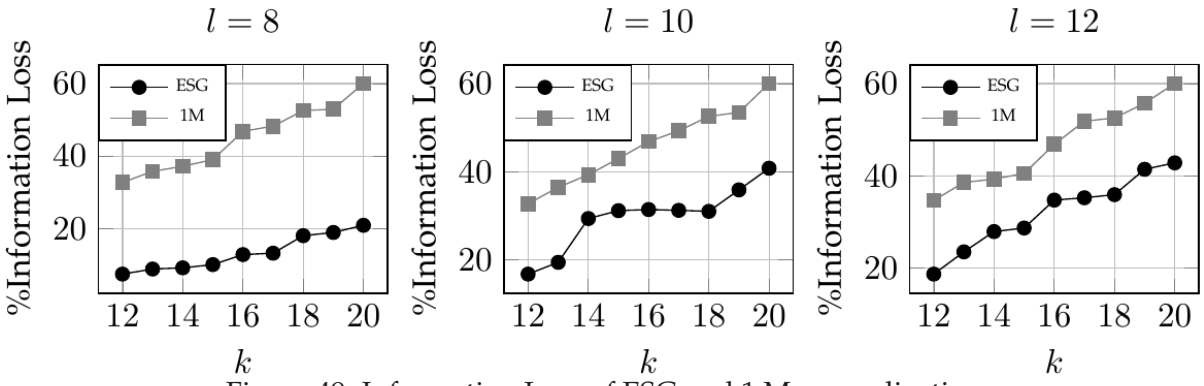


Figure 49: Information Loss of ESG and 1:M generalization

- Studying the impact of elastic safe grouping on preserving the correlations between identifying and sensitive values.
- Evaluating utility loss in terms of suppressions in a dataset anonymized using elastic safe grouping and safe grouping.
- Evaluating the performance of elastic safe grouping.

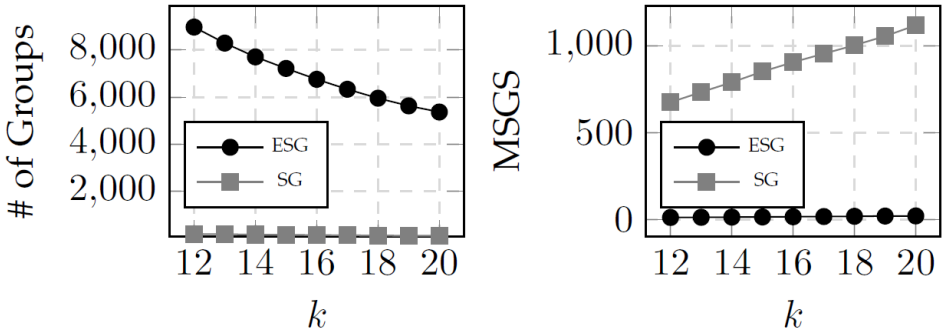


Figure 50: Evaluating the number of QI-groups and the MSGS in ESG and Safe Grouping

4.8.3.1 Evaluating the growth in number and size of QI-groups

In this section, we examine the growth in number and size of QI-groups. We vary k from 12 to 20 and leave l equal to 10. At each execution, we retrieve the number of QI-groups and the MSGS.

Let m be the number of the QI-groups produced in post-anonymization. m is computed as follows:

- using safe grouping, $m = \frac{|U|}{k}$ where, $|U|$ is the number of distinct individuals in T.
- using ESG, $\frac{|U|}{\delta} \leq m \leq \frac{n}{\delta}$ where, n is the number of tuples in T.
- using anatomy, $m \approx \frac{n}{l}$ where, l is the l-diversity privacy constant.

We consider that the smaller the size of the QI-groups, the higher the association of identifying and QI values is to their sensitive values. The results as shown in Figure 50 shows that ESG presents a considerable improvement over safe grouping for the number of QI-groups.

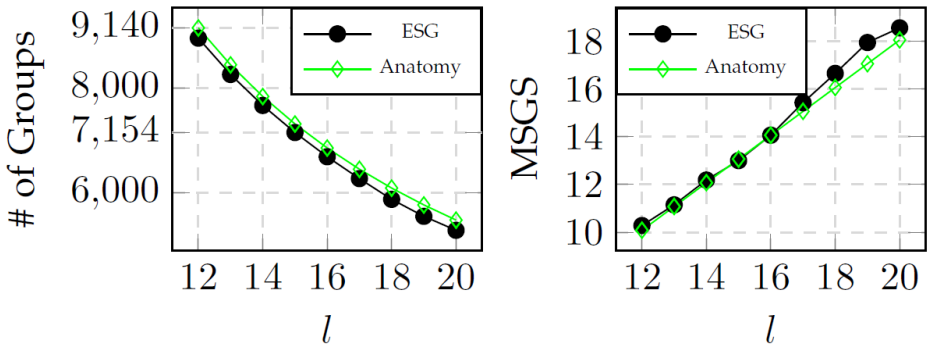


Figure 51: Evaluating the QI-groups numbers in Anatomy, Safe Grouping, and ESG

Alternatively, we compare the results of ESG and anatomy, where at each execution, we vary l from 12 to 20, and leave k constant ($k = 10$) since ESG is sensitive to the change of both l and k because of $\delta = max(k, l)$(see Step 3 in the ESG algorithm). Figure 51 shows that ESG provides almost the same utility as anatomy; keeping the associations between sensitive and identifying values suitable for discovery but exposing only safe correlations.

4.8.3.2 Evaluating the impact of ESG on the correlations between identifying and sensitive values

In this experiment, we use the RCE measure defined in Section 4.8.2.1 to evaluate how correlations are preserved after the anonymization of the table using both, ESG and safe grouping. Specifically, we learn significant correlations from the original table T (i.e., correlations having a confidence greater than or equal to a $maxConf$ threshold and a support greater than or equal to a $minSup$ threshold) and compute after that, at each run, their observed confidence from the anonymized table T^*. Finally, we compare the resulting average RCE measure using both, ESG and safe grouping algorithms.

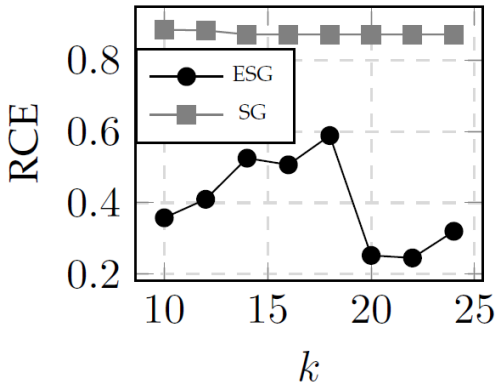


Figure 52: Comparing RCE using ESG and safe grouping

Here, we use $maxConf = 0.5$ and $minSup = 0.0001$ to identify 33 significant correlated identifying and sensitive values from table T. At each run, we vary k from 10 to 24 while $l = 10$. The results from Figure 52 show that the error of safe grouping remains almost invariant, since each of the individuals, has their identifying values stacked in the same QI-group. In ESG, however, the error is lower and tends to change with k, since some of the groups are merged to

create l-diverse QI-groups. Summarizing, the results show that significant correlations learned from the original table are better preserved using ESG.

4.8.3.3 Evaluating utility loss in T^*

In this experimental setup, we evaluate the utility loss when using safe grouping and ESG. Both algorithms guarantee privacy by either suppressing tuples and/or anonymizing identifying values.

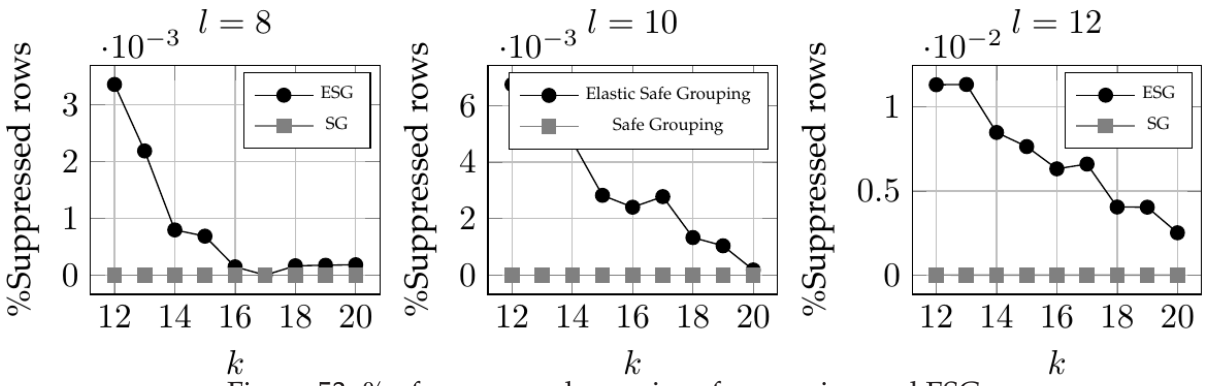


Figure 53: % of suppressed rows in safe grouping and ESG

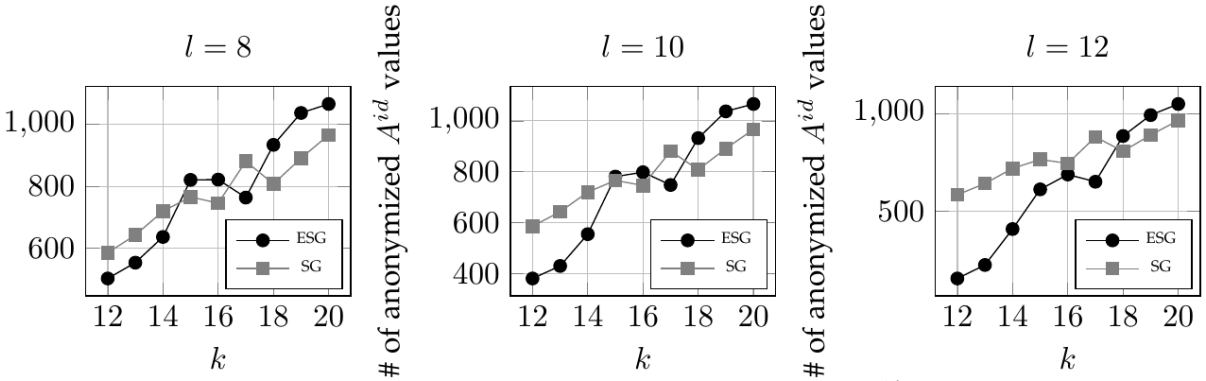


Figure 54: Evaluating the number of anonymized A^{id} values

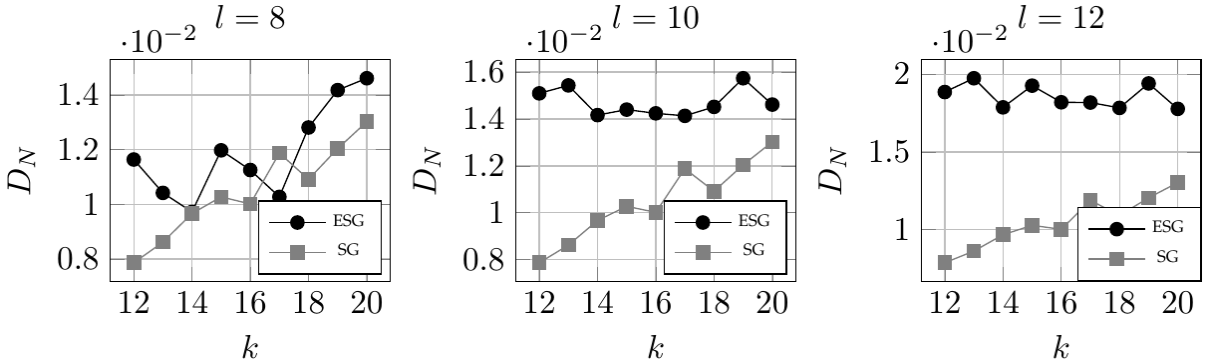


Figure 55: Comparing KL-divergence between safe grouping and ESG for $l = 8, 9$, and 10

We perform three separate tests. In the first test, we evaluate the percentage of rows suppressed to ensure l-diversity in both safe grouping and ESG. In the second, we evaluate the number of identifying values anonymized (i.e., suppressed/encrypted) to ensure the uniform

distribution of identifying values in each of the QI-groups (Requirement 2 in our privacy constraint in Section 4.6.1).

Finally, we compare the divergence in the distributions of A^{id} values in a dataset anonymized using ESG and safe grouping. For that matter, we use the D_N measure to determine the difference between the distribution of identifying values in the original table versus the distribution of identifying values in the anonymized table; anonymized using ESG, and safe grouping.

Each of the tests is repeated three times per anonymization technique where, in each run, we vary k from 12 to 20 and leave l constant for $l = 8, 10$, and 12. At each execution, with given values for k and l, we compute the percentage of suppressed rows, the number of anonymized identifying values, and D_N.

Test 1: Percentage of suppressed rows

The results from Figure 53 show that for small values of k, safe grouping performs better than ESG. The former suppresses few to no tuples at all compared to the outcome of ESG as the latter creates l-diverse QI-groups in each of the buckets. All the remaining QI-groups, the ones that are not l-diverse, are entirely suppressed (Step 17 in the Divide function). In safe grouping, however, only a few tuples are suppressed from the QI-group in succession and as long as l-diversity is not met [9, 10]. This explains why for small values of k, safe grouping retains more tuples than ESG, but when k increases, QI-groups grow in size thus, more likely l-diverse. As noticed, for $k = 20$ the percentage of tuples suppressed in both safe grouping and ESG is almost the same. Another aspect to show is that when l increases the percentage of suppressed tuples increases too since the number of QI-groups that respect l-diversity tends to decrease.

Test 2: Number of anonymized identifying values

From Figure 54, we can see that the number of anonymized identifying values increases with k. This could happen whenever we stack additional identifying values in a QI-group where both, ESG and safe grouping tend to anonymize more values to ensure the uniform distribution in the QI-group. For instance, if a bucket B contains identifying values of 4 distinct individuals, u_0, u_1, u_2, and u_4 having $|T_{u_0}| = 1$, $|T_{u_1}| = 2$, $|T_{u_2}| = 4$, and $|T_{u_3}| = 6$. For $k = 2$, two QI-groups are created, $QI_1 = \{T_{u_0}, T_{u_1}\}$ and $QI_2 = \{T_{u_2}, T_{u_3}\}$. The total number of anonymized identifying values is equal to 3.

For $k = 3$, only one QI-group is created in which we add tuples of 4 distinct individuals $QI_1 = \{T_{u_0}, T_{u_1}, T_{u_2}, T_{u_3}\}$. The number of identifying values to anonymize is now equal to 9. Another finding is that there is a negative correlation between the number of anonymized identifying values and l, since the process of ensuring the uniform distribution of identifying values is applicable only for l-diverse QI-groups. That is, when l is large, both anonymization algorithms, safe grouping, and ESG, produce fewer l-diverse QI-groups, and thus, the number of values to be anonymized tends to decrease. Note that a suitable trade-off between values of k and l is likely to achieve an acceptable number of anonymized identifying values and suppressed tuples .

Test 3: Comparing the divergence in the distributions of A^{id} values in pre- and post-

anonymization

Here, we summarize our findings regarding the loss in utility due to the suppression/anonymization of tuples and identifying values. We choose to compute D_N of the distributions of identifying values before and after the anonymization of the dataset using both ESG and safe grouping. The results in Figure 55 show that safe grouping provides less divergence at the beginning but converges however to reach values produced by ESG when k increases. Overall, the anonymization cost is relatively small where we can see that the value of D_N is at most equal to 0.02 for $l = 12$.

CHAPTER 5

Differential Privacy for Image Classification

5.1 Introduction

In this part, we aim to design and develop an anonymous full-duplex image classification framework under Differential Privacy. We work under the assumption that both the cloud and the querier are semi-trusted entities; thus, their data should remain safe and confidential. That is, neither the querier nor the cloud should be able to link a particular individual from the other party to an image while maintaining, to a certain extent, suitable classification accuracy. We use Principal Component Analysis (PCA) to transform images into anonymized vectors (differentially private synopsis of PCA vectors) and we ensure that the individuals in these vectors remain unidentifiable.

5.2 Related Work

The K-Nearest Neighbors (K-NN) [161] is one of the most popular and influential data mining algorithms in the literature. However, many adaptive attacks [94] can form a real threat to data privacy in K-NN based systems. Li et al. propose in [94] a privacy-preserving system based on Kernel density estimation using Gaussian Kernel instead of K-NN.

Their system, as most of the other related works, uses computation over encrypted data. They describe four roles:

- The data owners submit encrypted data to the system.
- The queriers submit encrypted queries to receive classification results.
- The host role, possessed by the cloud, stores the incoming encrypted data and hosts the classification.
- Finally, the Cryptographic Service Provider (CSP) owns both encryption and decryption keys.

They demonstrate that Distance-Learning attacks under the K-NN system can breach data privacy if an untrusted data owner is, at the same time, a querier and propose to use Kernel density estimation instead of K-NN. In this paper, due to the noisy vectors and the sampling, using K-NN is safe.

The work described in [149] turns the K-means clustering algorithm to a differentially private algorithm, where noise is added to the centroids in a way that respects the requirements of Differential Privacy. Differentially private K-Means is divided into two approaches: interactive and non-interactive.

Interactive approach [149] is based on a query that can be used just once, can serve only one querier and only for one task. Any mechanism that is based on this approach returns a noisy result to the user. Each query has a budget $\alpha = \sum_i \alpha_i$ given by the database owner. Each execution i makes the budget loses α_i of its value. The query cannot be executed anymore when the α budget is less than α_i. Hence, this approach has many restrictions on privacy preservation,

Authors	Function	Trusted Parties	Time Consuming	Queries	Classification
Li et al.	Encryption	CSP + Trusted data owner	High	Encrypted	Kernel Density Estimation
Nassar et al.	Partially Homomorphic Encryption	Cloud + Trusted data owner	High	Non-private	K-NN
Our work	Differential Privacy	Anonymization Service + Trusted data owner	Low	Differentially Private	K-NN + Kernel Density Estimation + SVM

Table 12: Comparing related works with our work.

especially if the budget is small where the number of queries could be insufficient. Besides, the data owner should validate the query. The non-interactive approach algorithms [149] return a noisy synopsis data set. The querier can send queries to this synopsis to get noisy statistical data. This approach has no limits nor restrictions to the number and the sender of the queries.

Several other works [122, 152, 64] rely on encryption for privacy. Taheri et al. in [152] propose a method for face authentication in the encrypted domain. In [64], they propose a general framework for multi-biometric template protection based on homomorphic probabilistic encryption. The work in [122] is similar to our proposed approach but relies on partially homomorphic encryption called Paillier's encryption. This type of encryption is a public key scheme; this means that the encryption can be done using a public key, while decryption can only be done by a trusted party that possesses the private key. The technique, however, assumes that the cloud is a trusted party, and thus the privacy of the dataset is threatened.

As shown in Table 12, the two other works rely on encryption, and all the works need a trusted data owner. The trusted data owner is always required, or the system model cannot be private. The other trusted party is the cryptographic service for the first work, the cloud for the second (which can cause a severe threat to the dataset), and the anonymization service for our work. These encryption and decryption tend to be time-consuming. Finally, we consider that using noise addition techniques like Differential Privacy can provide better performance and keep the data suitable for classification.

The benefits and disadvantages of other techniques and mechanisms should be studied to find out their capabilities in this domain, like pixelization, blurring, and PCA-like mechanisms. Although we think that blurring or pixelization cannot provide the same level of privacy and utility as Differential Privacy, but this issue needs more study in future works.

5.3 Principal Components Analysis

PCA [129] is a singular value decomposition to reduce data dimensionality. Given data points in an m-dimensional space, PCA projects them into lower-dimensional space while preserving as much information as possible. Consider a matrix $S(n \times m)$, in this work, each matrix column represents an image of m features. These features are, for instance, the RGB values (or the grayscale values) of the image pixels. The features are subtracted from their respective

empirical means. Assuming a normal (Gaussian) probability distribution of errors, orthogonal transformations naturally arise. The PCA is an orthogonal transformation (actually a coordinate rotation) that aligns the transformed axes with the directions of maximum variance:

$$S = U \sum V^T \tag{1}$$

where U and V are orthogonal matrices, and $\sum$ has the set of singular values. $U(m \times m)$ has for columns the eigenvectors of the covariance matrix C of S. The first column of U is the eigenvector having the largest absolute eigenvalue (the first principal component). The most significant variance is in the direction of the first principal component. The most significant variance on the subspace orthogonal to this vector is in the direction of the second principal components, and so on. The matrix $B = U^T S$ is therefore useful for dimensionality reduction of the original data. In effect, $B_r = U_r^T S$, where U_r is the smaller matrix having only the first $r < m$ columns of U, is a reduced representation of the data. B_r is an $r \times n$ matrix. Data reconstruction is governed by $S_r = U_r B_r$ (an $n \times m$ matrix), which is a lossy version of the original data S [16].

In our mechanism, we intend to add enough noise to the PCA vectors of images in a way that the individuals in the images remain unidentifiable, but at the same time, the vectors can be used to train the classifier.

5.4 Differentially Private Image Classification Framework

We assume that the first party (the data owner) holds a dataset of face images. The images belong to different classes, and each image is labeled with its class number. The second party, which is the end-user, has one face image of an unknown class and wants to predict this class, or a multi-face image and wants to know how many faces have a specific reaction. In non-private settings, this would be easy: We use a database with labeled images to train the classifier, and we predict an unknown image based on the classifier's result.

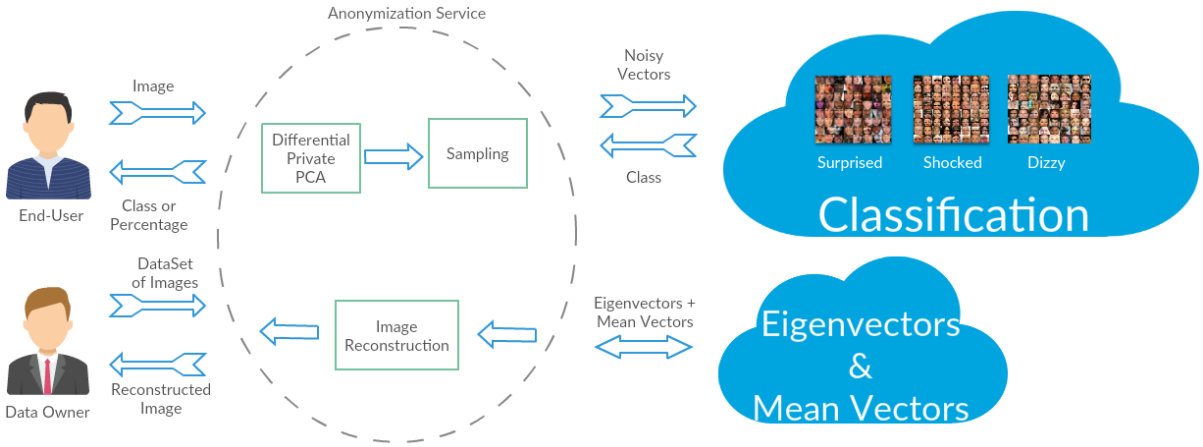


Figure 56: *General Architecture of a Differentially Private image classification system*

In private settings, we want to imagine a man in the middle asking for information about the image (the query) and the image database (or the classification model). This man in the middle must not be able to recognize the identity of any subject no matter how many questions he asks.

Both parties must add noise to make the whole process Differentially Private. We want as a result that any single image in the database be masked by reducing every image into a noisy vector. The query image is masked as well. However, to ensure an accurate classification, adding noise to a database of images and requests must be done using a global privacy parameter.

In our framework (see Fig.56), four types of actors are listed:

- *Trusted Data Owner:* sends the essential dataset to anonymization service, and later, can send more images with trusted classification.
- *Untrusted Data Owner:* sends images to the anonymization service, with untrusted classification. Trusted and untrusted data owners may request a noisy image from the synopsis dataset to be sure that the face is unrecognizable. The principal component vectors, the eigenvectors, and the mean vectors are required to reconstruct images, so the service keeps the eigenvectors and means vectors on a different cloud.
- *Querier:* sends one face or multi-face images to the anonymization service with the aim of getting their classifications.
- *Data Host:* receives three types of vectors:

  1) Trusted data owner vectors (sample noisy images) are used to train the classification algorithm.

  2) Untrusted data owner vectors are classified, and the output is returned to the anonymization service to check the truthfulness of the data owner classification.

  3) Querier vectors are classified, and the output is returned to the querier.

5.4.1 Anonymization service

The service transforms every image in the primary dataset into a vector using PCA and adds Differentially Private noise to the vectors. It performs safe sampling and returns a synopsis dataset, and checks the truthfulness of the untrusted data owner classification. The process breaks down as follows:

The first step is to concatenate a set of images into one matrix S as PCA reduces a set of images into a set of vectors. Then, we compute the covariance matrix $C = \frac{1}{m}S^T S$; where m is the number of columns of S.

PCA is turned into a Differentially Private mechanism by adding differential private noise to C. The noise matrix L is generated by sampling its elements from a probability distribution. Then, the noisy covariance N is calculated: $N = L + C$.

To get the vectors of the images, we should first calculate characteristics vectors and values called respectively eigenvectors u and eigenvalues λ using this equation: $(N - \check{}\lambda I)u = 0$. Eigenvalues can be found by calculating $det(N - \check{}\lambda I) = 0$, and then the eigenvectors are calculated. All the eigenvectors will be concatenated in one matrix U, and the PCA matrix P is computed by multiplying S by U. Each column of P is a PCA vector. If a trusted data owner

sends the set of images, then the vectors will be sampled to create a synopsis dataset sent to the cloud.

5.4.2 Privacy-preserving PCA

To make PCA a privacy-preserving mechanism using differential privacy, we should turn the covariance matrix C into a noisy matrix. Therefore, we generate a noise matrix from a probability distribution and add it to the covariance matrix. Practically, using C in the PCA process returns a memory error due to the enormous dimensions of C. For this reason, C is substituted by a matrix $A = \frac{1}{m}SS^T$ where data matrix $S \in \mathbb{R}^{n \times m}$. Then each row of the matrix P is a PCA vector instead of each column.

5.4.2.1 Generating matrix N with Laplacian noise

We choose Laplace distribution [75] to achieve differential privacy. The probability density function of Laplace distribution $Lap(\mu, b)$ is:

$$P(x \mid \mu, b) = \frac{1}{2b} e^{\frac{-|x-\mu|}{b}} \tag{1}$$

where $\mu = 0$, $b = \frac{2d}{n\alpha}$ and α is a privacy parameter.

For $S \in \mathbb{R}^{n \times m}$, we sample $\frac{d^2+d}{2}$ values from this distribution to fill the upper triangular part of the noise matrix L then the values of the lower triangle part are copied from opposite position. Finally, we add the noise matrix L to A.

5.4.2.2 Generating matrix N with Laplacian noise

For a long time, SULQ method [24] was the only Differentially Private approximation to PCA. This method based on Gaussian distribution guarantees weaker privacy than the differential privacy known as (α, δ)-differential privacy. Chaudhuri et al. in [31] proved that SULQ is not a good candidate for effective dimensionality reduction. They proposed a simple modification and called their method MOD-SULQ. This method, as in SULQ, is based on Gaussian distribution. First, we compute the parameter of the distribution:

$$\beta = \frac{d+1}{n\alpha}\sqrt{2\log\left(\frac{d^2+d}{\delta 2\sqrt{2\pi}}\right)} + \frac{1}{\sqrt{\alpha}n} \tag{2}$$

where α is a privacy parameter and δ is a relaxation parameter.

We generate after that a symmetric noise matrix based on the probability density function of Gaussian distribution $N(\mu, \beta^2)$:

$$P(x \mid \mu, \beta^2) = \frac{1}{\sqrt{2\beta^2\pi}} e^{-\frac{(x-\mu)^2}{2\beta^2}} \tag{3}$$

where $\mu = 0$. Finally the noise matrix is added to A.

5.4.3 sampling to create a synopsis dataset

To perform the sampling, we reconstruct some images from noisy vectors whose classifications were not affected by the added noise.

We compute two distance scores of the noisy image and compare them to two user-defined thresholds. First, we compare each image to its noisy version. Second, we compare the noisy image to the mean of the images with the same classification label.

If distance scores are less than these thresholds, we can consider the image as unidentifiable, and at the same time, its classification was not affected by the noise. Therefore this noisy vector will be added to our synopsis dataset.

If the data owner is untrusted, then the vectors are classified on the cloud, and the classification result is compared to the label sent by the data owner for each image. If the label of the image and the classification result are not identical, then the vector is dropped. Otherwise, the vector will be a part of the sampling process.

The images sent by data owners are already labeled. Therefore, the sampled vectors are classified and form the training dataset. If an end-user sends the images, then after applying the private-preserving PCA process on the images, the noisy vectors are directly sent to the cloud to be classified.

5.5 Experiments

Our experiments were applied on a dataset of Japanese face images (JAFFE) [106], containing 213 images of 7 facial expressions.

5.5.1 Experimental Settings

We have applied three classification algorithms to test which one returns better results.

The first algorithm is the Support Vector Machine that has proved its superiority among many other classification methods [38]. SVM is based on transforming the data in input space into data in featured space in a way that renders the classes linearly separable. Then, a line or a plane is drawn between the classes, and the classification is based on this line or plane.

The two other classification algorithms K-NN and Kernel Density Estimation, are from the same family. We have chosen K-NN and Kernel Density Estimation to check if K-NN is more accurate than the Kernel algorithm; in this case, we can declare that we can apply K-NN in our model with no concerns about distance-learning attacks in DO-Q Threat Model case [94].

Kernel calculates the influence of every point in each class on a given query relative to the distance between this point and the query. The Gaussian Kernel formula to calculate the influence is:

$$K(|| \ q - x_i \ ||) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}||q-x_i||^2} \tag{1}$$

where q is the query tuple, x_i is the $i-th$ data tuple, σ is a parameter chosen to maximize accuracy [94] and $||\ .\ ||$ is the $L2$ norm.

In K-NN, which is one of the most popular classification algorithms, the K nearest neighbors have the same influence on the query; the others have 0 impacts. This shows that choosing the parameter K is very critical to get the best result from this algorithm [13].

The two most notable observations we have found in the empirical study are: The need for a minimal privacy parameter (vast amount of noise) to provide privacy for images. Hence, adding more and more noise is unable to hide facial features.

For the first observation, we have studied the values of the elements inside the matrices and the vectors:

For a set of images of dimensions $n \times m$, in every image the value x of the pixel $\in [0, 255]$, then data is centered so $x_c \in [-128, 128]$.

We compute $A = X_c.X_c^T$ so $x_a = x_1^2 + x_2^2 + ... + x_{(n \times m)}^2$; where X_c is the centered data. Therefore, $x_a \in [0, (128 \times n \times m)^2]$.

In our experiments $n = m = 256$, so $x_a \in [0, 7.10^{13}]$, we have found that to provide enough privacy to the reconstructed images, we should have $m > [0, 7.10^{13}]$, where n is an element of the noise matrix, to get this value for m, the privacy parameter $\alpha \leq 10^{-15}$.

For the second observation, let's consider the function used to rebuild the images:

$$i = vec.U + mean \tag{2}$$

where *vec* is the noisy vector of the specified image, U is the matrix containing the eigenvectors, and *mean* is the vector containing the means of the dataset.

Then after reshaping i as $n \times m$ matrix, the matrix is normalized:

$$i = \frac{i - min(i)}{max(i) - min(i)} \times 255 \tag{3}$$

We assume that reconstructing images using the eigenvectors, the mean, and then normalizing the matrix makes it impossible for the noise to hide the face or its features.

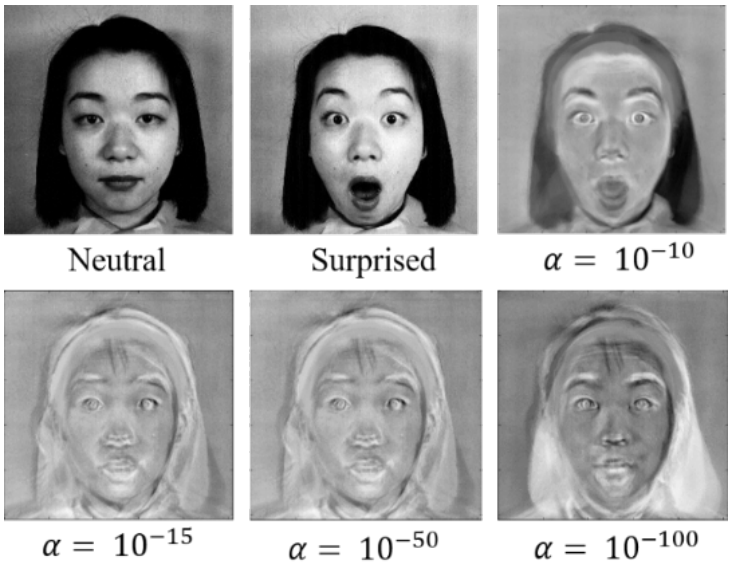


Figure 57: Image reconstruction based on Laplace noisy vectors.

From Figure 57, we can see that for a 256×256 image, the privacy parameter $\alpha < 1-^{-15}$ is not enough to provide privacy where, regardless of how small α is, the features are still identifiable.

5.5.2 Evaluating Privacy vs. Accuracy

First, we compare the three classification algorithms, Kernel Density Estimation, SVM, and K-NN. The latter surpasses the two other algorithms by far, as the accuracy of Kernel Density Estimation and SVM did not exceed the 50%. Thus, we have focused on K-NN and compared its performances based on privacy parameter α, probabilistic distributions (Laplace and Gaussian) and distance functions (Chebyeshev, Euclidean, and Manhattan) used in K-NN.

In Figure 58, we can notice how the accuracy decreases when α decreases. For $\alpha = e^{-10}$, the

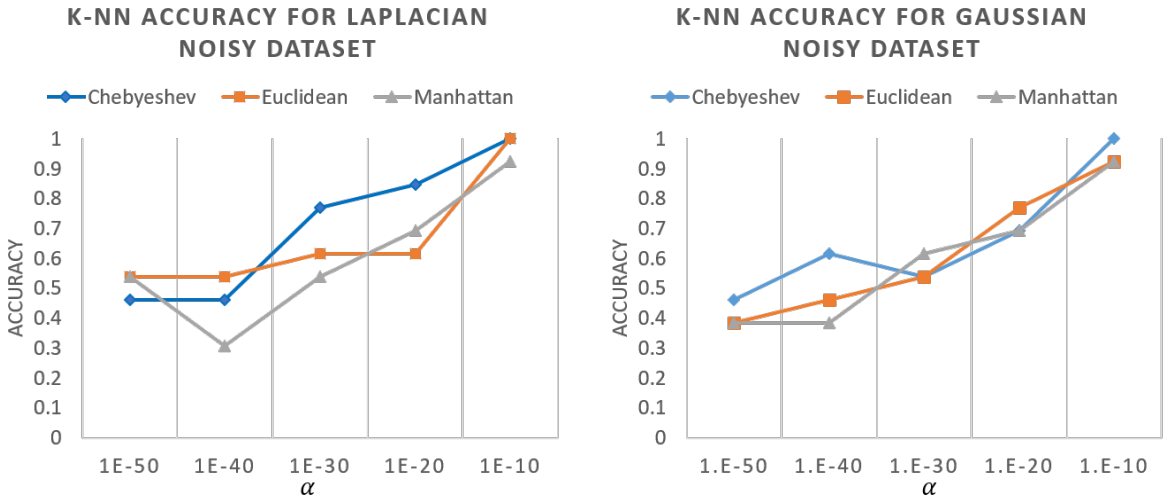


Figure 58: Accuracy Score for K-NN Classification.

accuracy ranges between 0.92 and 1, but in the previous section, we have seen how the noisy images with $\alpha < e^{-15}$ are identifiable.

Chebyshev Distance outperforms the two other distances for Laplacian noisy dataset. For α between e^{-15} and e^{-20}, the accuracy is 0.9 when using Chebyshev, which forms a good trade-off between privacy and utility.

For the Noisy Gaussian dataset, the three distance functions have close results. For α between e^{-50} and e^{-15}, the accuracy is between 75% and 85%. Hence, for a better trade-off between privacy and utility, the data owner should apply the Laplace mechanism with a privacy parameter in the range of $[e^{-50}, e^{-15}]$ using the Chebyshev distance in K-NN classification.

CHAPTER 6

Conclusion

6.1 Recap

In this thesis we present the crucial role played by the digital data in several fields especially in e-commerce and user-centric social networks businesses and how the heterogeneous data generated by users can help make more profit. We discuss the risk of privacy breaches while analysing this data and their possible damages on users and companies at the same time. We propose many Blowfish and Differential Privacy mechanisms as a solution to help the companies benefit from three type of data with the minimum risk possible.

In the first use-case, we propose three mechanisms to produce a good balance between utility and privacy on graph datasets. We detail the types of graphs (static and dynamic), the types of attacks (passive and active) and types of mechanism (offline and online). Then, we propose our three mechanisms to cover all these cases.

We have summarized Blowfish privacy, its formal model and the enhanced privacy-utility trade-off that it brings with respect to its predecessor, Differential Privacy. Then, we enrolled examples of its application to communication graph databases and their typical queries. We further studied the idea of privacy as a service with differentiation among different groups of individuals. We showed that this relaxation is formally feasible and proved its utility through the enrollment of several queries and computing their sensitivity. We work under the settings of binary differentiation (standard vs. VIP) and binary communication status (0 or 1). In addition, we propose a mechanism to provide anonymized results for more sophisticated queries such as local clustering coefficients, lengths of shortest paths, and some centralities.

Then, we proposed, implemented, and evaluated a BP mechanism to provide privacy for sequentially released graphs. We have proved that DP cannot ensure strong privacy for subgraphs in an intersection of several releases. Then we have proposed our solution based on the BP definition. In the experiments, we have proposed a method for the data owners to help them calibrating the trade-off of privacy and utility of their graphs. We have proved that our mechanism preserves the graphs' utility better than a DP mechanism and provides strong privacy for subgraphs.

For the third mechanism, we presented an active attack on dynamic graphs. These graphs are anonymized and published based on an online approach. We explain how edge-DP cannot deal with active attacks on dynamic graphs and how node-DP has inferior results on the utility side. Then we propose our mechanism called Node-Detention Differential Privacy (NNDP), where suspicious nodes are hidden in a hyper node called Detention node for one or more releases. Our mechanism is a new type of node-DP that differs from others in that it's the first to return an anonymized graph instead of the anonymous query result. The experiments prove that NNDP can provide a very high level of privacy and a very acceptable level of utility compared to two baseline node-DP mechanisms.

In our second use-case, we proposed a technique that safely exposes significant correlations between individuals and their sensitive values; correlations bounded by a privacy constant δ to preserve utility for associations discovery. We adopted a divide-and-conquer strategy to spread

the correlations safely into several QI-groups and put it into effect in an elastic safe grouping algorithm. We studied the efficiency of our algorithm by comparing the resulting utility to safe grouping algorithm and to anatomy. The results of our experiments showed considerable improvement for associations discovery compared to safe grouping, while compared to anatomy, results were close.

In the third use-case we focus on image datasets where we have proposed, implemented and evaluated a private image classification framework based on Differentially Private Principal Components Analysis. Using this framework, we ensure that the individuals in the image dataset are kept safe on a semi-trusted cloud service by adding Pifferentially Private noise to the images' PCA vectors. User requests are distorted as well to keep the participating individuals unidentifiable. We elaborated a set of experiments to evaluate the trade-off between the accuracy and the privacy of the dataset against several classification algorithms, namely K-NN, Kernel Density Estimation, and SVM. K-NN has shown to be very promising. We identified as well that a privacy parameter within the range of $[e^{-15}, e^{-20}]$ must be used to balance between the privacy and accuracy of K-NN using the Laplacian mechanism.

6.2 Future Work

The three mechanisms presented in this thesis for graph anonymization rely on a trusted central authority. The individuals presented in the graph might not trust the central authority and refuse to share their row data with it before being anonymized. For this reason, in future work, we will propose

- a local version of Blowfish Privacy definition for graphs, and apply it on graphs with multiple levels of privacy by proposing a decentralized mechanism to compute the noise added to the results of the queries

- a decentralized mechanism to protect particular subgraphs in dynamic graphs where the nodes composing these graphs decide the status of their subgraphs in the published version of each release of the dynamic graph

- a decentralized system under a local version of Differential Privacy where other nodes, especially the neighbors, decide if the status of a node is hidden or free. We will present a profound study about the challenges and advantages that don't appear in the centralized approach. In addition to this, we will focus on the detention node by presenting a deep study about the idea of having a set of detention nodes in each release instead of just one. How many detention nodes should we have in each graph? How do we distribute the suspicious nodes between these detention nodes? What is the form of an edge between two hidden nodes in two different Detention? What are the drawbacks and the advantages of having one Detention node versus many Detention nodes?

For the transactional data, in the near future, we intend to extend our technique to provide privacy against strong adversaries; adversaries that can combine multiple values in a multi-dimensional correlation, while preserving the utility of the dataset.

For the image processing, we intend to evaluate the efficiency of our approach in a real application scenario where collaborative attacks in which some of the data owners may be semi-trusted as well.

The digital data generated by online users and customers have other types than the ones focused on in this thesis. Two of these types are locations and texts. Therefore in the future, we will propose

- a Blowfish Private mechanism for locations that provides useful information after analysis and ensures that the users' exact locations could not be exposed. This mechanism also will consider the correlation problems represented by the recurrences of locations for the same user that could reveal private data about them.
- a Differential Private mechanism for data mining on unstructured text data that should provide a good balance between the utility of the collected text content and the privacy of the users who have written the posts or the replies.

My Publications

[1] Elie Chicha, Bechara Al Bouna, Mohamed Nassar, and Richard Chbeir. Cloud-based differentially private image classification. *Wireless Networks*, pages 1–8, 2018.

[2] Elie Chicha, Bechara Al Bouna, Kay Wünsche, and Richard Chbeir. Exposing safe correlations in transactional datasets. *Service Oriented Computing and Applications*, pages 1–19, 2021.

[3] Elie Chicha, Bechara Al Bouna, Mohamed Nassar, Richard Chbeir, Ramzi A Haraty, Mourad Oussalah, Djamal Benslimane, and Mansour Naser Alraja. A user-centric mechanism for sequentially releasing graph datasets under blowfish privacy. *ACM Transactions on Internet Technology (TOIT)*, 21(1):1–25, 2021.

[4] Elie Chicha, Mohamed Nassar, Bechara Al Bouna, and Richard Chbeir. Node-detention differential privacy for active attacks on dynamic graph datasets. 2021.

[5] Mohamed Nassar., Elie Chicha., Bechara Al Bouna., and Richard Chbeir. Vip blowfish privacy in communication graphs. In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications - Volume 3: SECRYPT*, pages 459–467. INSTICC, SciTePress, 2020.

Bibliography

[6] Taras Agryzkov, Leandro Tortosa, and Jose F Vicent. A variant of the current flow betweenness centrality and its application in urban networks. *Applied Mathematics and Computation*, 347:600–615, 2019.

[7] 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.

[8] Emel Akan. Us turns tide on huawei, ending ccp's 5g master plan, Dec 2020.

[9] Bechara al Bouna, Chris Clifton, and Qutaibah M. Malluhi. Using safety constraint for transactional dataset anonymization. In *DBSec*, pages 164–178, 2013.

[10] Bechara al Bouna, Chris Clifton, and Qutaibah M. Malluhi. Anonymizing transactional datasets. *Journal of Computer Security*, 23(1):89–106, 2015.

[11] Bechara al Bouna, Chris Clifton, and Qutaibah M. Malluhi. Efficient sanitization of unsafe data correlations. In *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Brussels, Belgium, March 27th, 2015.*, pages 278–285, 2015.

[12] David J. Aldous. Exchangeability and related topics. In *École d'été de probabilités de Saint-Flour, XIII—1983*, volume 1117 of *Lecture Notes in Math.*, pages 1–198. Springer, Berlin, 1985.

[13] Oren Anava and Kfir Levy. k*-nearest neighbors: From global to local. In *Advances in Neural Information Processing Systems*, pages 4916–4924, 2016.

[14] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914, 2013.

[15] Adeel Anjum and Guillaume Raschia. Banga: An efficient and flexible generalization-based algorithm for privacy preserving data publication. *Computers*, 6(1):1, Jan 2017.

[16] Uri M Ascher and Chen Greif. *A first course on numerical methods*, volume 7. Siam, 2011.

[17] 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, 2007.

[18] Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for aol searcher no. 4417749. *New York Times*, 9 August, 2006.

[19] 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.

[20] Smriti Bhagat, Graham Cormode, Divesh Srivastava, and B Krishnamurthy. Prediction promotes privacy in dynamic social networks. In *Proceedings of the 3rd Wonference on Online social networks*, 2010.

[21] Joachim Biskup, Marcel PreuB, and Lena Wiese. On the inference-proofness of database fragmentation satisfying confidentiality constraints. In *Proceedings of the 14th Information Security Conference*, Xian, China, oct 26-29 2011.

[22] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 410–419. IEEE, 2012.

[23] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[24] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138. ACM, 2005.

[25] Béla Bollobás and Bollobás Béla. *Random graphs*. Number 73. Cambridge university press, 2001.

[26] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

[27] Piotr Bródka, Krzysztof Skibicki, Przemysław Kazienko, and Katarzyna Musiał. A degree centrality in multi-layered social network. In *2011 International Conference on Computational Aspects of Social Networks (CASoN)*, pages 237–242. IEEE, 2011.

[28] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicencc Torra. An algorithm for k-degree anonymity on large networks. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 671–675. IEEE, 2013.

[29] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicencc Torra. k-degree anonymity and edge selection: improving data utility in large networks. *Knowledge and Information Systems*, 50(2):447–474, 2017.

[30] Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/, 1996.

[31] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. Near-optimal differentially private principal components. In *Advances in Neural Information Processing Systems*, pages 989–997, 2012.

[32] Ajay Chawla. Pegasus spyware–'a privacy killer'. *Available at SSRN 3890657*, 2021.

[33] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *Theory of Cryptography Conference*, pages 363–385. Springer, 2005.

[34] Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

[35] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664. ACM, 2013.

[36] Xihui Chen, Ema Këpuska, Sjouke Mauw, and Yunior Ramírez-Cruz. Active re-identification attacks on periodically released dynamic social graphs. In *European Symposium on Research in Computer Security*, pages 185–205. Springer, 2020.

[37] Xihui Chen, Sjouke Mauw, and Yunior Ramírez-Cruz. Publishing community-preserving attributed social graphs with a differential privacy guarantee. *Proceedings on Privacy Enhancing Technologies*, 2020(4):131–152, 2020.

[38] Sungmoon Cheong, Sang Hoon Oh, and Soo-Young Lee. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing-Letters and Reviews*, 2(3):47–51, 2004.

[39] Sean Chester, Bruce M Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and Sistla Venkatesh. Why waldo befriended the dummy? k-anonymization of social networks with pseudo-nodes. *Social Network Analysis and Mining*, 3(3):381–399, 2013.

[40] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM Trans. Inf. Syst. Secur.*, 13:22:1–22:33, July 2010.

[41] Jessica R Cohen and Mark D'Esposito. The segregation and integration of distinct brain networks and their relationship to cognition. *Journal of Neuroscience*, 36(48):12083–12094, 2016.

[42] Chenyun Dai, Gabriel Ghinita, Elisa Bertino, Ji-Won Byun, and Ninghui Li. Tiamat: a tool for interactive analysis of microdata anonymization techniques. *Proceedings of the VLDB Endowment*, 2(2):1618–1621, 2009.

[43] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*, pages 1–15. San Diego, CA, 2009.

[44] Julie Hirschfeld Davis. Hacking of government computers exposed 21.5 million people, Jul 2015.

[45] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138. ACM, 2016.

[46] Josep Domingo-Ferrer and Jordi Soria-Comas. From t-closeness to differential privacy and vice versa in data anonymization. *Knowl.-Based Syst.*, 74:151–158, 2015.

[47] Yitao Duan, Jingtao Wang, Matthew Kam, and John Canny. Privacy preserving link analysis on dynamic weighted graph. *Computational & Mathematical Organization Theory*, 11(2):141–159, 2005.

[48] Raquel Pita Guerreiro Marcelino Duarte. *Case Study: Facebook in face of crisis*. PhD thesis, 2020.

[49] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[50] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052, pages 1–12, Venice, Italy, July 2006. Springer Verlag.

[51] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[52] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[53] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

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

[55] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[56] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

[57] NasserYazdani Fatemeh Amiri and Azadeh Shakery. Bottom-up sequential anonymization in the presence of adversary knowledge. *Information Sciences*, 405:316–335, june 2018.

[58] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[59] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[60] Benjamin CM Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)*, 42(4):1–53, 2010.

[61] Franz-Stefan Gady. New snowden documents reveal chinese behind f-35 hack, Jan 2015.

[62] Jack Goldsmith. The failure of internet freedom. *Emerging Threats (New York: Knight First Amendment Institute, 2018)*, pages 9–12, 2018.

[63] Sergio Gómez. Centrality in networks: finding the most important nodes. In *Business and Consumer Analytics: New Ideas*, pages 401–433. Springer, 2019.

[64] Marta Gomez-Barrero, Emanuele Maiorana, Javier Galbally, Patrizio Campisi, and Julian Fierrez. Multi-biometric template protection based on homomorphic encryption. *Pattern Recognition*, 67:149–163, 2017.

[65] Qiyuan Gong, Junzhou Luo, Ming Yang, Weiwei Ni, and Xiao-Bai Li. Anonymizing 1:m microdata with high utility. *Knowledge-Based Systems*, 115(Supplement C):15 – 26, 2017.

[66] Qiyuan Gong, Ming Yang, Zhouguo Chen, Wenjia Wu, and Junzhou Luo. A framework for utility enhanced incomplete microdata anonymization. *Cluster Computing*, 20(2):1749–1764, Jun 2017.

[67] Xinyu Gong. 5g and china-us relations: Competition and intervention. 2021.

[68] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Theory of cryptography conference*, pages 339–356. Springer, 2012.

[69] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *9th IEEE International Conference on Data Mining (ICDM'09)*, pages 169–178. IEEE, 2009.

[70] 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.

[71] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1447–1458. ACM, 2014.

[72] Shen-Shyang Ho and Shuhua Ruan. Differential privacy for location pattern mining. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, pages 17–24. ACM, 2011.

[73] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE, 2014.

[74] Anco Hundepool and LCRJ Willenborg. μ-and τ-argus: Software for statistical disclosure control. In *Third International Seminar on Statistical Confidentiality*, 1996.

[75] Wuxuan Jiang, Cong Xie, and Zhihua Zhang. Wishart mechanism for differentially private principal components analysis. In *AAAI*, pages 1730–1736, 2016.

[76] Murat Kantarcioglu, Ali Inan, and Mehmet Kuzu. Anonymization toolbox, 2010.

[77] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[78] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference (TCC)*, pages 457–476. Springer, 2013.

[79] Leo Kelion. Amazon: How bezos built his data machinehttps://www.bbc.co.uk/news/extra/clqyzenmbi/amazon-data.

[80] Daniel Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD Conference*, pages 127–138, 2009.

[81] 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.

[82] Nitin Kohli and Paul Laskowski. Epsilon voting: Mechanism design for parameter selection in differential privacy. In *2018 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 19–30. IEEE, 2018.

[83] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 933–943. International World Wide Web Conferences Steering Committee, 2018.

[84] Vito Laterza. Could cambridge analytica have delivered donald trump's 2016 presidential victory? an anthropologist's look at big data and political campaigning. *Public Anthropologist*, 3(1):119–147, 2021.

[85] Glenn Lawyer. Understanding the influence of all nodes in a network. *Scientific reports*, 5(1):1–9, 2015.

[86] Jaewoo Lee and Chris Clifton. How much is enough? choosing ε for differential privacy. In *International Conference on Information Security*, pages 325–340. Springer, 2011.

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

[88] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Workload-aware anonymization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–286, 2006.

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

[90] Jure Leskovec. *Autonomous systems AS-733 @ONLINE*, 2000. https://snap.stanford.edu/data/as-733.html.

[91] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[92] James Andrew Lewis. Criteria for security and trust in telecommunications networks and services, May 2020.

[93] B. Li, Y. Liu, X. Han, and J. Zhang. Cross-bucket generalization for information and privacy preservation. *IEEE Transactions on Knowledge and Data Engineering*, 30(3):449–459, March 2018.

[94] Frank Li, Richard Shin, and Vern Paxson. Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners. In *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop*, pages 53–64. ACM, 2015.

[95] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

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

[97] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 32–33, New York, NY, USA, 2012. ACM.

[98] Tiancheng Li and Ninghui Li. Injector: Mining background knowledge for data anonymization. In *ICDE*, pages 446–455, 2008.

[99] Tiancheng Li, Ninghui Li, Jian Zhang, and Ian Molloy. Slicing: A new approach for privacy preserving data publishing. *IEEE Trans. Knowl. Data Eng.*, 24(3):561–574, 2012.

[100] Xiang-Yang Li, Chunhong Zhang, Taeho Jung, Jianwei Qian, and Linlin Chen. Graph-based privacy-preserving data publication. In *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2016.

[101] Zhuolin Li, Xiaolin Zhang, Haochen Yuan, Yongping Wang, and Jian Li. Distributed privacy preserving technology in dynamic networks. *International Journal of High Performance Computing and Networking*, 15(3-4):223–232, 2019.

[102] Kun Liu, Kamalika Das, Tyrone Grandison, and Hillol Kargupta. Privacy-preserving data analysis on graphs and social networks. In *Next Generation of Data Mining (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*, pages 419–438, 2008.

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

[104] Zhenpeng Liu, Yawei Dong, Xuan Zhao, and Bin Zhang. A dynamic social network data publishing algorithm based on differential privacy. *Journal of Information Security*, 8(04):328, 2017.

[105] Xuesong Lu, Yi Song, and Stéphane Bressan. Fast identity anonymization on graphs. In *International Conference on Database and Expert Systems Applications*, pages 281–295. Springer, 2012.

[106] Michael J. Lyons, Julien Budynek, and Shigeru Akamatsu. Japanese female facial expressions (jaffe), database of digital images, 1997.

[107] Tinghuai Ma, Yuliang Zhang, Jie Cao, Jian Shen, Meili Tang, Yuan Tian, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. Kdvem: a k-degree anonymity with vertex and edge modification algorithm. *Computing*, 97(12):1165–1184, 2015.

[108] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. *l*-diversity: Privacy beyond *k*-anonymity. In *Proceedings of the*

22nd IEEE International Conference on Data Engineering (ICDE 2006), Atlanta Georgia, April 2006.

[109] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE, 2006.

[110] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. *l*-diversity: Privacy beyond *k*-anonymity. *ACMTransactions on Knowledge Discovery from Data (TKDD)*, 1(1), March 2007.

[111] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[112] Tereza Machova. The discourse of surveillance and privacy: biopower and panopticon in the facebook-cambridge analytica scandal, 2021.

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

[114] Bill Marczak, John Scott-Railton, Sarah McKune, Bahr Abdul Razzak, and Ron Deibert. Hide and seek: Tracking nso group's pegasus spyware to operations in 45 countries. Technical report, 2018.

[115] Sjouke Mauw, Yunior Ramírez-Cruz, and Rolando Trujillo-Rasua. Robust active attacks on social graphs. *Data mining and knowledge discovery*, 33(5):1357–1392, 2019.

[116] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[117] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[118] Bharadwaj Mehrotra and Chhaya Abhinav. Pegasus has given privacy legislation a jab of urgency. 2021.

[119] Yvonne Mülle, Chris Clifton, and Klemens Böhm. Privacy-integrated graph clustering through differential privacy. In *EDBT/ICDT Workshops*, volume 157, 2015.

[120] Maurizio Naldi and Giuseppe D'Acquisto. Differential privacy: An estimation theory-based method for choosing epsilon. *arXiv preprint arXiv:1510.00917*, 2015.

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

[122] Mohamed Nassar, Nathalie Wehbe, and Bechara Al Bouna. K-nn classification under homomorphic encryption: application on a labeled eigen faces dataset. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 546–552. IEEE, 2016.

[123] Ahmet Erhan Nergiz and Chris Clifton. Query processing in private data outsourcing using anonymization. In *The 25th IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSEC-11)*, Richmond, Virginia, July 11-13 2011.

[124] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 665–676, 2007.

[125] Hiep Nguyen, Abdessamad Imine, and Michaël Rusinowitch. Network structure release under differential privacy. *Transactions on Data Privacy*, 9(3):26, 2016.

[126] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[127] Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, David R O'Brien, and Salil Vadhan. Differential privacy: A primer for a non-technical audience. In *Privacy Law Scholars Conf*, 2017.

[128] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. Ranking of closeness centrality for large-scale social networks. In *International workshop on frontiers in algorithmics*, pages 186–195. Springer, 2008.

[129] Victor Powell and Lewis Lehe. Principal component analysis @ONLINE, 2015.

[130] Fabian Prasser, Florian Kohlmayer, Ronald Lautenschläger, and Klaus A Kuhn. Arx-a comprehensive tool for anonymizing biomedical data. In *AMIA Annual Symposium Proceedings*, volume 2014, page 984. American Medical Informatics Association, 2014.

[131] Davide Proserpio, Sharon Goldberg, and Frank McSherry. A workflow for differentially-private graph synthesis. In *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, pages 13–18. ACM, 2012.

[132] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets. *Proceedings of the VLDB Endowment*, 7(8):637–648, 2014.

[133] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 425–438, 2017.

[134] Gu Qiuyang, Ni Qilian, Meng Xiangzhao, and Yang Zhijiao. Dynamic social privacy protection based on graph mode partition in complex social network. *Personal and Ubiquitous Computing*, 23(3-4):511–519, 2019.

[135] Roxana Radu and Cedric Amon. The governance of 5g infrastructure: between path dependency and risk-based approaches. *Journal of Cybersecurity*, 7(1):tyab017, 2021.

[136] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 107–116, 2009.

[137] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, pages 531–542. Citeseer, 2007.

[138] Paul Ressel. De Finetti-type theorems: an analytical approach. *Ann. Probab.*, 13(3):898–922, 1985.

[139] Linda Risso. Harvesting your soul? cambridge analytica and brexit. *Brexit Means Brexit*, 2018:75–90, 2018.

[140] Franccois Rousseau, Jordi Casas-Roma, and Michalis Vazirgiannis. Community-preserving anonymization of graphs. *Knowledge and Information Systems*, 54(2):315–343, 2018.

[141] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.

[142] 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.

[143] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[144] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Tech. rep. SRI-CSL-98-04, SRI Computer Science Laboratory*. technical report, SRI International, 1998.

[145] Mark Scott. Cambridge analytica did work for brexit groups, says ex-staffer, Jul 2019.

[146] Entong Shen and Ting Yu. Mining frequent graph patterns with differential privacy. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 545–553, 2013.

[147] Gajendra Singh, Vinod Sharma, and Uday Singh. Different image encryption techniques-survey and overview. 2016.

[148] Jordi Soria-Comas and Josep Domingo-Ferrer. Differential privacy via t-closeness in data publishing. In *Eleventh Annual International Conference on Privacy, Security and Trust, PST 2013, 10-12 July, 2013, Tarragona, Catalonia, Spain, July 10-12, 2013*, pages 27–35, 2013.

[149] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 26–37. ACM, 2016.

[150] Latanya Sweeney. *Computational disclosure control: a primer on data privacy protection*. PhD thesis, Massachusetts Institute of Technology, 2001.

[151] Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[152] Motahareh Taheri, Saeed Mozaffari, and Parviz Keshavarzi. Face authentication in encrypted domain based on correlation filters. *Multimedia Tools and Applications*, pages 1–25, 2017.

[153] Chih-Jui Lin Wang, En Tzu Wang, and Arbee LP Chen. Anonymization for multiple released social network graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 99–110. Springer, 2013.

[154] Hui (Wendy) Wang and Ruilin Liu. Hiding outliers into crowd: Privacy-preserving data publishing with outliers. *Data & Knowledge Engineering*, 100 Part A:94 – 115, 2015.

[155] Ke Wang, Peng Wang, Ada Waichee Fu, and Raymond Chi-Wing Wong. Generalized bucketization scheme for flexible privacy settings. *Information Sciences*, 348:377 – 393, 2016.

[156] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *Transactions on data privacy*, 6(2):127, 2013.

[157] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 329–340. Springer, 2013.

[158] Yue Wang, Xintao Wu, Jun Zhu, and Yang Xiang. On learning cluster coefficient of private networks. *Social network analysis and mining*, 3(4):925–938, 2013.

[159] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, Philip S. Yu, and Jian Pei. Can the utility of anonymized data be used for privacy breaches? *ACM Trans. Knowl. Discov. Data*, 5(3):16:1–16:24, August 2011.

[160] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th international conference on extending database technology*, pages 111–122, 2010.

[161] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[162] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *Proceedings of 32nd International Conference on Very Large Data Bases (VLDB 2006)*, Seoul, Korea, September 12-15 2006.

[163] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

[164] Yunyun Yang, Gang Xie, and Jun Xie. Mining important nodes in directed weighted complex networks. *Discrete Dynamics in Nature and Society*, 2017, 2017.

[165] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *proceedings of the 2008 SIAM International Conference on Data Mining*, pages 739–750. SIAM, 2008.

[166] Haifeng Yu, Phillip B Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 3–17. IEEE, 2008.

[167] Haifeng Yu, Michael Kaminsky, Phillip B Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278, 2006.

[168] Liangwen Yu, Yonggang Wang, Zhengang Wu, Jiawei Zhu, Jianbin Hu, and Zhong Chen. Edges protection in multiple releases of social network data. In *International Conference on Web-Age Information Management*, pages 669–680. Springer, 2014.

[169] Rong Yue, YiDong Li, Tao Wang, and Yi Jin. An efficient adaptive graph anonymization framework for incremental data publication. In *5th International Conference on Behavioral, Economic, and Socio-Cultural Computing*, pages 103–108. IEEE, 2018.

[170] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *International Workshop on Privacy, Security, and Trust in KDD*, pages 153–171. Springer, 2007.

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

[172] 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.