Improving Privacy Preserving in Modern Applications

By
Mengmeng Yang

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Deakin University
September 2018
I am the author of the thesis entitled

*Improving Privacy Preserving in Modern Applications*

submitted for the degree of *Doctor of Philosophy*

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'  

Full Name: ..........Mengmeng Yang.................................................................

Signed: ..............................................................................................................

Date: ..........................24/01/2019.................................................................
I certify the following about the thesis entitled (10 word maximum)

**Improving Privacy Preserving in Modern Applications**

submitted for the degree of **Doctor of Philosophy**

a. I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.

b. The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.

c. That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

d. That any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

e. All research integrity requirements have been complied with.

'I certify that I am the student named below and that the information provided in the form is correct'

**Full Name:** .................................................. **Mengmeng Yang** .................................................................

**Signature Redacted by Library**

**Signed:** ............................................................................................................................................................

**Date:** ........................................................................... **24/01/2019** .............................................................................
DEAKIN UNIVERSITY
SCHOOL OF
INFORMATION TECHNOLOGY

The undersigned hereby certify that they have read and recommend to the Faculty of Science and Technology for acceptance a thesis entitled “Improving Privacy Preserving in Modern Applications” by Mengmeng Yang in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: September 2018

External Examiner: 

Research Supervisor: Jingyu Hou

Examing Committee: 

ii
To my family...
Table of Contents

Table of Contents iv

List of Tables viii

List of Figures ix

Acknowledgements xi

Publication List xii

Abstract xiv

1 Introduction 1
  1.1 Background ....................................................... 1
  1.2 Research Questions ............................................. 3
  1.3 Thesis Outline .................................................. 6

2 Literature Review 9
  2.1 Privacy Attacks .................................................. 9
    2.1.1 Tabular Data Attack ....................................... 9
    2.1.2 Graph Data Attack ......................................... 12
    2.1.3 Location Privacy Attack .................................. 15
    2.1.4 Attacks for Other Applications ........................... 17
  2.2 Privacy Technologies ............................................ 18
    2.2.1 Encryption .................................................. 18
    2.2.2 Anonymization ............................................. 20
    2.2.3 Differential Privacy ....................................... 22
    2.2.4 Other Technologies ......................................... 28
  2.3 Privacy Metrics ................................................ 30
5 Privacy-Preserving in Location-based Services

5.1 Introduction .................................................. 101
5.2 Location Privacy Preservation Framework based on a Third Party ... 105
  5.2.1 Notations ................................................. 105
  5.2.2 Problem Definition and Assumption ......................... 105
  5.2.3 Framework ............................................... 106
  5.2.4 Privacy Protection Scheme based on Johnson-Lindenstrauss Transform .................................................. 108
5.3 Privacy Protection Algorithms for Two Basic POI queries .......... 113
  5.3.1 K-nearest POI Queries ................................ 113
  5.3.2 Range Queries ......................................... 119
5.4 Privacy and Utility .......................................... 123
  5.4.1 Privacy Analysis ........................................ 123
  5.4.2 Utility Analysis ........................................ 124
5.5 Evaluation and Discussion .................................. 127
  5.5.1 Experimental Setup ................................... 127
  5.5.2 Performance of the Proposed Method ..................... 130
  5.5.3 The Impact of POI Density ................................ 136
  5.5.4 The Impact of Transition Matrix $M$ .................... 141
  5.5.5 Overhead Analyses .................................... 142
5.6 Related Work ................................................. 143
5.7 Summary ...................................................... 145

6 Privacy-preserving in Crowdsensing System .......................... 147
6.1 Introduction .................................................. 147
6.2 Fundamentals of Crowdsensing ................................ 150
  6.2.1 Crowdsensing .......................................... 150
  6.2.2 Framework ............................................. 151
6.3 Mobile Crowd sensing under Differential Privacy Protection .... 152
  6.3.1 Problem Definition and Assumptions ..................... 152
  6.3.2 Sanitized Data Release ................................ 154
List of Tables

3.1 Notations .............................................. 45
3.2 Characteristics of the datasets .......................... 62
3.3 Values of various parameters ......................... 64

4.1 Notations .............................................. 75
4.2 New query generation ................................. 79
4.3 Parameters ............................................. 88

5.1 Notations .............................................. 105
5.2 Information access .................................. 112
5.3 Parameter settings .................................. 129

6.1 Notations .............................................. 153
6.2 Parameter Setting .................................... 168
# List of Figures

1.1 Key contributions of the proposed methods under four application scenarios .................................................. 8

2.1 A degree attack [148] .................................................................................................................. 10
2.2 A degree attack ......................................................................................................................... 12
2.3 A neighbourhood attack ........................................................................................................ 13
2.4 An embedded sub-graph attack .............................................................................................. 14
2.5 A fingerprint attack .................................................................................................................. 14
2.6 A mutual friends attack ........................................................................................................... 15

3.1 The Johnson–Lindenstrauss recommendation process ......................................................... 51
3.2 Description of item rating ......................................................................................................... 62
3.3 Impact of $m$ on prediction performance ............................................................................. 65
3.4 Comparison with other related works .................................................................................. 67
3.5 Comparison with non-privacy method .................................................................................... 69

4.1 The private multifunctional aggregation system model ..................................................... 76
4.2 Aggregation process ................................................................................................................. 78
4.3 Performance with different sizes of query set .................................................................... 89
4.4 Performance with different sizes of query set .................................................................... 90
4.5 Performance with different privacy budgets ......................................................................... 92
4.6 Performance with different privacy budgets ......................................................................... 93
4.7 Performance with different sizes of training set .................................................................. 96
4.8 Performance with different sizes of training set

5.1 Client-server structure

5.2 Location privacy framework

5.3 Map sanitization

5.4 The process of $k$-anonymization

5.5 JL transform for $k$-NN queries

5.6 JL transform for range queries with radius $r$

5.7 JL transform for range queries with radius $\hat{r}$

5.8 Location placement

5.9 $k$-NN query performance

5.10 $k$-NN query performance

5.11 Range query performance

5.12 Range query performance

5.13 The effect of density on the $k$-NN and range queries

5.14 The effect of density on the $k$-NN and range queries

5.15 The effect of $m$ on $k$-NN and range queries

5.16 Computational cost of the proposed method

6.1 A framework for private spatial crowdsensing

6.2 Partition data domain

6.3 Performance by varying $\epsilon$

6.4 Performance by varying $\epsilon$

6.5 The performance by varying MTD

6.6 The performance by varying ESR

7.1 Key contributions of the proposed methods under four application scenarios
Acknowledgements

I would like to thank a number of people who constantly provided me with advice, help, support and understanding during the time at Deakin University working on my PhD.

Firstly, I would like to express my deepest gratitude and appreciation to my supervisors Dr. Tianqing Zhu, Prof. Wanlei Zhou and Prof. Yang Xiang for their many suggestions, friendly encouragement and constant support during my research over the last three years. They guided me through difficulties when I struggled and encouraged me when I feel depressed. I appreciate all the time they have spent on discussing my research ideas, listening to my problems, editing my papers, and taking away all my doubts and worries. I am honoured to have received their supervision.

Second, I would also like to thank the academic and general research staff of the School of Information Technology, Deakin University. Thank Ms. Alison Carr, Ms. Lauren Fisher and Ms. Kathy Giulieri for their great secretarial work that enabled me to attend academic conferences. I also owe a great deal of thanks to my research friends who helped me during my research and daily life during my PhD study: Lichuan Ma, Bo Liu, Liu Liu, Longxia Huang, and so many others.

I am also grateful to my husband and parents for their patience and love. Without them, this work would never have come into existence.

Melbourne, Australia
Mengmeng Yang
September
Publication List

Submitted Papers


Refereed Journal Articles


**Refereed Conference Papers**


Abstract

Big data has great potential to revolutionize our lives through its predictive power and provide insights far beyond what we know about ourselves. It has become a necessity in today's world. It is expected to grow exponentially in the coming years. Big data has brought us great benefits, for example, advertisements focused on what you actually want to buy, wearable or implantable devices that can monitor your health and notify your doctor if something is going wrong. However, it can also cause great privacy issues. As more and more personal information is collected by service providers, the data collected reveals personal sensitive information, such as where they are, who they communicate with, what they buy, what they watch, what disease they have, and so on. Huge data sets can be used not only for legitimate purposes but also for abuse. Due to the possibility of malicious use, people pay more attention to the security and privacy threats while enjoying the benefits. Besides, the data privacy problem may create obstacles to the anticipated growth and opportunities of Big Data. How should we address these privacy concerns without denying the benefits of big data? This thesis aims to find out privacy issues in various modern applications, such as recommendation system, Internet of Things, location-based service and crowdsourcing, improve privacy protection levels, and at the same time maintain the availability of data.

First, we explore the privacy problem in the recommendation system. Neighbourhood-based collaborative filtering is a popular recommendation method that is particularly vulnerable to k nearest neighbour (KNN) attack, that the attacker can obtain the rating history of the victim by searching the list of recommended items. To solve
this problem. We propose a Johnson-Lindenstrauss-based method to preserve the information of individual users while improving the performance of the recommender system.

Second, we explore the privacy problem in the Internet of Things. Data aggregation is considered to be an essential research topic in the Internet of Things, and privacy is an important issue for data aggregation, as the sensory data may be sensitive. For example, the health data, such as blood pressure, the temperature can disclose the user’s health status, and the utility data, such as the electricity, can be used to profiling the customers’ life pattern and routine. To solve the problem, we propose a machine learning based privacy preserving multifunctional data aggregation method, which allows multiple aggregation functions calculation without disclosing the user’s privacy.

Third, we explore the privacy problem in the location-based services. A typical example of location-based services is POI query service, as users have to submit their own location in exchange of services. Their location information will be disclosed to the service provider, which can be untrusted. To hide the user’s location information, we propose a new privacy framework that includes a semi-trusted third party. Under our privacy framework, neither the server nor the third party knows the exact location of the user.

Finally, we explore the privacy problem in the crowdsourcing applications. The workers contribute data collected by their mobile devices to employers for rewards. However, to be able to assign tasks more efficiently, workers need to submit their exact location to the Server as well. Different from traditional location privacy-preserving, the overall workers’ position distribution need to be considered in task assignment stage. To solve the privacy problem in the crowdsourcing application, we propose a worker’s density based privacy preserving data release method. The method achieves differential privacy that enables the participation of workers without disclosing their location privacy.
In summary, this thesis makes following contributions: 1) propose a privacy-preserving multifunctional data aggregation method solve the privacy problem in Internet of things; 2) propose a Johnson-Lindenstrauss-based method and personalized privacy-preserving collaborative filtering method solve the privacy problem in recommendation system; 3) propose a new semi-trusted third party based privacy framework that protects the user’s location information in POI query services; 4) propose a worker’s density based privacy-preserving data release method that enable effective task allocation without disclosing worker’s privacy in crowdsourcing application. These contributions enable rapid development of big data applications.

Keywords: Privacy-preserving, Recommendation system, Internet of Things, Location-based services, Crowdsensing applications.
Chapter 1

Introduction

1.1 Background

Due to the rapid growth of computers and the technology that is capable of capturing data, the data is increasing exponentially, they are collected from the everyday interactions with digital products or services, including mobile devices, information sensing, social media and so on. The big data allows today’s advanced analytic technology to reveal hidden patterns or identify secret correlations that enable us to learn and make smarter decisions. For example, companies can now accurately predict what their customers want to buy, and when. In addition, the big data change our world in a number of ways. For example, data-driven medicine help to spot disease early by analysing a large number of medical records; sensor data can be analysed to predict and response to natural and man-made disasters.

Big Data gives us unprecedented insights and opportunities, but the collected big data contains a large amount of personal or sensitive information, which raises big privacy concerns. For example, website cookies have been used to track web browsing for a long time, which can be used to see what the user has been doing on
the web; services like Google Maps track the user’s location information at real time by default, which could disclose the user’s life pattern; on the social network, many users regularly post pictures with their location tagged, which let people know where they are. These privacy threats limit the development of big data when the user’s private information is not properly protected.

A variety of techniques have been proposed to protect the privacy information. $k$-anonymity [125] is the most popular privacy-preserving technique. It guarantees that the information for each person cannot be distinguished from at least $k - 1$ other records. The $k$-anonymity can be achieved by suppression or generalization method. $l$-diversity [90] requires the group of records to contain at least $l$ different sensitive values. $l$-diversity handles the weakness of the $k$-anonymity that protects the identity is not enough to protect the sensitive attributes of the individual. To prevent skewness attack, Li et al. proposed a privacy model, called t-closeness [77], which requires the distribution of the sensitive attributes in the group to be close to the distribution of the attribute in the overall dataset. Many other extensions based on these techniques were also proposed, such as $(X, Y)$-anonymity [149], $(c, l)$-diversity [90], $(X, Y)$-privacy [149] and so on. Generally, these methods can only defend one kind of specific attack and have no ability to resist the newly developed approaches and multiple background knowledge attacks.

Differential privacy, which is proposed by Dwork [32] in 2006, is a mechanism that makes little difference to the results of the query with the addition or deletion of any tuple by adding random noise on the output. It works well on the tabular dataset preserving privacy, and also be applied to many other application scenarios [92, 160, 165, 178], because it does not need to model background knowledge that
is still a challenge for traditional anonymization methods. In other word, differential privacy has a very strong assumption about the attacker’s background knowledge that it assumes the attacker knows everything except the information he wants to know. Therefore, differential privacy can defend the various kinds of background knowledge attack. Besides, differential privacy is based on mathematics, which provides a quantitative assessment method and makes the level of privacy protection comparable. However, differential privacy introduces lots of noise to hide the differences between the query results, which affects the utility of the perturbed dataset.

1.2 Research Questions

This thesis studies the privacy problems in four application scenarios, recommendation system, Internet of Things, location-based services and crowdsensing system. We introduce the research questions examined in each application scenario in detail as follow.

**Recommendation system.** Recommendation systems have become increasingly popular as a result of the significant growth in online information. Instead of exploring the entire content of the Web, users can easily obtain related information via recommendation. However, a recommendation system requires a substantial amount of historical user data to generate accurate predictions. The release of such data to a recommendation system raises concerns over user privacy. We focus on the neighbourhood-based collaborative filtering method, which is particularly vulnerable to attacks on privacy. There are many protection methods are proposed including differential privacy protection. Unfortunately, existing privacy preservation methods
based on differential privacy protect user privacy at the cost of utility, aspects of which have to be sacrificed to ensure that privacy is maintained. How to protect the user’s privacy properly, while enhancing the utility is an essential challenge for differential privacy in the recommendation system.

The traditional way of achieving differential privacy is adding noise, such as the Laplace mechanism, which inevitably reduces the utility of the dataset. Can we find a solution that defuses the conflict between privacy and utility in the recommendation system?

**Internet of Things.** Data aggregation plays an important role in the Internet of Things. The study and analysis of aggregated data provide various services and benefits for people. However, the privacy raises a big concern, as the raw sensory data may be sensitive and disclose the users sensitive information. To achieve the object that preserves the users data privacy, numerous schemes have been proposed in the last decades. Most methods are based on encryption technology, which computationally and communicationally expensive. In addition, most methods only can calculate a single aggregation function. While in practice, the idea aggregation scheme should allow the flexible aggregation queries to meet the diversified aggregation goals. How to achieve multifunctional aggregation within only one round communication and reduce system overhead, while preserving the data privacy is a challenge.

We prefer to apply differential privacy technology to guarantee the data privacy. However, for multifunctional aggregation, the correlations between queries increase the sensitivity significantly, which reduces the accuracy of the aggregation result. How can we increase the utility without consuming the privacy-preserving level?
**Location-based Services.** The growing popularity of location-based services is giving untrusted servers relatively free reign to collect huge amounts of location information from mobile users. This information can reveal far more than just a user’s location but other sensitive information, such as the user’s interests or daily routines, which raises strong privacy concerns. A typical application of location-based services is Point of Interest (PoI) query that returns the user’s nearby PoIs. Existing solutions based on differential privacy are to add a large amount of noise to hide the worker’s real location, which reduces the accuracy of returned PoIs. Especially for high level privacy protection. Can we improve the accuracy of returned PoIs without lowering the level of privacy protection?

**Crowdsensing system.** Mobile crowdsensing techniques use workers with mobile devices to collect data and send it to the task requester for rewards. However, to ensure the optimal allocation of tasks, a centralized server needs to know the precise location of each user, but exposing the workers’ exact locations raises privacy concerns. A typical solution of differential privacy protects location privacy in the crowdsensing system by introducing a trusted third party. The third party partitions the domain of worker locations into small cells and release the perturbed statistical results of each cell to the crowdsensing server. However, the method is based on the assumption that the distribution of workers is uniform, which is not practical in real life and this uneven distribution would cause significant errors during the task assignment process. How should we solve the problem of uneven distribution of workers and improve the success rate of task assignment?
1.3 Thesis Outline

This section aims to establish the structural organization of the thesis. According to four research issues addressed in this thesis: privacy-preserving in the recommendation system, privacy-preserving in the Internet of Things, privacy-preserving in location-based services and privacy-preserving in crowdsensing system, the chapters are organized as follows.

- Chapter 2 discusses the related privacy-preserving works in the following application scenarios, recommendation system, Internet of Things, location-based services and crowdsensing system, including the privacy techniques used, advantages and disadvantages of the proposed method.

- Chapter 3 addresses the privacy-preserving problem in the recommendation system, mainly focuses on the neighbourhood-based collaborative filtering method. This chapter adopts a novel data perturbation method, which perfectly combined with the neighbourhood-based collaborative filtering recommendation method. Specifically, the Johnson-Lindenstrauss transform is used to project the original rating dataset to a lower dimensional space. This process not only maintains the relative distance between records, we theoretically prove that this transform on the rating dataset satisfies differential privacy. In addition, this randomized process enhanced the accuracy of the recommendation.

- Chapter 4 proposes a privacy-preserving method for data aggregation, which plays an important role in the Internet of Things. This chapter proposes a multifunctional data aggregation method under differential privacy. The proposed
method is based on machine learning and can support a wide range of statistical aggregation functions, including both additive and non-additive aggregation. Also, the method is designed under fog computing architecture, which extends the cloud computing to the edge of the network that alleviates the cloud burden and improves the communication efficiency by reporting the aggregation results only to the cloud server.

- Chapter 5 presents a new privacy framework to protect the location privacy in the Point of Interest query service. The proposed privacy framework includes a semi-trusted third party. Under the privacy framework, both the server and the third party only hold a part of the users location information. Neither the server nor the third party knows the exact location of the user. In addition, benefit from the character of Johnson-Lindenstrauss transform, though the real location information is perturbed, the relative distances between the user and PoIs are nearly not changes. Therefore, the proposed method improves the performance significantly, while providing a strict privacy guarantee.

- Chapter 6 proposes a data release mechanism for crowdsensing techniques that satisfies differential privacy, providing rigorous protection of worker locations. The partitioning method is based on worker density and considers non-uniform worker distribution. In addition, this chapter proposes a geocast region selection method for task assignment that effectively balances the task assignment success rate with worker travel distances and system overheads.

- Chapter 7 summarizes the contributions of this thesis, and presents some possible suggestions and extensions for further research.
Chapter 2

Literature Review

This chapter provides an extensive literature review on privacy attack models, privacy-preserving technologies and privacy metrics. We systematically analyze how the data privacy can be disclosed, what kind of privacy technology should be used in a specific application scenario, and how to evaluate the privacy provided by the proposed solution.

2.1 Privacy Attacks

We classify the privacy attack models according to the type of the dataset. Specifically, we survey the privacy attacks for the tabular dataset, graph dataset, location dataset and other applications.

2.1.1 Tabular Data Attack

The tabular dataset is an arrangement of data in rows and columns, each row represents a record of an individual and each column is an attribute of the person. The
form of the table is

\[ D(\text{Explicit}-\text{identifier}, \text{Quasi}-\text{identifier}, \text{Sensitive}-\text{attributes}, \text{Non}-\text{sensitive attributes}) \]

(2.1.1)

where \text{Explicit}-\text{identifier} is a set of attributes can identify the person’s identity directly, such as the name. \text{Quasi}-\text{identifier} is a set of attributes can potentially identify the person’s identity. \text{Sensitive}-\text{attributes} is a set of sensitive attributes such as disease and salary. \text{Non}-\text{sensitive attributes} is a set of attributes outside of previous scopes.

Normally, the dataset is published without the \text{Explicit}-\text{identifier}, however, it is not enough to protect the individual’s privacy. Various attack model can be used to identify the person’s identity or the sensitive information. We summaries the typical attack models as follows.

**Record Linkage Attack.** Record linkage attack identifies the victim’s record in one table by linking the record to a record in another public table.

For example, a hospital published the patient records as shown in Fig. 2.1a, assume the attacker has access to an external table shown in Fig. 2.1b, and the

![Patient table](image1.png) ![External table](image2.png)

Figure 2.1: A degree attack [148]
person who has the record in table $a$ has the record in table $b$. Combining the tables in the common attributes will link the victim’s identity to his disease. For example, Bob is identified as a hepatitis patient by linking attributes $<\text{Engineer, Male, 35}>$ after the join. Minkus et al. [96] successfully identified the Facebook user who resides in the city by linking the Facebook user’s social ties to the city’s voter registration records.

**Attributes Linkage Attack.** Attributes linkage attack may not identify the record of the victim, but can infer the victim’s sensitive value by observing the sensitive values belong to the same group. For example, from table $a$, the attacker can confidently infer that all the female writers at age 30 have Flu, and all the female dancers at age 30 have HIV.

Both record and attribute linkage attack assume the attacker already knows that the victim’s record already exists in the released table.

**Table Linkage Attack.** Table linkage attack identifies whether the victim’s record exists in the released table. As in some cases, the presence or absence of the victim’s record already reveals his sensitive information.

For example, assume the table $a \subseteq$ table $b$. There are three records in table $b$ with attributes $<\text{Dancer, Female, 30}>$, and two records in table $a$ with same attributes. Therefore, the attacker can infer that Emily’s record has $2/3$ probability presents in table $a$. The presence of her records reveals that she is an HIV patient.
2.1.2 Graph Data Attack

The social network is a very popular platform where people make new friends and share their interests. More and more social network data are published for many applications, such as social network analysis and data mining. Unlike the case in traditional relational datasets, all identities in the social network are connected with each other by edges, which allow the attackers to launch the structural based attacks.

Background knowledge plays an important role in modelling privacy attacks. This section summaries the background knowledge can be used by the attacker to launch structure attack.

**Vertices Degree.** Vertices degree represents how many direct connections between a node and its neighbours. Once the degree of the user is different from others in the graph, the vertex is re-identified. For example, in Fig. 2.2, node 3 and node 4 can be identified directly if the adversary knows Carl has three friends and Danna has only one friend.

![Figure 2.2: A degree attack](image)

Tai et al. [137] identified a new attack called friendship attack, which is based on degree pair of an edge. They launched both degree and friendship attacks on the 20Top-Conf dataset and proved that the friendship attack causes a much more privacy disclosure than the degree attack.
Neighbourhood. Neighbourhood refers to the neighbours of an individual who have connections with each other. Attackers make use of this kind of structural information to identify individuals [179]. For example, in Fig. 2.3, if attackers know Bob has three friends and two neighbours and they connected with each other, Bob can be recognized in the anonymized graph.

Figure 2.3: A neighbourhood attack

Ninggal et al. [103] proposed another kind of attack called neighbourhood-pair attack, which uses a pair of neighbourhood structural information as background knowledge to identify victims. Such attacks assume attackers know more information than neighbourhood attacks do, so attackers have a higher chance to distinguish users in a dataset.

Embedded sub-graph. Sub-graph refers to a subset of the whole graph. Some adversaries create few fake nodes and build links using a specific way before the data is published, and then match the target graph with a reference graph based on the sub-graph which has been planted. In Fig. 2.4, the grey part is the original graph, the black part is the sub-graph embedded by the adversary. Normally, the embedded sub-graph is unique and easy for attackers to identify after the dataset is released.

Link Relationship. The relationship between two vertices also can be acquired by an adversary. Wang et al. [155] considered that the public users’ identities are public
Figure 2.4: An embedded sub-graph attack

and not sensitive. They utilized the connection between victims and public users to perform attacks. For example, in Fig. 2.5, A, B, and C are public users, such as BBC and Michael Jackson. Their identities are publicity, and if attackers know vertex d has one hop to A and C and two hops to B, d can be identified.

![Diagram](image)

**Figure 2.5: A fingerprint attack**

Sun et al. [134] committed a mutual friend attack. Their algorithm identifies a pair of users who connect to each other based on the number of mutual friends. For example, in Fig. 2.6, the numbers on the edge represent the number of mutual friends between two nodes. If the adversary knows Alice and Ford have two mutual friends, then she/he can identify a and c combined with other reference information (e.g. degree).

**Attributes of Vertices.** Attributes of individuals in a social network are represented as labels to vertices. Attackers may get the attributes of victims, such as age,
2.1.3 Location Privacy Attack

Location privacy disclosure happens when the location information leaves the user’s sensing device. To hide the user’s exact location information, privacy-preserving technologies are performed on it. Even though, the user location information still can be inferred. We surveyed the location attack models in this section as follows.

**Context Linking Attack.** In the context linking attack, the attacker utilises the context information of the victim to decrease the privacy. For example, Gruteser and Grunwald [48] proposed a *personal context linking attack*, which is based on personal context information, such as personal preferences and habits. For example, if the attacker knows the victim would like to visit the club at a certain time, and the victim simply protects his location by cloaking region, the attacker can increase the infer accuracy simply by decreasing the cloaking region to the club within the region.
Krumm [74] proposed a *map matching attack* that decrease the obfuscated area by removing the irrelevant areas, such as lakes. Shokri et al. [131] proposed a *probability distribution attack*, which utilize the victim’s distribution context information to infer the victim’s location. Specifically, the attacker gets a probability distribution function of the user position in the obfuscated area. Then, the attacker can identify the area where the victim located at with high probability.

**Region Intersection Attack.** Region intersection attack [138] increases the precision of the obfuscated location by calculating the intersections between the user’s multiple imprecise position updates or queries. The intersection can be used to infer the user’s sensitive region. For trajectory privacy protection, a series of obfuscated regions are generated when the user reaches a spot, the intersection between the obfuscated regions reduces the user’s privacy significantly.

**Machine Learning based Attack.** Li et al. [76] proposed a machine learning based attack method to infer users’ demographics from the disclosed locations. To infer a specific attribute of a user, they identify a group of users who have similar location traces with each other, and check the attributes of the users in the same group who set their attributes as public. Murakami and Watanabe [98] proposed a learning method, which uses matrix factorization to accurately estimate personalized transition matrices from a small amount of training data. They applied the proposed learning method to the localization attack, which identifies the real location of a user at a given time instant from an obfuscated trace.
2.1.4 Attacks for Other Applications

Various attack models are designed according to different application scenario. Except for the aforementioned attacks for tabular data, graph data and location privacy, privacy attacks happen in other applications as well.

For example, smart meters have been wildly used in more and more households. It brings convenience while raising privacy concern. The purpose of attacks in the smart meter is to infer the user’s behaviour and habits by observing the power usage data. Dinesh et al. [29] utilized the uncorrelated spectral information of active power signals with low sampling rate to identify the turned-on residential appliances and estimate their energy consumption, which further discloses the user’s behaviour. Greveler et al. [47] showed that the personal TV watching habits can be inferred by simply analysing the electricity usage profile. And in many cases, the view content also can be identified based on the smart meter power consumption data. Fan et al. [39] proposed a reactive power based attack, which extracts reactive power-based appliance signatures, and identifies the turned-on events of the appliance based on each appliance’s unique signature.

Privacy attack also happens in the recommendation system. The most popular one is the \( k \) nearest neighbour attack. Suppose the attacker knows \( m \) ratings of the items of an intended victim, the attacker can then compromise the victim’s privacy by creating \( k - 1 \) fake users with the same ratings to replicate their recommendation list. In addition, Van [145] proposed an attack on contactless payments. Liu and Li [87] identified a collusion privacy attack against online friend search engine.
2.2 Privacy Technologies

Various privacy technologies have been proposed in the past few decades. This section mainly surveys three categories: the encryption-based methods, anonymization methods, and differential privacy related methods. We also discussed other new developed privacy technologies in the last section.

2.2.1 Encryption

Encryption is a direct method to protect the user’s data. Nobody can know the other user’s data without description. Therefore, the user’s data can be stored safely in the application server or the cloud. However, the encrypted data cannot contribute to the various services and meaningless to the researchers.

One class of approaches is to release these encrypted data is that adding access control protocol. For example, Qian et al. [117] proposed the personal health record (PHR) should be encrypted before outsourcing it to the cloud, and the patent should be able to decide who can access what kind of PHR. Only the users with the corresponding secret keys can access the corresponding encrypted PHRs. Ruj et al. [124] proposed a decentralized access control scheme for secure data storage in the cloud, and the proposed scheme supports anonymous authentication that the cloud verifies the authenticity of the user without knowing his identity.

Another class of methods is utilising some encryption technologies that allows performance over encrypted data. There are mainly two technologies: secure multiparty computing and homomorphic encryption.
Secure Multi-party Computing. Secure multi-party computing was originally introduced by Yao [166], and it allows different parties to jointly collaborate to compute a specific function on their private data, while preserving the data privacy of each party. The secure multi-party computing ensures the independence of the input and the correctness of the calculation without disclosing the input value to other participants in the calculation. Secure multi-party computation has many applications in the area where people need to operate numbers which are sensitive and private.

Tso et al. [144] introduced a method to prevent data disclosure from inside attack in wireless medical sensor networks. The proposed method is based on the secure multi-party computation protocol and has been implemented in the FairplayMp framework. Also, the proposed approach supports private-preserving data mining for medical data stored in the distributed hospitals or institutions. Wang et al. [154] proposed a privacy-preserving protocol to estimate the edit distance between two genome sequence. The secure multi-party computation technology is used to compute the set intersection. Secure multi-party is also used in Point of Interest recommendation [153] that provide the users with accurate PoIs without disclosing their real location information.

Secure multi-party computing is designed for few sides security calculations, and it is not easy to expand to a collaborative environment that has many users, because the cost of calculation increases exponentially. Therefore, secure multi-party computing is not widely used for deep learning techniques.

Homomorphic encryption. Homomorphic encryption was proposed by Rivest et al. [123] in 1987. It is a way of encryption method that allows the computation over ciphertexts. The calculation result over ciphertexts, when decrypted, matches
the result of the operation performed on the plaintext. The purpose of homomorphic encryption is to allow the certain operations can be performed on the encrypted data, which preserves the original data information.

Homomorphic encryption is widely used in various areas. For example, Alabdulatif et al. [7] applied it to the data clustering tasks. They introduced a distributed data processing framework, which is based on a fully homomorphic. The proposed framework utilizes MapReduce to perform distributed computations on a large number of cloud Virtual Machines. Abdallah and Shen [4] proposed a lightweight lattice-based homomorphic privacy-preserving data aggregation scheme for the residential electricity consumers in the smart grid. Kim et al. [73] studied the privacy problem in matrix factorization-based recommendation system, and proposed the first privacy-preserving matrix factorization using fully homomorphic encryption. Homomorphic encryption is also used for privacy preserving anomaly detection [7], outsourced calculation of rational numbers [86], and location-based services [38]. However, due to the large computational overhead, most of them are still far from being ready to use in practical applications.

### 2.2.2 Anonymization

Data anonymization is a very important method for preserving data. It’s a type of information sanitization that removes the personally identifiable information, so that the person cannot be identified from the data or the sensitive attributes can not be matched to the specific person.
**k-anonymity.**  

$k$-anonymity [125] is the most popular anonymity technology, which was first introduced by Sweeney and Samarati in 1998. We say the release data satisfy $k$-anonymity if the information for each person cannot be distinguished from at least $k - 1$ other individuals in the same dataset. In other word, the personal information is hidden in a group that each individual in the group has the same information with others and cannot be distinguished. The group is referred to the equivalence class.

$k$-anonymity can be used in various application domains, which including location-based service [80,105], clustering [85], Internet of Things [14] and so on. For location-based services, multiple ways can be used to achieve $k$-anonymity. For example, Niu et al. [105] proposed a dummy-location selection algorithm to protect user’s location privacy against adversaries with side information. In the dummy locations methods, $k - 1$ false locations are reported together with the user’s real location to guarantee $k$-anonymity. Song et al. [133] protected the private location by applying cloaking region technology that satisfies $k$-anonymity by including at $k$ users’ locations in the cloaked region.

$k$-anonymity is easy to achieve and widely used, however, it only considers the identity without considering the sensitive attributes. If all the sensitive value in the equivalence class are the same, the user’s sensitive information will be disclosed.

**l-diversity.**  

$l$-diversity solves the weakness the $k$-anonymity by requiring the equivalence class has at least $l$ different sensitive values. That is, $l$-diversity protect the user’s sensitive information by increasing the diversity of the sensitivity value in the equivalence class.

A stronger notation of $l$-diverse is the definition of entropy $l$-diversity, which is defined as follow:
Definition 1 (Entropy $l$-diverse). An equivalence class is entropy $l$-diverse if the entropy of the distribution of the sensitive value is at least $\log(l)$.

$$\sum_{s \in S} P(QID, s) \log(P(QID, s)) \geq \log(l), \quad (2.2.1)$$

where $s$ indicate the sensitive value, and the $P(QID, s)$ indicate the proportion of $s$ in equivalence class.

Though the $l$-diversity considers the diversity of sensitive values in the equivalence class, it does not consider the distribution of the sensitive values. Though the released data satisfy the $l$-diversity, if one sensitive value accounts for a large proportion in the equivalence class, the user’s information will be easily disclosed by skewness attack.

t-closeness. t-closeness [77] improves the $l$-diversity by requiring the distribution of the sensitive value in the equivalence class close to the corresponding distribution in the original dataset. That is, the distance between the distribution of the sensitive value in the original dataset and equivalence class is bounded by $t$.

Many other extensions are also be proposed, for example, $(\alpha, k)$-anonymity [150], $(k, e)$-anonymity [172] and $m$-invariance [158]. Most of these methods add the different constraint on the sensitive values in the equivalence class, which enhanced the privacy protection. However, the anonymization method does not consider the attacker’s background knowledge. User’s sensitive information and even the identity can easily be disclosed once the attacker holds the corresponding background knowledge.

2.2.3 Differential Privacy

differential privacy. Differential privacy is a provable privacy notation, developed by Dwork et al. [35] that has emerged as an essential standard for preserving privacy
in a variety of areas. It was originally introduced for statistical databases, requiring that changes to a single data record should result in no statistical differences to a query’s output. The formal definition of differential privacy is:

*Definition 2 (ε-Differential Privacy).* A randomized algorithm $\mathcal{M}$ gives $\epsilon$-differential privacy for any pair of neighboring datasets $D$ and $D'$ where, for every set of outcomes $\Omega$, $\mathcal{M}$ satisfies

$$
Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \Omega].
$$

Datasets $D$ and $D'$ are neighbouring datasets, which only differ in one individual record. This definition ensures that the presence or absence of an individual will not significantly affect the output of the query.

Normally the differential privacy can be achieved by adding random noise to the results of the query. *Laplace* mechanism and *Exponential* mechanism are two mechanisms that usually used. The definitions are shown as follows.

*Definition 3 (Laplace mechanism).* Given a function $f : D \to \mathbb{R}$ over a dataset $D$, 3 provides $\epsilon$-differential privacy.

$$
\hat{f}(D) = f(D) + \text{Laplace}(\frac{s}{\epsilon}).
$$

*Definition 4 (Exponential mechanism).* Let $q(D, r)$ be a score function of the dataset $D$ that measures the score of outputting $r$. Then the exponential mechanism $\mathcal{M}$ satisfy $\epsilon$-differential privacy if:

$$
\mathcal{M}(D) = \text{return } \propto \exp\left(\frac{eq(D, r)}{2\Delta q}\right),
$$

where $\Delta q = \max \| q(D, r) - q(D', r) \|$, which is the sensitivity of $q$. 

22
A Laplace mechanism is used for numeric output, and a Exponential mechanism is used for non-numeric output. Both mechanisms are associated with the sensitivity, which determines how much perturbation is needed. Two types of sensitivity are usually used in differential privacy: the global sensitivity [136], which measures the maximal change over all the neighbour datasets, and the local sensitivity [104], which measures the change over the records related to the query. The details are shown as follows.

**Definition 5 (Global sensitivity).** For a query $Q : D \rightarrow \mathbb{R}$, the global sensitivity of $Q$ is defined as follow:

$$GS = \max_{D,D'} \|Q(D) - Q(D')\|_1.$$  \hspace{1cm} (2.2.5)

**Definition 6 (Local sensitivity).** For a query $Q : D \rightarrow \mathbb{R}$, the global sensitivity of $Q$ is defined as follow:

$$LS = \max_{D'} \|Q(D) - Q(D')\|_1.$$  \hspace{1cm} (2.2.6)

Observe that the global sensitivity from Definition 5 is $GS = \max_x LS$. The global sensitivity is widely used, but it only considers the worst-case that induces much redundant noise to the output. Local sensitivity avoids the unnecessary noise by only considering the query related records.

Two privacy budget compositions [94] are widely used in the design of mechanisms [63]: the sequential composition and the parallel composition, as defined in Definition 7 and Definition 8, respectively.

**Definition 7. Sequential Composition:** Suppose a method $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ has $m$ steps, and each step $\mathcal{M}_i$ provides $\epsilon$ privacy guarantee, the sequence of $\mathcal{M}$ will provide $(m \times \epsilon)$-differential privacy.
Definition 8. Parallel Composition: Suppose a method \( \mathcal{M} = \{M_1, ... M_m\} \) has \( m \) steps, and each step \( M_i \) provides \( \epsilon \) privacy guarantee on a disjointed subset of the entire dataset, the parallel of \( \mathcal{M} \) will provide \( \max \epsilon_1, ..., \epsilon_m \)-differential privacy.

The sequential composition measures the privacy level for a sequence of differentially private computations. When a series of randomized mechanisms have been performed sequentially on the same dataset, the total privacy guarantee proposed will be calculated by adding up the privacy budgets for each step. The parallel composition applies to the situation where each \( M_i \) is applied on disjointed subsets of the dataset. The ultimate level of privacy guarantee only depends on the largest privacy budget.

Extensions of Differential Privacy. With the application of the traditional differential privacy, many extensions are proposed to solve more complex problems. We review the proposed new differential privacy mechanisms in this section.

Approximate Differential Privacy [34]. For a given metric on the input space, differential privacy requires that the distance between two neighbouring inputs at most 1, the probability that the outputs of performing the random algorithm differs at most \( \exp(\epsilon) \). Approximate differential privacy relaxes this requirement by additionally allowing for an additive slack \( \delta \). Specifically,

**Definition 9 ((\( \epsilon, \delta \))-Differential Privacy).** A randomized algorithm \( \mathcal{M} \) gives \( (\epsilon, \delta) \)-differential privacy for any pair of neighboring datasets \( D \) and \( D' \) where, for every set of outcomes \( \Omega \), \( \mathcal{M} \) satisfies

\[
Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \Omega] + \delta.
\]  

(2.2.7)

Approximate differential privacy weakens the privacy guarantee but allows data
release results more accurate. Many privacy-preserving methods are based on approximate differential privacy, for example, Du et al. [31] proposed a differential privacy-based query model for sustainable fog computing supported data center. Lin et al. [81] designed two frameworks for privacy-preserving auction-based incentive mechanisms that achieve approximate social cost minimization.

**Distributed Differential Privacy** [34]. Traditional differential privacy considers a trusted aggregator who can access the participants’ real data and release the perturbed statistics. However, the aggregator might be untrusted. To solve this problem, Emura and Keita [37] proposed the concept of distributed differential privacy, which extended the approximate differential privacy to a setting that the distributed entities contribute the perturbed data to the control aggregator. The aggregator can be untrusted and possibly colludes with a subset of the participants.

**Definition 10 ((ε, δ)-Distributed Differential Privacy).** Suppose ϵ > 0, 0 ≤ δ ≤ 1 and 0 < γ ≤ 1. We say that the data randomization procedure, given by the joint distribution \( r := (r_1, \ldots, r_n) \) and the randomization mechanism \( M \) achieves \((ε, δ)\)-distributed differential privacy with respect to the mechanism \( M \) and under γ fraction of uncompromised participants if the following condition holds.

\[
Pr[M(D) ∈ Ω] ≤ \exp(ε) · Pr[M(D') ∈ Ω] + δ .
\] (2.2.8)

Joy [66] believed that the released aggregate information under distributed privacy protection only reveals the underlying ground truth distribution and nothing else, and they proposed a sampling mechanism, which achieves differential privacy in the distributed setting. Distributed differential privacy is also widely used in the situation that answering queries about the privacy data that is spread across multiple different
databases. Narayan and Haeberlen [100] introduced two new primitives, BN-PSI-CA and DCR, that can be used to answer queries over distributed databases with differential privacy guarantees. Zhang et al. [171] addressed the problem of distributed knowledge extraction with differential privacy guarantee in the data mining.

Joint Differential Privacy [34]. Private matching and allocation problems have not been considered in the differential privacy literature. While, Kearns et al. [70] introduced a variant which they call joint differential privacy, which requires that simultaneously for every player $i$, the joint distribution on the suggested actions to all players $j \neq i$ be differentially private in the type of agent $i$.

Definition 11 (Joint Differential Privacy). A mechanism $\mathcal{M}$ satisfies $(\epsilon, \delta)$-joint differential privacy if for every $i$, any pair of $i$-neighbours $D, D'$, and for every subset of outputs $\Omega \subseteq \mathcal{R}^{n-1}$,

$$Pr[\mathcal{M}(D)_{-i} \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D')_{-i} \in \Omega] + \delta . \quad (2.2.9)$$

Later, Hsu et al. [57] gave algorithms to accurately solve the private allocation problem when bidders have gross substitute valuations using joint differential privacy. Tong et al. [143] studied the location privacy problem in the scheduling of ridesharing services. They proposed a jointly differentially private scheduling protocol for protecting riders’ location information and minimizing the total additional vehicle mileage in the ridesharing system.

Geo-Indistinguishability. Andres et.al [11] proposed the concept of Geo-indistinguishability, which extended the traditional differential privacy to the location privacy scenarios. The main idea behind this notion is that, for any radius $r > 0$, the user enjoys $\epsilon r$-privacy within $r$. The level of privacy is proportional to the radius $r$. The details of definition is shown as follow.
Definition 12 (Geo-Indistinguishability). A mechanism $K$ satisfies $\epsilon$-geo-indistinguishability if for all $x, x'$:

$$d_p(K(x), K(x')) \leq \epsilon d(x, x') \quad (2.2.10)$$

For all points $x'$ within a radius $r$ from $x$, the definition forces the corresponding distributions to be at most $\epsilon r$ distant.

Geo-indistinguishability is widely used for protecting the location privacy [132, 156, 161, 169]. For example, Wang et al. [156] solved the location privacy problem in the mobile crowdsensing application. The uploaded locations for optimal task allocation are protected under geo-indistinguishability guarantee. Mao et al. [91] presented a scheme for aggregating the installation ratio for applications with privacy-preserving on mobile devices. Hua et al. [59] proposed an improved geo-indistinguishability location perturbation mechanism for location-based services, which significantly reduced the privacy cost and can support multiple queries.

2.2.4 Other Technologies

**Caching.** Caching system has been proposed to enhance the privacy in various application scenario by pre-fetching the data on a device before it is actually needed [9]. The data can be accessed locally when it is needed, which reduces the interaction between the user and the service providers and reduces the risk of privacy information exposure. Caching method is widely used in location-based services.

Peng et al. [113] proposed a collaborative trajectory privacy preserving scheme for continuous queries, in which trajectory privacy is ensured by caching-aware collaboration between users, and no need for any fully trusted parties. The main idea behind the proposed algorithm is that it allows the mobile user to communicate with
multi-hop neighbours and share the valid information between each other. Users’ collaborative caching reduces the number of queries sent to the server, thereby reducing the amount of private information exposed to the server. Zhang et al. [177] proposed a caching-based method to protect location privacy in continuous location-based services. The proposed scheme adopts a two-level caching mechanism to cache the users’ result data at both the client and the anonymizer sides. Liu et al. [82] proposed a framework which enhances the privacy of location-based services by actively caching in the wireless vehicular network scenario. Niu et al. [106] proposed a privacy metric to model the effect of caching. In addition, the proposed a caching-aware dummy selection algorithm, which is combined with multiple privacy technologies and side information to achieve a higher privacy degree.

**Game Theory.** Game theory is the study of mathematical models of strategic interaction between rational decision-makers [99]. Game theory answers the question of how the defender reacts to the attacker. The strategic interaction between them is captured by a two-player game where each player tries to maximize his or her own interests. The effectiveness of a defence mechanism relies on both of the defender’s and attacker’s strategic behaviours.

Recent years, game theory has been applied in privacy-preserving area. For example, Wu et al. [157] studied the privacy problem in correlated data publishing. They modelled the trade-off problem between privacy and utility as a game problem, and analysed the utility efficiency of the proposed method from the point of game theory. Khaledi et al. [71] investigated attacks where person location can be inferred using the radio characteristics of wireless links and modelled the radio network leakage attack using a Stackelberg game. They used a greedy method to obtain the optimal strategy
for the defender. The experimental results showed that the proposed game theoretic solution significantly reduces the chance of an attacker infer the user’s location.

2.3 Privacy Metrics

The privacy metrics are used to evaluate the privacy level in a system or the privacy protection level provided by the proposed privacy protection method. A large number of metrics have been proposed in the literature. We category the existing privacy metrics according to the output of the algorithms.

2.3.1 Uncertainty

Uncertainty means the situation in which something is not known or not certain. The uncertainty metrics in the privacy area evaluate how far the attacker’s estimation is to the certain correct value. Most uncertainty metrics are built on anonymity parameter and entropy. We review the related matrices as follows.

**Anonymity parameter.** Some literature protects the user’s privacy by hiding an individual or the sensitive attribute in an anonymity set in which the attacker cannot identify the correct one. The most popular one is the $k$-anonymity. The size of the anonymity set is $k$, the user’s information cannot be identified to the other $k - 1$ users. Therefore, the privacy level can be measured by the size of the anonymity set as

\[ PM = k; \] (2.3.1)

Similar to $k$-anonymity, the extensions of $k$-anonymity try to protect the sensitive attributes by adding some qualifications. The qualification can be used to evaluate
the privacy-preserving level.

For example, \((\alpha, k)\)-anonymity [150] requires the frequency of the single sensitive value in the equivalence class has to be less than \(\alpha\). Therefore, the privacy level can be evaluated by

\[
PM = (\alpha, k)
\]  \hspace{1cm} (2.3.2)

Smaller \(\alpha\) indicate higher privacy protection, as the probability of inferring the victim’s sensitive attributes becomes lower.

\((k, e)\)-anonymity [172] additionally requires the range of the attributes in the equivalence class must be greater than \(e\). Bigger \(e\) and \(k\) indicate higher privacy level. Therefore,

\[
PM = (k, e)
\]  \hspace{1cm} (2.3.3)

\(l\)-diversity bounds the diversity of sensitive information. Similar with \((k, e)\)-anonymity, \(l\)-diversity requires \(l\) distinct values in each equivalence class. Therefore, the privacy level can be evaluated by

\[
PM = l;
\]  \hspace{1cm} (2.3.4)

The main weakness of this metric is that it only counts the number of records in the anonymity set, doesn’t consider the attacker’s background knowledge. If the attacker holds some background knowledge, it cannot provide the corresponding level of privacy protected measured by the anonymity set size.

**Entropy.** The entropy in the information theory refers to disorder or uncertainty. When a lower-probability event occurs, the event carries more information than the higher-probability event happens. As a privacy metric, the entropy measures the uncertainty associated with inferring the sensitive value of an individual.
Shannon entropy [127] is the basis for many other metrics. The Shannon entropy is defined as follow.

\[
H(X) = - \sum_{x_i \in X} p(x_i) \log_2 p(x_i),
\]

(2.3.5)

where \( x_i \) is the value in the set of all possible values, \( p(x_i) \) is the probability of the value \( p(x_i) \) to be the target.

Max-entropy [25] is the upper bound of the Shannon entropy. It is a conservative measure of how certain the adversary is of his estimate. Specifically,

\[
H_{\text{max}} = \log_2 |X|
\]

(2.3.6)

The max-entropy only depend on the size of the variable values set, which represents the ideal privacy situation for the user.

Min-entropy is the lower bound of Shannon entropy, which is the worst-case scenario because it only depends on the user for whom the adversary has the highest probability regardless of whether this is also the true outcome [26]. Specifically,

\[
H_{\text{min}} = -\log_2 \max_{x \in X} p(x)
\]

(2.3.7)

Silvia et al. [116] used Shannon’s entropy as the measure of user privacy in social tagging systems. Javier et al. [111] investigated mathematically the privacy utility trade-off posed by the suppression of tags, measuring privacy as Shannons entropy of the perturbed profile. Peters and Maxemchuk [114] used Shannon entropy to compare the performance of the distributed application to a centralized one regarding storing and processing electronic health records. In [51] and [131], min-entropy is considered rather than the usual Shannon entropy.

Rényi, Alfréd [122] introduced Rényi entropy, which is a more general metric that
based on Shannon entropy. The formula is shown as follow.

\[ H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \sum_{x \in X} p(x)^\alpha \quad (2.3.8) \]

The Shannon entropy is the special case when \( \alpha \to 1 \). Also, the more \( \alpha \) grows the more Rényi entropy approaches to min-entropy, and the more \( \alpha \) reduces to zero, the more Rényi entropy approaches to max-entropy.

*Conditional entropy* is used to measure the attacker’s uncertainty about the user’s real sensitive information when the prior knowledge is known. The uncertainty of the attacker regarding the value of \( x \) given \( z \) can be measures as the entropy of the posterior as follow:

\[ H(x|z) = -\sum_{x \in X} p(x|z) \log(p(x|z)) \quad (2.3.9) \]

Oya et al. [108] utilized the conditional entropy to justify the defences created by the proposed strategies for location privacy-preserving. Wang and Jia [147] used it to quantify the degree of disclosure risk in a medical data publishing. Osia et al. [107] verified and evaluated the privacy of the features extracted by the private feature extractor using conditional entropy.

There are also many other entropy-based metrics, such as *unreliability* that measures the attacker’s uncertainty about which items are related, *cross-entropy* that measures the uncertainty in predicting the original dataset from the clustered model and *Kullback-Leibler (KL) divergence* [119] evaluates the distance between two distribution. Readers can refer to paper [146] for more entropy-based metrics.

Although entropy has an intuitive interpretation of the information that the attacker needs, the value of entropy does not convey many meaning [51]. The entropy indicates the attacker’s uncertainty, but does not state how accurate the attacker’s
estimation is [131]. In addition, many works of literature argue that the entropy is strongly influenced by outlier values as the privacy metric (i.e., if some element of the probability distribution is very unlikely, the entropy of the source increases very much [26]).

### 2.3.2 Error/Accuracy

**Error.** The error metrics quantify the error the attacker makes to infer the user the real information. This type of metrics are applicable to all domains and widely used in recommendation system and location privacy.

Mean square error (MSE) describes the error between observations $x$ by the attacker and the true output $x'$ of the query.

\[
MAE = \frac{1}{|X|} \sum_{x \in X} \| x - x' \|^2
\]  

(2.3.10)

Meng et al. [95] used MSE to evaluate the privacy protection level by conducting reconstruction attack on the rating dataset. MSE is also used in reconstructing user data in participatory sensing [43]. In addition, the MSE is widely used as the utility metric in the literature as well. For example, Tan et al. [139] used it to evaluate the image quality and Wang et al. [152] used MSE to measure the utility loss in anonymized mobile context streams.

Similar metrics, such as root mean square error (RMSE) [20] and mean absolute error (MAE) [20], are also used to measure the accuracy. For example, Polatidis et al. [115] used both RMSE and MAE to measure the accuracy of generated recommendations of the proposed protection method.

Expected estimation error [131] measures the expected distance between the real value and estimated value. The estimation is computed over the posterior probability
of the attacker’s estimation $x$ on his or her observation $y$. Specifically,

$$\text{Error} = \sum_{x \in X} p(x \mid y)d(x, x'),$$  \hspace{1cm} (2.3.11)

where $d(\cdot)$ is the distance metric. In location privacy, $d(\cdot)$ measures the distance between the estimated location and the true location, $p(x \mid y)$ means the probability that the attacker infers the location is $x$. The bigger the value error, the stronger the protection of the user’s location.

Hoh and Gruteser [55] proposed a similar distance error, called expectation of distance error, which captures how accurate the attacker can estimate the user’s position. The metric is defined as follow:

$$\text{Error} = \frac{1}{nT} \sum_{t \in T} \sum_{h \in \mathcal{H}} p_{h,t}(x)d_{h,t}(x, x'),$$  \hspace{1cm} (2.3.12)

where $n$ is the number of users and $T$ is the total observation time. $d_{h,t}(x, x')$ indicates the distance error between the real location and the location in hypothesis space $\mathcal{H}$ at time step $t$.

These distance-based privacy metrics can also be used in other application scenarios if an appropriate distance metric is available. For example, the distance metric used in paper [61] depends on how the values of genetic variations are encoded.

Percentage incorrectly classified measures the percentage of incorrectly classified users or events. This metric is wildly used in machine learning area for classification application.

$$\text{Error} = \frac{|I'|}{|I|},$$  \hspace{1cm} (2.3.13)

where $I'$ indicates the incorrect classification. $|I|$ is the size of the instance set.

Kalyani and Aniket [68] studies the privacy preserving data mining problem and
evaluate the performance of the proposed method by analysing the incorrectly classified instance. Narayanan and Shmatikov [101] used it to measure how often the highest probability in the attacker’s estimate does not correspond to true genotype.

**Accuracy.** Accuracy metrics quantify the accuracy of the attacker’s estimation. The inaccurate estimate indicates higher privacy protection. Most metrics in this category are used to measure the geographic precision in location-based services.

*Size of Uncertainty Region* [22] represents the minimum region that the attacker can narrow down to locate the target user’s location. This metric is used for the cloaking region based location privacy-preserving method. The user’s location is hiding in a cloaking region, the bigger size of the final region for the attacker, the higher level of protection provided.

\[
PM = \text{Area}(R),
\]

where \( R \) refers to the cloaking region.

*Coverage of Sensitive Region* is another expression form, which is shown as follow.

\[
PM = \text{Area}\left(\frac{R_s \cap R}{R}\right),
\]

where \( R_s \) is the sensitive region for a user and \( R \) is the attacker’s uncertainty region. Eq. 2.3.15 indicates the proportion that the attacker can link the user’s location to his sensitive region.

*Confidence Interval Width* [6] is a type of interval estimate, computed from the statistics of the observed data, that might contain the true value of an unknown variable. The privacy at confidence coefficient \( \gamma \) is given by the width of confidence region \([u(X), v(X)]\) for the attacker’s estimation.

\[
PM = |u(X) - v(X)|, p(u(X) \leq x \leq v(X)) = \gamma/100
\]
The confidence level is designated prior to examining the data. Most commonly, the 95% confidence level is used.

2.3.3 Indistinguishability

Indistinguishability metrics indicate whether the attacker can distinguish the real sensitive value of the user from the other values. Most of these metrics are associated with differential privacy mechanism.

Differential privacy states that the outputs of the random mechanism performed on the two neighbour datasets differ by at most $e^\epsilon$. That is to say, the value of a variable to be protected cannot be distinguished with other values under differential privacy protection. The $\epsilon$ measures the biggest difference between the true value and the other values. The smaller value of $\epsilon$ indicates higher privacy protection.

Most extensions of differential privacy are based on the approximate differential privacy that introduces a slack parameter $\delta$ relaxes the requirement of traditional differential privacy. For example, distributed differential privacy and joint differential privacy. The smaller $\epsilon$ and $\delta$, the more difficult for the attacker to identify the real value of the variable. When $\delta = 0$, the method satisfies pure differential privacy.

Geo-indistinguishability is also an extension of the traditional differential privacy, and it is applied to the location privacy scenarios. The privacy level depends on the privacy budget $\epsilon$ and the radius $r$. The user’s location is protected under geo-indistinguishability guarantee means the user’s true location cannot be distinguished with other locations in the region with radius $r$ by the attacker.
2.4 Summary

This chapter surveys various attacks happened on different types of dataset and different application scenarios, the privacy technologies and privacy metrics that how to evaluate the privacy level based on different privacy-preserving methods. This thesis studies the privacy problems in four application scenarios: recommendation system, Internet of Things, location-based services and crowdsensing system. We identify the privacy problems first, and propose the correspondence solutions based on existing privacy-preserving technologies according to the possible attacks. The analysis based on the privacy metrics proved the effectiveness of the proposed solutions.
Chapter 3

Privacy-preserving in Recommendation System

3.1 Introduction

The boom in web-based services has caused an exponential increase in the number of users of these services, resulting in a significant growth in the amount of online information. This information growth has triggered the need for personalized recommendations, which are the key business drivers for many companies. Research shows that 75% of users’ Netflix choices are driven by recommendations [62]. This personalized recommendation is derived from users’ personal data, such as purchase records and browsing history, raising significant concerns about maintaining privacy [120].

Neighbourhood-based collaborative filtering, for example, is a popular recommendation method that is particularly vulnerable to attacks on privacy [181]. The literature shows that continuous observation of recommendations with a certain amount of background information makes it possible for an individual’s transaction history to be inferred, known as a $k$ nearest neighbour (KNN) attack [19]. If attackers know $m$ items of user $u$, they can create $k$ fake users with the same rating as victim $u$ on these
m items. Such attackers can obtain the rating history of user $u$ by directly searching the list of recommended items directly. Other recommender methods, such as matrix factorization, also suffer from privacy breach problems [15,41].

It is highly desirable to devise a solution that will guarantee user privacy. Differential privacy, a powerful privacy model, is widely accepted for providing rigorous privacy guarantees for aggregate data analysis. Differential privacy ensures that one individual cannot significantly affect the output of a query. This is normally achieved by injecting random noise, calibrated according to the sensitivity to the result of the query. This process was first introduced to collaborative filtering by McSherry and Mironov [93] in 2009. Shen et al. [129,130] and Zhu et al. [181] subsequently contributed to the privacy preservation problem in recommendation systems via differential privacy. However, all existing methods guarantee user privacy at the cost of analytical utility, since accuracy is decreased as a result of the privacy operation. Privacy and utility are inherently conflicting concepts in the literature. Can we find a solution that defuses this conflict, while supporting the same goals?

Introducing randomization improves performance in some research areas [24,42], and we have also identified randomization as the major mechanism for achieving differential privacy. Is it possible to adopt a particular randomization method in the collaborative filter, not only to guarantee user privacy, but also to enhance the accuracy of prediction?

There are two challenges to achieve this goal:

1. **Selecting randomization method**: Neighbours are of the utmost importance for utility in neighbourhood-based collaborative filtering. However, randomized methods can destroy the distance between users, reducing the quality of the
neighbours selected, and decreasing the accuracy of the prediction. How can we retain the distance between users? What kind of randomization should be chosen?

2. Achieving differential privacy: Unlike traditional differential privacy methods, which add noise to the result of the query, randomized methods directly perturb the original dataset. How can we make sure this randomization achieves differential privacy objectives?

For the first challenge, we observe that the Johnson-Lindenstrauss-transform is an excellent solution for retaining the utility. It is a linear transformation that preserves the distance property, which is important for neighbourhood-based collaborative filtering. For the second challenge, Jeremias et al. [16] prove that the Johnson-Lindenstrauss transform preserves edge differential privacy in graph sanitization. Based on this finding, we prove that it also guarantees differential privacy in a rating dataset. The extensive experiments show that this transform not only maintains distance, but also enhances the accuracy of the final prediction.

The contributions in this chapter are as follows:

- We propose a privacy preserving collaborative filtering method that guarantees differential privacy without compromising prediction accuracy.

- We theoretically prove that the proposed method satisfies $\epsilon$-differential privacy, where $\epsilon$ is related to the dataset. In addition, we theoretically analyse the utility of the proposed method.

The rest of the chapter is organized as follows. In Section 3.2, we discuss the related work and Section 3.3 introduce the preliminaries. We propose our privacy
preservation method and theoretically analyse the privacy and utility in Section 3.4 and Section 3.5 respectively. Section 3.6 details the results of the experiments, and Section 3.7 concludes the chapter.

### 3.2 Related Work

Notation for differential privacy was first proposed by Dwork [32] in 2006. Since it provides rigorous privacy preservation and can be proven by mathematical theory, differential privacy has become an important standard for evaluating privacy levels and been applied in several areas. It was first introduced to collaborative filtering in 2009 by McSherry and Mironov [93], who adapted four leading collaborative filtering algorithms to differential privacy. User privacy was guaranteed by directly adding noise to the aggregated query results.

Zhu et al. [181] proposed an effective privacy preserving algorithm for neighbourhood-based collaborative filtering. They held that it is not only the ratings of the user that should be protected, but also 'who the neighbour is' should not be inferred by adversaries. The Exponential mechanism was adopted to find $k$-nearest neighbours and Laplace noise was added to mask the rating given by a certain neighbour with recommendation-aware sensitivity.

Rachid et al. [49] proposed a distance-based differential privacy to preserve user profiles in a recommendation system. They replaced each item probabilistically with a random or related item within a certain distance. The distance between any two items is calculated on the basis of similarity. Defining an upper bound, there is a high probability that each item will be replaced by an item with a distance of less than the upper bound and a low probability that the replacement will be a random
item. This method is not suitable for sparse datasets since there is insufficient data to identify truly similar items; The replacement items would be too distant in the original file, which can reduce utility.

Shen et al. [130] designed a practical privacy framework for personalized recommendation via differential privacy. They consider service providers to be untrustworthy and perturb the users’ data before it leaves the device. The proposed method is based on the Laplace mechanism and guarantees user privacy by injecting calibrated Laplace noise into each item category. Noise calibration is based on the underlying data properties through correlation between categories. Good performance is achieved as a result of minimizing noise.

The Stretching mechanism proposed by Mohammad et al. [8] is much closer to our method. The original dataset was mapped onto another with different direction by multiplying a shrinkage matrix whose diagonal coefficients are a user’s privacy preferences. Laplace noise is added to query outcomes with a fixed privacy parameter.

The works described above provide a rigorous privacy guarantee for users, but all existing methods have to sacrifice elements of utility in varying proportions to preserve user privacy. Unlike these methods, our method not only guarantees user privacy, but also improves performance.

3.3 Preliminaries

3.3.1 Notation

The dataset $D$ is a $user \times item$ numerical rating dataset. Let $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ be the set of users and $\mathcal{T} = \{t_1, t_2, ..., t_d\}$ be the set of items. The dataset $D$ can be
represented as a \( n \times d \) matrix, where \( n \) is the number of users and \( d \) is the number of items. We put this in the form of a vector, then \( D = [u_1, u_2, ..., u_n]^T \), where \( u_i = [r_{u_i1}, r_{u_i2}, ..., r_{u_id}] \). \( r_{u_it} \) is the rating that user \( u_i \) gives to item \( t \), so that vector \( u_i \) represent user \( u_i \)'s rating records for all items. \( \mathcal{T}_{co} = \{ t \in \mathcal{T} | r_{it} \neq 0, r_{jt} \neq 0 \} \) is the set of co-rated items by user \( u_i \) and \( u_j \). We use \( s(u_i, u_j) \) to denote the similarity between user \( u_i \) and \( u_j \) and \( N_k(u_i) \) to denote the set of \( u_i \)'s \( k \) nearest neighbours. More notations are shown in Table 3.1.

3.3.2 Collaborative Filtering

Collaborative filtering predicts user interests on specific items by collecting preference information from users closely share their interests. Collaborative filtering methods can be divided into two categories: neighbourhood-based methods and model-based methods. In this chapter, we consider the user-based nearest neighbour algorithm; two stages are involved:

- **Neighbour Selection:** All pairs of users’ similarity are calculated. Choose the \( k \) users with the highest similarity as the closest neighbours.

- **Rating Prediction:** Uses ratings from the \( k \)-closest neighbours selected in Step 1 to calculated possible ratings that the predicted user may give to the item.

Many measurement metrics have been proposed to estimate the similarity between two users. We use two popular metrics, Cosine-based Similarity and Pearson’s Correlation Coefficient. The cosine similarity between two users is defined as follows:

\[
COS_{\text{sim}}(u_i, u_j) = \frac{u_i \cdot u_j}{\|u_i\| \|u_j\|}
\]  

(3.3.1)
Table 3.1: Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of users</td>
</tr>
<tr>
<td>$d$</td>
<td>Number of items</td>
</tr>
<tr>
<td>$D$</td>
<td>Original dataset</td>
</tr>
<tr>
<td>$D'$</td>
<td>Neighbouring dataset</td>
</tr>
<tr>
<td>$u_i$</td>
<td>Users in dataset $D$</td>
</tr>
<tr>
<td>$u'_i$</td>
<td>Users in dataset $D'$</td>
</tr>
<tr>
<td>$s(u_i, u_j)$</td>
<td>Similarity between $u_i$ and $u_j$</td>
</tr>
<tr>
<td>$N_k(u_i)$</td>
<td>Neighbour set of user $u_i$</td>
</tr>
<tr>
<td>$r_{u,t}$</td>
<td>Ratings that user $u_i$ gives to item $t$</td>
</tr>
<tr>
<td>$r_{max}$</td>
<td>The maximum rating</td>
</tr>
<tr>
<td>$T_{co}$</td>
<td>Set of co-rated items</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of neighbours</td>
</tr>
<tr>
<td>$X$</td>
<td>Transfer matrix</td>
</tr>
<tr>
<td>$m$</td>
<td>Dimension of transfer matrix</td>
</tr>
<tr>
<td>$X_c$</td>
<td>Column of matrix $X$</td>
</tr>
<tr>
<td>$X_{ci}$</td>
<td>Entries of matrix $X_c$</td>
</tr>
</tbody>
</table>
The Pearson’s Correlation Coefficient is defined as follows:

\[
PCC_{\text{sim}}(u_i, u_j) = \frac{(u_a - \bar{u}_a)(u_b - \bar{u}_b)^T}{\| u_a - \bar{u}_a \| \| u_b - \bar{u}_b \|} \quad (3.3.2)
\]

where \( u_a = \bigcup_{t \in T_{\text{co}}} r_{at} \). The ratings of co-rated items are used to estimate the similarity. Items that are not rated by either user are not considered.

In step two, all the ratings for item \( t_j \) from users in \( N_k(u_i) \) contribute to the prediction. The specific calculation method is as follows:

\[
r_{u_i t} = \frac{\sum_{u_j \in N_k(u_i)} s(u_i, u_j)r_{u_j t}}{\sum_{u_j \in N_k(u_i)} r_{u_j t}} \quad (3.3.3)
\]

### 3.3.3 Differential Privacy

Notation for differential privacy was first proposed by Dwork [32] in 2006. Since it provides rigorous privacy preservation and can be proven by mathematical theory, differential privacy has become an important standard for evaluating privacy levels and been applied in several areas. Consider the input is privacy dataset \( D \), which is a collection of data from different individuals. We say that datasets \( D \) and \( D' \) are neighbouring datasets if they only differ in one individual record. Differential privacy provides a strong privacy guarantee that the outputs of the queries on the neighbouring datasets will be statistically similar. The formal definition of differential privacy is presented as follows:

**Definition 13 (\( \epsilon \)-Differential Privacy).** A randomized algorithm \( \mathcal{M} \) gives \( \epsilon \)-differential privacy for any pair of neighbouring datasets \( D \) and \( D' \), and for every set of outcomes \( \Omega \), \( \mathcal{M} \) satisfies:

\[
Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \Omega] \quad (3.3.4)
\]
\( \epsilon \) is the privacy parameter, which is defined as the privacy budget [33]. It controls the privacy preservation level. A smaller \( \epsilon \) represents the greater privacy.

**Definition 14 (Neighbouring Dataset).** The datasets \( D \) and \( D' \) are neighbouring datasets if and only if they differ in one record. This is denoted as \( D \oplus D' = 1 \).

'\( \oplus \)' indicates the difference between two datasets. To make the problem much clearer and easier to understand, we assume the two corresponding records in \( D \) and \( D' \) have different \( t \) ratings.

### 3.3.4 Johnson-Lindenstrauss Transform

The Johnson-Lindenstrauss transform projects the points in a high dimension to a lower dimensional space while the Euclidean distances between any pair of points are preserved [65]. The Johnson-Lindenstrauss Lemma has been used in many areas, such as dimensionality reduction, compressed sensing and graph embedding.

**Lemma 3.3.1 (Johnson-Lindenstrauss Lemma [65]).** For any set \( S \) of \( n \) points in \( \mathbb{R}^d \), given \( 0 \leq \delta \leq 1/2 \), \( m = \Omega(\log(n)/\delta^2) \), there is a matrix \( X \in \mathbb{R}^{m \times d} \), for all \( u, v \in S \), there is a map \( f : \mathbb{R}^d \to \mathbb{R}^m \), such that

\[
(1 - \delta)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \delta)\|u - v\|^2 \tag{3.3.5}
\]

in which, \( f(u) = Xu, f(v) = Xv \).

The Johnson-Lindenstrauss Lemma states that a random matrix \( X \) can project the dataset from \( d \) dimension to \( k \) while maintaining the relative distance and retaining \( f(d_{u,v})/d_{u,v} \) in the range \((1-\delta, 1+\delta)\). Several constructions for \( X \) have been proposed [5,63]. We use Gaussian distribution to generate random matrix \( X \).
Gaussian Distribution

Gaussian distribution is also called normal distribution. It’s probability density is as follow:

\[ P_X(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]  

(3.3.6)

Normal distribution is often denoted by \( N(\mu, \sigma^2) \), where \( \mu \) is mean or expectation of the distribution and \( \sigma^2 \) is variance. When a random variable \( X \) is distributed with mean \( \mu \) and variance \( \sigma^2 \), we write \( X \sim N(\mu, \sigma^2) \).

**Theorem 1.** If we take two random variables \( N_1 \sim N(\mu_1, \sigma_1^2), \ N_2 \sim N(\mu_2, \sigma_2^2) \), then the linear combination \( N = aN_1 + bN_2 \) follows Gaussian distribution \( N(a\mu_1 + b\mu_2, a^2\sigma_1^2 + b^2\sigma_2^2) \).

**Theorem 2.** For any matrix \( X \in \mathbb{R}^{d \times n}, x \in X \) suppose the random matrix \( A \in \mathbb{R}^{k \times d} \) and \( A \sim N(\mu, \sigma^2) \), we have \( E(\|Ax\|^2) = \|x\|^2 \).

**Proof.** Suppose the entries of \( A \) are \( N(0, 1/\tau) \),

\[
(\|Ax\|^1)^2 = (\sum_{i=1}^d A_i x_i)^2
\]

by the theorem, we get

\[
\sum_{i=1}^d A_i x_i \sim N(0, \sum_{i=1}^d \frac{x_i^2}{\tau})
\]

so,

\[
E[(\sum_{i=1}^d A_i x_i)^2] = \sum_{i=1}^d \frac{x_i^2}{\tau} = \frac{\|x\|^2}{\tau}
\]
then,

\[ E[\|Ax\|^2] = \tau E[(\|Ax\|)^2] = \tau \times \frac{\|x\|^2}{\tau} = \|x\|^2 \]

\[ \Box \]

**Theorem 3.** If the random matrix \( A \sim N(\mu, \sigma^2) \), then, for any fixed matrix \( X, x \in X \), it satisfies Johnson-Lindenstrauss Lemma with a very high probability.

**Proof.** Suppose the entries of \( A \) are \( N(0, 1/k) \),

For the upper bound:

\[ \Pr[\|Ax\|^2 > (1 + \varepsilon)\|x\|^2] = \Pr[\sum_{i=1}^{k} M_i^2 > (1 + \varepsilon)\|x\|^2, M \sim N(0, \frac{\|x\|^2}{k})] = \Pr[\sum_{i=1}^{k} N_i^2 > (1 + \varepsilon)k, N \sim N(0, 1)] \]

by Lemma,

\[ \Pr[\sum_{i=1}^{k} N_i^2 > (1 + \varepsilon)k] \leq e^{-k(\varepsilon^2 - \varepsilon^3)} \]

\[ = e^{-O((1-\varepsilon)\log(n))} \]
which means:

\[ Pr[\|Ax\|^2 > (1 + \varepsilon)\|x\|^2] \leq e^{-O((1-\varepsilon)\log(n))} \]

We can see that the probability \( Pr[\|Ax\|^2 > (1 + \varepsilon)\|x\|^2] \) is very small, similar proof to the lower bound.

3.4 Privacy-preserving Collaborative Filtering

In this section, we present our solution for the privacy preservation issue in neighbourhood-based collaborative filtering. Defending against KNN attack. We first define the problem and describe the specific algorithm used to solve this problem, then we provide the theoretical analysis of how our method guarantees users’ privacy, while maintaining utility for recommendation purposes.

3.4.1 Problem Definition

Given a rating dataset \( D \), each record is a user’s item rating vector \( U_i \). The objective is to perturb the dataset and predict user \( u_i \)’s interest in items that \( u_i \) has not rated, such that (a) the perturbation satisfies \( \varepsilon \)--differential privacy, which strictly preserves the users’ privacy information and prevent KNN attack; and (b) the similarity calculated on the perturbed dataset is not very different to that calculated on the original dataset; that is, the neighbours selected are almost the same as those in the non-privacy operation. This enables greater utility to be retained and more accurate predictions to be made.
3.4.2 Scheme Overview

The data in recommendation systems are stored as user profiles, and item profiles include the ratings. We absorb the users’ ratings on the items and represent them in $user \times item$ matrix form. The proposed method directly perturbs the dataset (matrix) rather than the query result. The following operations are conducted on the perturbed dataset, and the process is shown in Fig. 3.1.

![Diagram of recommendation process](image)

Figure 3.1: The Johnson–Lindenstrauss recommendation process

The key step is data perturbation. We explore a linear transformation method that changes the original dataset while retaining the important elements for recommendation purposes. Details are presented in the next section.

3.4.3 The Algorithm of Private Neighbour Collaborative Filtering

Traditional differential privacy methods are achieved by adding random noise to the results of queries. The most commonly used mechanisms are Laplace and Exponent. Select the k-nearest neighbours through Exponent mechanism, make sure it impossible for attackers to figure out who the neighbours are. Add Laplace noise to the similarity between users, make sure the attacker cannot infer the rating given by a specific person. In this section, we introduce a Johnson-Lindenstrauss transform-based
privacy preserving collaborative filtering method (JLCF) that applies a Johnson-Lindenstrauss transform to the privacy dataset to preserve users’ rating history and resist a KNN attack. JLCF preserves users’ privacy by perturbing the original dataset through linear transformation. It acquires a new matrix by multiplying the original dataset to obtain a transition matrix. The similarity calculation and neighbourhood selection operations are reflected in the new matrix; the steps are shown in Algorithm 1.

Algorithm 1 shows the proposed preservation method for neighbourhood-based collaborative filtering. The transition matrix $X$ is constructed in Steps 3 to 7. $X$ is a $d \times m$ matrix and the entries of $X$ are sampled from 0 mean Gaussian distribution, which has been proven to satisfy the construction of $X'$ in the Johnson-Lindenstrauss Lemma. We obtain a new perturbed matrix by multiplying the original dataset with $X$ at Step 8. The new matrix $A$ has the property of the Johnson-Lindenstrauss transform; that is, it maintains the relative distances of the original dataset. The $l_2$ distance between any two users in $A$ is compared with $D$, and the error is fixed in the range of $[1 - \delta, 1 + \delta]$, which guarantees the utility of the new matrix for prediction purposes as the calculation of similarity has a linear relationship with $l_2$ distance. Following transformation, we calculate the user similarity with the other users in Steps 10 to 15, and the top $k$ users are classified as user $u_i$’s $k$-nearest neighbours in Steps 16 and 17. Step 18 predicts the ratings of user $u_i$ on item $t$ based on its neighbours’ rating records. Lastly, the rating prediction is returned.

Neighbours play a crucial role in neighbourhood-based collaborative filtering, because prediction is based entirely on the ratings or purchase history of those neighbours. A KNN attack is able to disclose a users’ identity because the attackers only
Algorithm 1 Johnson Lindenstrass Privacy Preserving Collaborative Filtering (JLCF)

Require: Dataset $D \in \mathbb{R}^{n \times d}$, projected dimension $m \ll d$, number of neighbours $k$

Ensure: $r_{u_i,t}$.

1: $N_k(u_i) \leftarrow 0$
2: $r_{u,t} \leftarrow 0$
3: for $i = 1$ to $d$ do
4:    for $j = 1$ to $m$ do
5:        $X[i,j] \leftarrow \text{normrnd}(1, 1/m, 1, 1);$  
6:    end for
7: end for
8: $A \leftarrow DX;$  
9: for $i = 1$ to $n$ do
10:    for $j = 1$ to $n$ do
11:        if $j \neq i$ then
12:            $S(u_i, u_j).\text{sim} \leftarrow \cos(u'_i, u'_j)/pcc(u'_i, u'_j)$
13:            $S(u_i, u_j).\text{user} \leftarrow u_j$
14:        end if
15:    end for
16:    $S(u_i, u_j) \leftarrow \text{sort}(S)$
17:    $N_k(u_i) \leftarrow u_j \in S(1:k)$
18:    $r_{u_i,t} \leftarrow \frac{\sum_{u_j \in N_k(u_i)} S(u_i, u_j) \cdot \text{sim} \cdot r_{u_j,t}}{\sum_{u_j \in N_k(u_i)} r_{u_j,t}}$
19: end for
20: return $r_{u_i,t}$
need a small amount of background knowledge to target the neighbours of the victim. In the proposed method, the original dataset is perturbed by the transfer matrix. Even though the attackers have the information, they cannot identify the $k$-nearest neighbours of the victim, demonstrating that the proposed method is capable of resisting KNN attack.

### 3.5 Privacy and Utility Analysis

The main privacy operation and core technique of our method is to perturb the original dataset with a transition matrix. In this section, we theoretically analyse how this operation contributes to differential privacy preservation and why it maintains prediction utility.

#### 3.5.1 Privacy Analysis

In this section, we prove that our method satisfies $\epsilon$-differential privacy according to the following definition, where $\epsilon = \frac{1}{2} \frac{r^2_{\max}}{\text{tr}(L)}$.

**Theorem 4.** Algorithm 1 guarantee $\epsilon$-differential privacy.

**Proof.** To prove that Algorithm 1 satisfies $\epsilon$-differential privacy, we need to prove that:

$$Pr[f(D) \in \Omega] \leq \exp(\epsilon) \cdot Pr[f(D') \in \Omega] \quad (3.5.1)$$

We observe that the output of perturbed matrix $A$ is composed of $m$ identically distributed columns. Each column is created by multiplying the original data with a vector $X_c \in \mathbb{R}^d$, whose entries are sampled from iid Gaussian distribution $N(0, \sigma^2)$,
where \( \sigma^2 = 1/m \). Therefore, we prove Theorem 4 by showing that each column of the perturbed matrix satisfies Equation 3.5.1.

Let \( X_c \) denote the entries of vector \( X_c \), where, \( 1 \leq i \leq d \). As \( X_c \sim N(0, \sigma^2) \), therefore, \( X_c \) follows multi-dimensional Gaussian distribution, denote it as \( \{X_c \sim N(0, \Sigma) | 0 \leq i \leq m\} \), where \( \Sigma = \begin{bmatrix} \sigma^2 & \cdots & \sigma^2 \end{bmatrix} \). According to the linear combination property of multi-dimensional Gaussian distribution, the transformed matrix \( A \sim N(0, D\Sigma D^T) \), let \( L = D\Sigma D^T, L' = D'\Sigma D'^T \). We have

\[
\frac{Pr[f(D) \in \Omega]}{Pr[f(D') \in \Omega]} = \frac{PDF_D(x)}{PDF_D'(x)}
= \frac{1}{\sqrt{(2\pi)^{\text{rank}(L)}|L|}} \cdot \exp\left(-\frac{1}{2}x^TL^{-1}x\right)
= \frac{1}{\sqrt{(2\pi)^{\text{rank}(L')}|L'|}} \cdot \exp\left(-\frac{1}{2}x^TL'^{-1}x\right)
= \sqrt{\frac{|L'|}{|L|}} \cdot \exp\left(-\frac{1}{2}(x^TL^{-1}x - x^TL'^{-1}x)\right)
\]

If \( L \) is full rank, the inverse matrix of \( L \) is \( L^{-1} \). However, when \( L \) is not full-rank, there is no inverse matrix for \( L \). Instead, we can use a pseudo-inverse \( L^+ \) to denote the generalization of the inverse matrix. The most widely used is the Moore-Penrose pseudoinverse. It is clear that \( L^{-1} = L^+ \) when \( L \) is full-rank according to the definition of the Moore-Penrose inverse [89]. The pseudo-determinant of \( L \) is \( |L|_+ = |A| \), when \( L \) is full-rank, \( |A|_+ = |A| \). Therefore, we have

\[
\frac{Pr[f(D) \in \Omega]}{Pr[f(D') \in \Omega]} = \sqrt{\frac{|L'|_+}{|L|_+}} \cdot \exp\left(-\frac{1}{2}(x^TL^+x - x^TL'^{+}x)\right)
\]

\( x \in \mathbb{R}^n \) is a column vector with \( n \) dimension. \( L = D\Sigma D^T = \sigma^2 DD^T \). Let \( F = D' - D \), which is a matrix with less than \( t \) non-zero entries on the same row. We
have

\[ x^T L' x = \sigma^2 x^T D' D'^T x \]

\[ = \sigma^2 x^T (D + F)(D + F)^T x \]

\[ = \sigma^2 x^T DD^T x + x^T (FD^T + DF^T + FF^T)x \]

Given both matrix \( F \) and \( D \) are sparse, matrix \( F \) only has \( t \) non-zero entries. Therefore, the matrix \( FD^T \) and \( DF^T \) are both zero matrices with a high probability. Matrix \( FF^T \) only has one non-negative entry on the main diagonal. Therefore, \( x^T FF^T x \geq 0 \). We have

\[ x^T L' x \geq \sigma^2 x^T DD^T x = x^T L x \] \[(3.5.4)\]

The result of 3.5.4 can be rewritten as \( x^T (L' - L)x \geq 0 \), meaning that \( L' - L \) is a positive semi-definite matrix. \( L \) and \( L' \) are the real symmetric matrices and satisfy the definition of a Hermitian matrix. Therefore, we can write \( L' \succeq L \). Roughly, we can deduce that \( L^+ \succeq L'^+ \). Further, \( L^+ - L'^+ \) is positive semi-definite, therefore, \( x^T (L^+ - L'^+)x \geq 0 \). For further details, please refer to the literature [56]. Equation 3.5.3 can be written as follow:

\[ \frac{Pr[f(D) \in \Omega]}{Pr[f(D') \in \Omega]} \leq \sqrt{\frac{|L'_{+}|}{|L_{+}|}} \] \[(3.5.5)\]

As \( L \) is a \( n \times n \) Hermitian matrix, its singular value decomposition is \( L = U \Sigma V^* \). \( U \) and \( V^* \) are \( n \times n \) unitary matrices, and \( \Sigma \) is a \( n \times n \) diagonal matrix with non-negative real numbers on its main diagonal, denoted as \( \sigma_i \), \( 1 \leq i \leq n \). The entries \( \sigma_i \) of \( \Sigma \) are known as single values of \( L \) and listed in descending order. If \( \text{rank}(L) = z \), \( |L_{+}| = \prod_{i=1}^{z} \sigma_i \). Matrix \( L' \) is also a Hermitian matrix. We denote its single values as \( \xi_i \),
1 \leq i \leq n. Therefore, \( \sum_{i=1}^{n} \xi_i = tr(L') \leq tr(L) + tr(Q) \leq \sum_{i=1}^{n} \sigma_i + r_{max}^2 n. \) \( tr(L) \) is the trace of the matrix \( L \). It is defined as the sum of the values on the main diagonal. Denote \( \delta_i = \xi_i - \sigma_i \), \( \sum_{i=1}^{n} \delta_i \leq r_{max}^2 n. \) It holds that

\[
\frac{Pr[f(D) \in \Omega]}{Pr[f(D') \in \Omega]} \leq \sqrt{\Pi_{i=1}^{r} (1 + \delta_i / \sigma_i)}
\leq e^{\frac{1}{2} \sum_{i=1}^{n} \frac{\delta_i}{\sigma_i} = \frac{1}{2} r_{max}^2 n} 
\]

(3.5.6)

Let \( \epsilon = \frac{1}{2} r_{max}^2 n \), and we have \( Pr[f(D) \in \Omega] \leq e^{\epsilon} Pr[f(D') \in \Omega] \) and, therefore, the proposed method satisfies \( \epsilon \)-differential privacy.

\[\square\]

### 3.5.2 Utility Analysis

The utility in this chapter is measured by the accuracy of prediction, which in turn is based on the quality of the neighbours, defined on COS and PCC. We use COS as an example, and apply a well known utility definition suggested by Blum et al. [17].

**Definition 15** ((\( \alpha, \beta \))-useful). A database access mechanism \( M \) is \((\alpha, \beta)\)-useful with respect to COS query, if for every database \( D \), with probability at least \( 1 - \beta \), the output of the mechanism \( M \) satisfies:

\[
Pr[\max | \cos(u', v') - \cos(u, v) | \leq \alpha] \geq 1 - \beta 
\]

(3.5.7)

Based on this definition, we prove that the Johnson-Lindenstrauss transform is bounded by \( \alpha \) with high probability.

**Theorem 5.** For COS query on any pair of users, the output error caused by the Johnson-Lindenstrauss transform is less than \( \alpha \) with the probability at least \( 1 - \beta \). The proposed method is satisfied with \((\alpha, \beta)\)-useful when \( \alpha < \frac{2}{\delta - 1} \).
Proof.

The relationship between $l_2$ Euclidean distance and cosine similarity is:

\[
\|u - v\| = \|u\| + \|v\| - 2u^Tv = \|u\| + \|v\| - 2\|u\|\|v\|\cos(u, v)
\]

According to Johnson-Lindenstrauss Lemma 3.3.1, we get

\[
\cos(u, v) \geq \left(\frac{\|u\| + \|v\|}{2\|u\|\|v\|}(1 - \delta) - \|u' - v'\|\right)
\]

Then, we have

\[
\max|\cos(u', v') - \cos(u, v)| \leq \frac{\|u'\| + \|v'\| - \|u' - v'\|}{2\|u'\|\|v'\|}\frac{2\|u\|\|v\| - (\|u\| + \|v\|)(1 - \delta)}{2\|u\|\|v\|}(1 - \delta)
\]

\[
\leq \frac{\|u'\| + \|v'\| - (\|u\| + \|v\|)(1 - \delta)}{2\|u\|\|v\|}(1 - \delta)
\]

\[
\leq \frac{1}{2(\|u\| + \|v\|)(1 - \delta)} - \frac{1}{2}
\]

\[
\leq \frac{1}{2(\|u\| + \|v\|)(1 - \delta)} - \frac{1}{2}
\]

Accordingly,

\[
Pr[\max|\cos(u', v') - \cos(u, v)| > \alpha] = Pr\left[\frac{\|X\|}{2(1 - \delta)} - \frac{1}{2} > \alpha\right]
\]

\[
= Pr[\sqrt{dm}|x| > 2(\alpha + \frac{1}{2})(1 - \delta)]
\]

\[
= Pr[|x| > \frac{2(\alpha + \frac{1}{2})(1 - \delta)}{\sqrt{dm}}]
\]

57
Because variable \( x \sim N(0, 1/m) \), according to the property of Gaussian distribution, we have

\[
Pr[\max|\cos(u', v') - \cos(u, v)| > \alpha] = 2 \int_{\infty}^{\infty} \frac{\sqrt{m}}{\sqrt{2\pi}} e^{-\frac{m}{2} x^2} dx
\]

\[
= 2\sqrt{m} \left( \int_{0}^{\infty} e^{-\frac{m}{2} r^2} r dr \right)
\]

\[
= 2\sqrt{m} \sqrt{2\pi} \int_{0}^{\infty} e^{-\frac{m}{2} r^2} r dr
\]

\[
= 2\sqrt{m} \sqrt{2\pi} \left( -\frac{2\pi}{m} e^{-\frac{m}{2} r^2} \right)_{\infty}^{\infty}
\]

\[
= 2e^{-\frac{2}{\sqrt{d}}(\alpha + \frac{1}{2})(1-\delta)}
\]

let \( 2e^{-\frac{2}{\sqrt{d}}(\alpha + \frac{1}{2})(1-\delta)} = \beta \), we have

\[
e^{-\frac{2}{\sqrt{d}}(\alpha + \frac{1}{2})(1-\delta)} = \frac{\beta}{2}
\]

\[
\Rightarrow \frac{2}{\sqrt{d}}(\alpha + \frac{1}{2})(1-\delta) = -\ln\frac{\beta}{2}
\]

\[
\Rightarrow \alpha = \frac{\sqrt{d} \ln \frac{\beta}{2}}{\delta - 1} - \frac{1}{2} < \frac{\sqrt{d} \ln \frac{\beta}{2}}{\delta - 1}
\]

(3.5.8)

et \( 2e^{-\sqrt{\frac{d}{2}}(\alpha + \frac{1}{2})(1-\delta)} \leq \beta \), we have

\[
e^{-\sqrt{\frac{d}{2}}(\alpha + \frac{1}{2})(1-\delta)} \leq \frac{\beta}{2}
\]

\[
\Rightarrow \sqrt{\frac{2}{d}}(\alpha + \frac{1}{2})(1-\delta) \geq -\ln\frac{\beta}{2}
\]

\[
\Rightarrow \alpha \geq \frac{\sqrt{d} \ln \frac{\beta}{2}}{\delta - 1} - \frac{1}{2}
\]

(3.5.9)

\[\Box\]
Equation 3.5.8 shows that the error introduced by Johnson-Lindenstrauss is proportional to the square root of parameter $d$, which is the dimension of dataset $D$ and inversely proportionate to $\delta$. This means that a lower dimension dataset contributes to smaller error and a higher value of $\delta$ indicates that the similarity is closer to the original one. As $m = \Omega\left(\log(n)/\delta^2\right)$, in the experiment part, we observe in the experiment section that with the decrease in the value of $m$, prediction accuracy is closer to the traditional collaborative filtering method without the privacy operation.

For a neighbourhood-based method, the quality of neighbours is decisive inaccuracy, because the prediction is based on the neighbours’ rating history. In proposed method, the neighbours of user $u_i$ are selected based on Cosine Similarity, which is a two norm distance that is kept by Johnson Lindenstrass transform. It controls the error in the range of $[-\delta, \delta]$, where $0 \leq \delta \leq 1/2$. Science the original rating dataset is sparse, the neighbours selected based on Cosine Similarity might not the best ones that have the same taste with user $u_i$. In proposed method, Johnson Lindenstrass transform tighten the original dataset, make it have more reference for neighbour selecting. It is possible to help to choose much closer neighbours, which can contribute to better prediction. In experiment part, we apply Root Mean Square Error ($RMSE$) as the error metric to prove the utility of our method. The $RMSE$ is defined as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{u=1}^{n} (r_{ut} - \hat{r}_{ut})^2}$$  \hspace{1cm} (3.5.10)

$RMSE$ represents the performance of prediction algorithm. A lower value implies a better performance. Obviously, when the predict ratings are very close to the true answer, the $RMSE$ closes to 0.
3.5.3 Time Complexity Analysis

Given \( n \) vectors with \( d \) dimension, compute all distance pairs. The time complexity for traditional methods is \( O(n^2d) \), while, for our method of applying the Johnson-Lindenstrauss transform, the time complexity is \( O(n^2\log(n) + d\log(n)) \). If we map down the dimension from the originally very large space to a new much smaller dimensional space, \( \frac{\log(n)}{\delta^2} \) would be very small, reducing the time complexity significantly.

3.6 Experiment and Analysis

3.6.1 Datasets

In the experiments, we used the Netflix\textsuperscript{1} and MovieLens\textsuperscript{2} datasets. The Netflix Prize dataset is a real dataset released by Netflix, consisting of about 100 million movie ratings accumulated over several years, which is sufficient to evaluate the performance of the proposed method. Each movie was rated by 20 – 250 users and each user rated at least 20 movies. The MovieLens dataset has about 1 million ratings rated by 6040 users on 3900 movies. We randomly chose one rating per user as the test dataset and delete it from the original dataset. Our objective is to predict the missing ratings of each user using the remaining ratings, and we applied root mean square error (\( RMSE \)) to show the effectiveness of our method. \( RMSE \) is defined as follows:

\[
RMSE = \sqrt{\frac{\sum_{u_i,t \in T} (r_{u_it} - \hat{r}_{u_it})^2}{|T|}}
\]  \hspace{1cm} (3.6.1)

where \( r_{u_it} \) is the true rating user \( u_i \) gives to item \( t \), and \( \hat{r}_{u_it} \) is the predicted rating. \( T \) is the test dataset and \( |T| \) is its size. A lower \( RMSE \) represent a higher accuracy.

\textsuperscript{1}http://www.netflixprize.com
\textsuperscript{2}http://www.grouplens.org
### Table 3.2: Characteristics of the datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of item</th>
<th>Number of rated item</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20-500</td>
<td>501-1000</td>
</tr>
<tr>
<td>Netflix</td>
<td>5000</td>
<td>4826</td>
</tr>
<tr>
<td>MovieLens</td>
<td>3952</td>
<td>5644</td>
</tr>
</tbody>
</table>

**Figure 3.2: Description of item rating**

Table 3.3 and the Fig. 3.2 show the characteristics of the datasets we used. We can see that the datasets are high dimensional and the rating records are very sparse. Most users rated fewer than 500 items.

### 3.6.2 Experimental Setup

We conducted extensive experiments to evaluate the performance of our methods by answering the following questions:

- *How does parameter m affect the prediction?*
\( m \) is the dimension of the transfer matrix, which is an important parameter for the Johnson-Lindenstrauss transform. It determines the error introduced to the result of the query. A smaller \( m \) means a lower dimension, which reduces the computation complexity, but at the cost of accuracy. We test the effect of \( m \) in terms of \( RMSE \) in the experiment.

- **How does our method perform compared to related work?**

  To show the advantage of our method, we compared it to many other methods. We applied traditional differential privacy as the baseline mechanism, then compared it with the work by Mohammad’s [8] work.

- **How does our privacy operation affect prediction accuracy?**

  To determine how the Johnson-Lindenstrauss transform affects the final prediction, we compared it with a traditional collaborative filtering method that has no privacy operations in terms of \( RMSE \). In addition, we compared the Laplace and Stretching mechanisms with a traditional collaborative filtering method to illustrate the effect of these methods on prediction accuracy.

  For a fair comparison, all the experiments were deployed with the same privacy parameters. Section 3.4 shows that our method provides \( \epsilon \)-differential privacy, where 
  \[ \epsilon = \frac{\max_{i}^n r_{i}}{2 \cdot \text{tr}(L)}. \]
  Refer to the parameters of used datasets, The \( \epsilon < \frac{1}{2} \). We define \( \epsilon = \frac{1}{2} \) for all other methods to make them comparable. Stretching method [8] works by multiplying a scaling factor \( v_i \) to rescale each tuple, then releasing the perturbed query results with privacy parameter \( \epsilon \). We make \( v_i = \frac{1}{2} \) and \( \epsilon = 1 \). The Stretching method satisfies \( \epsilon \)-differential privacy, where \( \epsilon = v_i \epsilon = \frac{1}{2} \). Table 3.3 lists the parameters and default values used in our experiment. In addition, the sensitivity used in the
experiments is global sensitivity. The neighbours are selected based on two popular measurement metrics, Pearson Correlation Coefficient (PCC) and Cosine Similarity (COS).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>perturbed matrix dimension</td>
<td>500</td>
</tr>
<tr>
<td>$k$</td>
<td>number of neighbours</td>
<td>10</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>privacy parameter</td>
<td>0.5</td>
</tr>
<tr>
<td>$\mu$</td>
<td>expectation</td>
<td>0</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>variance</td>
<td>1</td>
</tr>
</tbody>
</table>

3.6.3 Effect of Dimension $m$

In the proposed method, $m$ is an important parameter that determines the dimension of the new matrix. It directly impacts the performance of the prediction. To determine how $m$ contributes to the final prediction, we change the value of $m$ from 100 to 5000 with a step of 100 for Netflix, and from 100 to 3900 with a step of 100 for MovieLens. The transfer matrix entries are sampled from standard normal distribution $N(0, 1)$.

Fig. 3.3 shows the results on Netflix and MovieLens, illustrating that the performance of the proposed method is greatly improved by increasing the value of $m$. When the dimension of the perturbed dataset increases, the $RMSE$ shows a downward trend on both Netflix and MovieLens. As shown in Fig. 3.3a, when $m = 10$ on Netflix, an $RMSE$ of 0.9833 is achieved. When $m$ is increased, the $RMSE$ decreases
Figure 3.3: Impact of $m$ on prediction performance

rapidly. When $m = 1000$, the $RMSE$ is at 0.9422. After that, the performance starts to level off. A similar result can be seen on MovieLens. As shown in Fig. 3.3b, when $m$ is in the range $[10, 1000]$, the changes in $RMSE$ is obvious. The prediction is much more accurate when the dimension is much higher. When it reaches a threshold, the performance is not expected to improve dramatically.

According to the Johnson-Lindenstrauss Lemma, the error caused by the linear transform is related to the dimension of the perturbed dataset. Give a distance error $Er < 2\delta$, the bigger $m$ becomes, the smaller $\delta$ becomes, and the error would be controlled in a much smaller range. Therefore, if $m$ is given a large value, performance is better than if $m$ is smaller.

It is clear that the $RMSE$ achieved by the proposed method achieved is very close to that of the non-privacy method, but the value is smaller. This means that when $k = 10$ (10 neighbours are selected), the performance of the proposed method is even better than traditional collaborative filtering without a privacy operation, whether on Netflix or Movielens. Traditional collaborative filtering achieves $RMSE$ of 0.9873
on Netflix. As shown in Fig. 3.3a, when \( m > 20 \), all the values of \( RMSE \) achieved by our method are smaller than 0.9873. This means that the proposed method even outperforms non-privacy method when \( m > 20 \). When \( m < 20 \), the \( RMSE \) achieved by our method is only 1% bigger than the non-privacy method. The results on the MovieLens dataset are shown in Fig. 3.3b. We observe that whatever the value of \( m \) (\( m \geq 10 \) in the experiment), the performance of the proposed method was always better than traditional collaborative filtering. This means that our method achieves its privacy preservation objective while enhancing prediction accuracy, which can be attributed to the randomized operation that contributes to the improvement in performance.

### 3.6.4 Comparison with Other Related Methods

In this section, we examine the performance of the proposed method in relation to state-of-art methods in terms of \( RMSE \). We compared our method with the traditional Laplace and Stretching mechanisms, using the same privacy parameters. We defined the dimension of the transformed matrix as \( m = 500 \). As shown in Fig. 3.3, when \( m = 500, k = 10 \), the proposed method had lower computational complexity and achieved good performance on both Netflix and MovieLens. To conduct a comprehensive examination, we varied the number of neighbours between 5 and 50 in Step 5 on two datasets, and defined the neighbours in both COS and PCC.

Fig. 3.4 shows the results of the comparison. It is clear that the proposed method outperforms all other methods in all configurations. Fig. 3.5a and 3.5b show the performance of all methods with the COS matrix on Netflix and MovieLens respectively. Fig. 3.5c and 3.5d show the performance of these methods with the PCC
Figure 3.4: Comparison with other related works
matrix on each of these datasets. As shown in Fig. 3.5a, all four methods achieve better $RMSE$ as $k$ increases. This is because the quality of the prediction depends on the quality of the selected neighbours. The greater the number of neighbours, the more likelihood there is of selecting good quality neighbours. When the number of neighbours exceeds a certain value, however, selecting too many bad neighbours can reduce prediction accuracy. Our method performed much better than the $Laplace$ and $Stretching$ mechanisms in all cases and across all $k$ values. When $k = 15$ on Netflix with the COS metric, The Johnson-Lindenstrauss transform method achieves $RMSE$ at 0.9344, outperforming the result of 0.9845 by the $Stretching$ mechanism by 5.01% and the result of 0.9876 by the $Laplace$ mechanism by 5.32%. When $k > 25$, the values of $RMSE$ stabilize and remain at level, but the Johnson-Lindenstrauss transform method still outperforms the other two methods. A similar result can be observed on Netflix with the PCC matrix in Fig. 3.5c.

Fig. 3.5b and 3.5d show the results on the MovieLens dataset. It is observed that the performance of the Johnson-Lindenstrauss transform also outperforms the other two methods across all $k$ values. When $k = 20$, the Johnson-Lindenstrauss transform method outperforms other methods by around 4% in terms of $RMSE$ regardless of whether the COS metric or PCC metric is used. This advantage is retained no matter how many neighbours are selected.

3.6.5 Comparison with Non-privacy Collaborative Filtering

To examine the effect of all these methods on prediction accuracy, we compare them with a traditional collaborative filtering method without a privacy operation. We call this the non-privacy method. As in Section 3.6.4, we vary the value $k$ from 5 to 50
in Step 5 on Netflix and MovieLens. The neighbours are still defined on both COS and PCC and $m$ is assigned a value of 500.

Fig. 3.5 shows the results on Netflix and MovieLens when $m = 500$. It is clear that the performance of both the Laplace and Stretching mechanisms is very close to the performance of the non-privacy method, but not as good. This means that although these two mechanisms can preserve users’ privacy, it is at the cost of utility. However, it is observed that the Johnson-Lindenstrauss transform significantly outperforms the
non-privacy method across all $k$ values. The utility of the dataset is not sacrificed, and the accuracy of the prediction is enhanced. When $k = 15$, the Johnson-Lindenstrauss transform method outperforms the non-privacy method by more than 4% on Netflix with the COS metric, and by around 3% with the PCC metric. In contrast, the prediction accuracy achieved by the other two methods is not as good as that achieved by non-privacy methods. Similar results can be seen on MovieLens with both the COS metric and the PCC metric, shown in Fig. 3.5b and Fig. 3.5d respectively.

The results of the experiments show that when $m = 500$, the proposed method is always better than the non-privacy method, no matter how many neighbours are selected. As mentioned in Section 3.6.3, the positive result is due to the randomized operation, which contributes to improved performance.

3.7 Summary

The privacy problem is a big concern in recommendation systems. Users would be more likely to contribute more extensively to systems if their personal information could be preserved. Differential privacy has become a proven, well-accepted privacy model for guaranteeing user privacy in recent years. In spite of this strong privacy guarantee, however, existing differential privacy methods preserve users’ privacy at the cost of utility. This chapter studies the privacy preservation problem in neighbourhood-based collaborative filtering, and proposes a Johnson-Lindenstrauss-based method to preserve the information of individual users while improving the performance of the recommender system. Theoretical analysis shows that the proposed method satisfies $\epsilon$-differential privacy. In addition, the proposed method outperforms the state-of-the-art methods.
Chapter 4

Privacy-Preserving in the Internet of Things

4.1 Introduction

Data aggregation is considered to be an essential research topic in the Internet of Things (IoT). For example, energy companies collect and aggregate utility data from sensors installed at customer sites, which is used to improve the overall reliability and efficiency of their infrastructure [79]. Likewise, in traffic monitoring systems, traffic flow data is collected by road-side sensors and used to analyze the network to improve services for drivers [163]. In wireless body area networks, health data is collected through mobile or wearable devices to monitor a user’s health indicators, but aggregated data is needed for medical research [52].

Given the often sensitive nature of the data involved, privacy is an important issue in data aggregation. For instance, health data, such as blood pressure and temperature, can reveal a user’s health status, and electricity usage patterns can be used to profile a customer’s lifestyle and daily routines [135]. For this reason, many people choose not to participate in sensory systems without a strong guarantee of
Methods to preserve the privacy of aggregated data have been developed by several scholars [54, 64, 118, 168, 173]. However, most are based on encryption technology, such as homomorphic encryption. For example, Dong et al.’s [30] data aggregation method for smart grids is based on ElGamal-based homomorphic privacy preservation, while Abdallah et al.’s scheme [3] introduce lightweight lattice-based homomorphic privacy preservation. Despite these efforts, there are many problems with the existing methods.

- **Computation overhead.** Homomorphic encryption typically results in massive computational overheads [79], which increases the burden of processing and analysis on cloud services. Additionally, these methods are not practical for sensors with limited energy.

- **Communication efficiency.** The communication overheads are high, especially when the system contains thousands of sensors with high reporting frequency, because each sensor needs to report its encrypted data to the cloud at the same time.

- **Single aggregation function calculation.** Most existing methods can only calculate a single aggregation function. In practice, the ideal aggregation scheme would allow flexible aggregation queries to meet diversified aggregation goals with only one round of communication.

To solve these problems, we propose a privacy-preserving data aggregation method based on machine learning within a fog computing architecture. Fog computing architectures distribute computation and data storage to the edge of the network, i.e.,
to devices that sit between the data source and the cloud server. This type of architecture reduces the amount of data transported to the cloud, improving efficiency and alleviating much of the burden on the server itself. Additionally, in our method, the aggregator resides at the center of the fog and only the aggregation results are reported to the cloud server, which significantly increases communication efficiency. Aggregation queries are answered by learning a model, which is trained to predict the query results through a process that satisfies differential privacy. Multiple aggregation functions can be calculated, including additive aggregation and non-additive aggregation. Finally, the method does not apply encryption technology, so the sensors only need to report raw data without the need for a complex cipher process.

In summary, this chapter offers the following contributions.

- We propose a novel privacy-preserving data aggregation method under fog computing architecture, which reduces the communication overhead and releases the cloud burdens.

- The proposed privacy-preserving data aggregation method is based on machine learning. The trained learning model can be used to predict the aggregation query results and supports multiple aggregation functions, which allows the server provides various services.

- The proposed data aggregation method satisfies differential privacy, which provides rigorous privacy protection for sensory data. Efficiently defend the differential attack that appears in most aggregation functions.

- We theoretically analyse the privacy and utility of proposed methods and extensive experimental results show that the proposed method generates highly
accurate aggregated results.

The rest of this chapter is organized as follows. Section 4.2 proposes the research problem. We present our privacy preservation method and theoretically analyze its privacy and utility in Sections 4.3 and 4.4, respectively. Section 4.5 details the results of the experiments. Section 4.6 discusses the related work, and Section 4.7 concludes the chapter.

4.2 Problem Statement

4.2.1 Notations

Let $S_{fc} = \{s_1, s_2, ..., s_g\}$ be a group of sensors. These sensors report the sensory data to the fog nodes $f_1$ and $f_2$. The fog node trains a learning model $M$ using the collected data and predicts the query results. Let $Q\{q_1, q_2, ..., q_t\}$ be a set of queries which generated by the fog center. Additional notations are shown in Table 4.1.

4.2.2 System Model

As shown in Fig. 4.1, the system model is composed of four entities: sensors, fog nodes, the fog center, and a cloud server. A description of each entity follows.

- **Sensors**: The sensors, which might be embedded in smart devices, collect the data. To address privacy concerns, the original data is partitioned and separately reported to two fixed fog nodes.

- **Fog nodes**: The fog nodes are efficient devices for computing and storing data that extend the edge of the cloud service. These devices serve as storage to
Table 4.1: Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>query</td>
</tr>
<tr>
<td>$Q$</td>
<td>query set</td>
</tr>
<tr>
<td>$s_i$</td>
<td>sensor</td>
</tr>
<tr>
<td>$f_i$</td>
<td>fog node</td>
</tr>
<tr>
<td>$S_Q$</td>
<td>query sensitivity</td>
</tr>
<tr>
<td>$S_{max}$</td>
<td>biggest value of query sensitivity</td>
</tr>
<tr>
<td>$QA$</td>
<td>query results</td>
</tr>
<tr>
<td>$\hat{QA}$</td>
<td>perturbed query results</td>
</tr>
<tr>
<td>$M$</td>
<td>training model</td>
</tr>
</tbody>
</table>

answer aggregation queries sent from the fog center.

- **Fog center**: The fog center is in charge of three important tasks. First, it transfers queries to the appropriate aggregation query set to be answered by the fog nodes. Second, it gathers the returned query results from the fog nodes. Third, it calculates the original query results and reports them to the cloud server.

- **Cloud server**: The cloud server is managed by the service provider and deployed as the aggregation application. This server is powerful and is used to process and analyze large amounts of aggregation data to provide information and assist with a wide range of services.
4.2.3 Adversary Model

In this chapter, we assume that the cloud server and the fog center are untrusted. Both will try to acquire the true values of the collected data, which is either sensitive or could be used to infer private information about the service users, or both. The fog nodes are semi-trusted, which means they are curious about the collected data but are not able to collude with each other.

4.2.4 Design Objectives

Our objective is to design an efficient data aggregation method that preserves the privacy of the users’ data and allows for multifunctional aggregation queries in an IoT setting. Within this problem, there are three primary objectives:

- to ensure multifunctional aggregation is implemented correctly. To suit practical
requirements, the method must include flexible aggregation functions to meet diverse analysis requirements for a wide and diverse range of services. Therefore, a mechanism that can satisfy multifunctional aggregation requirements and flexibly answer a range of data aggregation queries is highly desirable.

- to guarantee the privacy of the collected user data. Adversarial models consider possible privacy threats to an individual’s privacy and, given that the data collected often pertains to a user’s health or behavioral habits, the aggregation scheme developed must satisfy each individual’s privacy with a guarantee of $\epsilon$-differential privacy.

- to ensure the aggregation results are close to the results without privacy protection. As the proposed system needs to satisfy $\epsilon$-differential privacy, any noise added to the training set will reduce the accuracy of the aggregation results. (How accuracy is evaluated is defined in Definition 17.) Hence, the method must include a way to adjust the sensitivity and the amount of added noise to ensure the accuracy of the aggregation results are $(\alpha, \beta)$-useful.

### 4.3 Proposed Scheme

In this chapter, we propose a multifunctional aggregation framework based on machine learning. In general, the data collected from each region are used to train a learning model, which, in turn, is used to predict multiple query results. The predicted query results are then further processed to calculate the required aggregation function. This framework is able to deliver multifunctional aggregation in one round of communication without disclosing the sensory data to any party.
Fig. 4.2 illustrates the complete aggregation process. Within the framework, two fog nodes are in charge of collecting data from each region. Once a sensor collects some information, it randomly partitions the data into two parts and separately transmits one part to each of the two fog nodes. Because the fog nodes cannot collude, neither node can integrate or infer the true values of the sensor data. Each fog node receives data from many sensors, and once assembled, the fog node trains a learning model using the data it has received. Once trained, the learning model is able to predict the summation of any sensor’s value. To defend against differential attacks, the training dataset is generated using a process that satisfies differential privacy. The fog center fetches the query results from the two fog nodes, calculates the aggregation results, and returns those results to the cloud server.

4.3.1 Data Aggregation Protocol

This section presents the proposed privacy-preserving data aggregation method. The method includes three stages: processing the query, generating the sensor report, and predicting the query results while preserving privacy.
Query Processing

As previously mentioned, this method supports multiple functions simultaneously. Allowable query functions are $\text{min}$, $\text{max}$, $\text{medium}$, $\sigma$-$\text{percentile}$, $\text{average}$, and $\text{summation}$ aggregation. The cloud server sends all these queries together to the fog center. The fog center sends each newly generated query set to a fog node to be answered, and the fog node returns the results to the cloud server. In detail, the process is as follows:

- **Step 1**: Query set generation. The fog nodes cannot answer $\text{min}$, $\text{max}$, $\text{medium}$, $\sigma$-$\text{percentile}$, and average queries directly, which means the fog center must generate the proper queries first. To illustrate this process, consider a $\text{min}$ aggregation as an example. Assuming the query $q = \text{min}(s_1, s_3, s_4)$ represents the $\text{min}$ value of sensors $s_1$, $s_3$ and $s_4$, the fog center generates three independent queries to determine the value of each sensor, as shown in Table II. The same method is used for $\text{max}$, $\text{medium}$, and $\sigma$-$\text{percentile}$ queries.

<table>
<thead>
<tr>
<th>New Query</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$q_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$q_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

To calculate average queries, we simply sum the values of the queried sensor.

- **Calculating the original query results.** Assume a query set $Q(q_1, q_2, \ldots q_n)$ is a newly generated query that requires different aggregation functions, say, $\text{min}$,
max, medium, \( \sigma \)-percentile and average. The methods for calculating the corresponding query results are shown below:

- Min: \( \min_D = \min\{q_1(D), q_2(D), \ldots q_n(D)\} \)
- Max: \( \max_D = \max\{q_1(D), q_2(D), \ldots q_n(D)\} \)
- Medium: If \( n \) is odd, \( \text{med}_D = q_i(D) \), where
  \[
  \begin{cases}
  \lceil [\min, q_i(D)] \rceil \geq (n + 1)/2 \\
  \lceil [q_i(D), \max] \rceil \geq (n + 1)/2
  \end{cases}
  \]
  If \( n \) is even, \( \text{med}_D = (q_i(D) + q_j(D))/2 \), where
  \[
  \begin{cases}
  \lceil [\min, q_i(D)] \rceil \geq n/2 \\
  \lceil [q_i(D), \max] \rceil \geq n/2 + 1 \\
  \lceil [\min, q_j(D)] \rceil \geq n/2 + 1 \\
  \lceil [q_j(D), \max] \rceil \geq n/2
  \end{cases}
  \]
- \( \sigma \)-percentile: \( \text{per}_D = q_i(D) \), where
  \[
  \begin{cases}
  \lceil [\min, q_i(D)] \rceil \geq \lfloor \sigma n/100 \rfloor \\
  \lceil [q_i(D), \max] \rceil \geq \lfloor (100 - \sigma)n/100 \rfloor
  \end{cases}
  \]
- Average: \( \text{ave}_D = \frac{\sum q_i(D)}{n} \)
- Summation: \( \text{sum}_D = \sum_{i=1}^{n} q_i(D) \)

In the above, \( \min, \max, \text{medium}, \sigma \)-percentile, and average of a dataset \( D \) are denoted as \( \min_D, \max_D, \text{med}_D, \text{per}_D, \) and \( \text{ave}_D \), respectively. \([a, b]\) refers to the number of values that fall within the range \([a, b]\). Once calculated, the fog center sends the aggregation results back to the cloud server for further processing.
Sensor Report Generation

Assume that the sensors report their sensory data to the fog nodes every 15 minutes. And to provide the required range of services, they must report their data simultaneously. To avoid disclosing any real information to the fog nodes, a simple algorithm that resides on the sensor device partitions the data before it is sent. Specifically, each sensor $s_i \in S_{fc}$ gathers sensory data $m$ at time point $t_\gamma$ and carries out the following protocol:

- **Step 1:** A random number $\kappa \in 0\ m$ is generated for the current time point $t_\gamma$.

- **Step 2:** The sensor reports $\kappa$ to the fog node $f_1$ through a wireless network.

- **Step 3:** The sensor calculates the value of $\iota$ and reports it to the fog node $f_2$, where $\iota = m - \kappa$.

Predicting the query results while preserving privacy

After receiving all the reported data from the sensors, the fog node predicts the query results according to the following steps:

- **Step 1:** Generate a training set.

The fog node generates a query set $QS$ with $\nu$ queries. Each query includes $S_{fc}$ features, which are the features of the sensory data. Sensitivity needs to be considered during the process of generating the training set because, without proper calibration, substantial errors can occur. Query sensitivity is defined as follows:
Definition 4.3.1 (Query sensitivity:). Given a group of queries $Q(q_1, q_2, ..., q_\nu)$ over a data set $D$, the query sensitivity $S_Q$ is defined as follow:

$$S_Q = \max \sum_{i=1}^{\nu} \text{sign}(| q_i(D) - q_i(D') |),$$  \hspace{1cm} (4.3.1)

where $D'$ is the neighbouring dataset of $D$.

Query sensitivity evaluates how many queries results are affected by a single record. To reduce the query sensitivity, the feature being queried is controlled within $S_{\text{max}}$ times in each query set, where $S_{\text{max}} \leq \nu$.

To ensure the model satisfies differential privacy and can defend against differential attacks, Laplace noise is added to the query results. Specifically, the noisy answer $\hat{QA} = QA + \{Lap(S_{\text{max}}/\epsilon), Lap(S_{\text{max}}/\epsilon), ..., Lap(S_{\text{max}}/\epsilon)\}$, where $QA$ represents the vector of the query results.

- **Step 2:** Training the learning model.

  The training set generated in the last step $< Q, \hat{QA} >$ is used to train the learning model. Given the sensory data is made up of numerical values, the model $M$ could be trained using a variety of regression algorithms. In this chapter, we used a simple linear regression algorithm that demonstrated good performance during the experiments.

- **Step 3:** Predicting the query results.

  The trained model is then used to predict the results of fresh queries $Q$ sent by fog center. Specifically, $Q(\hat{D}) = MQ$, $Q(\hat{D})$ is the noisy answers of queries.

In summary, the proposed method addresses the three challenges mentioned in Section 4.1 - computation overheads, communication overheads, and multifunctional
aggregation. The lack of required encryption technology ameliorates the computation overhead, and introducing a machine learning process coupled with a fog architecture allows for more powerful computing power and greater storage capabilities. As such, the sensor nodes only need to report raw, unprocessed data, and the fog center distributes tasks to a number of fog nodes, which reduces the burden on the cloud server. Communication efficiency is improved by only reporting the aggregation results to the cloud server rather than all the sensory data. And the last section demonstrates the power of multifunctional aggregation within the proposed protocol.

4.4 Privacy and Utility

In this section, we theoretically analyze the privacy and utility of our method.

4.4.1 Privacy Analysis

In the proposed method, generating the training set is the only process that consumes the privacy budget. Theorem 10 shows that the proposed data release method satisfies \( \epsilon \)-differential privacy.

**Theorem 6.** Each record in a given dataset \( D \) represents the sensory data of one sensor, and each record is independent of the others. Thus, the proposed privacy-preserving aggregation method can provide \( \epsilon \) - differential privacy.

**Proof.**

Let \( Q \) be a set of training queries. Laplace noise is added to the query results, generating a noisy answer \( \hat{Q}(D) = Q(D) + Laplace(s_{max}/\epsilon) \). Throughout the entire
process, the original dataset $D$ can only be accessed by the training queries. The process for training the model is based on the training dataset, whereas the prediction process is based on the trained learning model. These processes do not consume any of the privacy budget and cannot disclose any private information because the original dataset is not interrogated. Therefore, every aspect of this aggregation method satisfies $\epsilon$-differential privacy. Additionally, the original sensory data is divided into two parts and reported separately to the two fog nodes. Each fog node conducts its protocols independently. Hence, each fog node also satisfies $\epsilon$-differential privacy.

In the analysis below, we examine the composite property of the privacy budget for the entire dataset to determine the privacy guarantee is satisfied.

**Theorem 7 (Parallel Composition [93]).** Assume we have a set of privacy mechanisms $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$, and each $\mathcal{M}_i$ provides $\epsilon_i$ privacy guarantee on a disjoint subset of the entire dataset, $M$ provides $\max(\epsilon_i)$ - differential privacy.

Theorem. 11 directly illustrates the privacy guarantee in the proposed method. The sensory data is sliced into two parts; therefore, the data received by the fog nodes are disjoint and independent of each other. According to Theorem. 11, the set of privacy mechanisms $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$ will consume the $\max\{\epsilon_1, \epsilon_2, \ldots, \epsilon_m\}$ of the privacy budget. In our method, each fog node is assigned the same privacy budget; therefore, the proposed method preserves differential privacy.

4.4.2 Utility Analysis

In this section, we apply a well-known utility definition suggested by Blum et al. [17] to measure the accuracy of the proposed privacy framework.
Definition 16 ((α, β)-useful). A mechanism \( M \) is \((α, β)\)-useful with respect to a set of queries, if for every data set \( D \), with a probability of at least \( 1 - β \), the output of the mechanism \( M \) satisfies

\[
Pr[\max |\hat{M}(cell_i) - M(cell_i)| \leq α] \geq 1 - β.
\]

(4.4.1)

Based on the definition of accuracy (Definition 17), we demonstrate that a certain value of \( α \) bounds the errors caused by our method with a high probability.

Theorem 8. The output errors of a set of the queries on collected data caused by the proposed method is bounded by \( α \) with a probability of at least \( 1 - β \). The proposed method is satisfied with \((α, β)\)-usefulness when \( α < \max \{ \sqrt{\frac{4sln2|H|}{m\sigma}}, \sqrt{\frac{n^2ln|H|}{m\sigma}} \} \).

Proof.

The errors caused by the proposed method occur when noise is added to the training set and when training the model. Suppose the chosen learning algorithm for the model has a hypothesis set \( H = \{h_1, h_2, \ldots, h_i\} \) of size \(|H|\). The error probability is denoted as follows:

\[
Pr[error] \leq Pr[error_n] + Pr[error_m],
\]

(4.4.2)

where \( error_n \) refers to the errors caused by adding noise, and \( error_m \) refers to the errors caused by the training model.

To satisfy differential privacy, \( Laplace \) noise is added to the entire training set. The level of error can be calculated using the properties of \( Laplace \) noise, presented as sums of \( Laplace \) random variables, as shown in Lemma 1.

Lemma 1 (Sums of \( Laplace \) random Variables [67]). Let \( \lambda_1, \lambda_2, \ldots, \lambda_m \) be a set of independent random variables drawn from \( Laplace(\sigma) \). Then for every \( α > 0 \),

\[
Pr(|\frac{1}{m} \sum \lambda_i| > α) = exp(-\frac{mα^2}{4σ}).__{4.4.3}
\]
As $error_n = \frac{\sum |f_i(D) - f_i(D)|}{m}$, we have $Pr[error_n > \alpha] = Pr[\frac{\sum |f_i(D) - f_i(D)|}{m} < \alpha]$. For each $f_i$, the number of errors are equal to the random variable $\lambda_i$ sampled from $Laplace(\frac{2}{\epsilon})$. Therefore, $Pr[error_n > \alpha] = Pr(\frac{\sum \lambda_i}{m} > \alpha)$. According to the Lemma 1,

$$Pr[error_n > \alpha] = exp(-\frac{m\alpha^2}{4\epsilon}) = exp(-\frac{m\alpha^2}{4s})$$

(4.4.4)

For all hypotheses $h \in H$, we then have

$$Pr[error_n > \alpha] = |H| \cdot exp(-\frac{m\alpha^2}{4s}).$$

(4.4.5)

Let $\beta = 2|H| \cdot exp(-\frac{m\alpha^2}{4s})$, we have $\alpha = \sqrt{\frac{4hsln|H|}{ms}}$.

The error $error_m$ can be analyzed with the help of the Chernoff-Hoeffding bound [67], shown as follow.

Lemma 2 (Real-valued Chernoff-Hoeffding Bound [67]). Let $X_1, ..., X_m$ be independent random variables with $E[X_i] = u$ and $a \leq X_i \leq b$ for all $i$, then for every $\alpha > 0$,

$$Pr(|\frac{\sum X_i}{m}| > \alpha) \leq 2exp(-\frac{2\alpha^2m}{(b-a)^2}).$$

(4.4.6)

All queries to train the model are range queries. If the dataset has $n$ records and each value is 1, the output of the query range from 0 to $n$. As $error_m = \frac{\sum |f_i(D) - f_i(M)|}{m}$, $Pr[error_m > \alpha] = Pr[\frac{\sum |f_i(D) - f_i(M)|}{m} > \alpha]$. According to Lemma 2, for each hypothesis $h \in H$, we have $Pr[error_m > \alpha] = Pr[\frac{\sum |f_i(D) - f_i(M)|}{m} > \alpha] \leq 2exp(-\frac{2\alpha^2m}{n^2})$. Thus, for all hypothesis, we then have

$$Pr[error_m > \alpha] \leq 2|H| \cdot exp(-\frac{2\alpha^2m}{n^2}).$$

(4.4.7)

Let $\beta = 2 \times 2|H| \cdot exp(-\frac{2\alpha^2m}{n^2})$, we have $\alpha = \sqrt{\frac{ln|H|}{m}}$.

Therefore,
\[
Pr[\text{error} > \alpha] \leq Pr[\text{error}_n > \alpha] + Pr[\text{error}_m > \alpha] \\
\leq |H| \exp(-\frac{m e \alpha^2}{4s}) + 2|H| \exp(-\frac{2 \alpha^2 m}{n^2}) \quad (4.4.8)
\]

Let \( \beta = |H| \exp(-\frac{m e \alpha^2}{4s}) + 2|H| \exp(-\frac{2 \alpha^2 m}{n^2}) \), we get that when \( \alpha < \max\{\sqrt{\frac{4 \ln \frac{|H|}{\beta}}{m e}}, \sqrt{\frac{n^2 \ln \frac{|H|}{\beta}}{m}}\} \), the accuracy of proposed method satisfies the \((\alpha, \beta) - useful\) definition. In other words, the error is controlled by \( \alpha = \max\{\sqrt{\frac{4 \ln \frac{|H|}{\beta}}{m e}}, \sqrt{\frac{n^2 \ln \frac{|H|}{\beta}}{m}}\} \) with a probability of at least \( 1 - \beta \).

\[\square\]

4.5 Experiment Evaluation

4.5.1 Experimental Setup

Dataset. We used two real-world datasets to evaluate the performance of our method. The Reference Energy Disaggregation Data Set (REDD) contains specific information about the electricity consumption of many real homes over several months. MHEALTH [112] is a mobile health dataset, which contains more than 1 million records, each comprising the data from 24 different sensor signals. Given each signal is at the same scale, we randomly chose one type of signal for evaluation.

Metrics. We used the mean absolute error (MAE) to evaluate the accuracy of the results, defined as follows:

\[
MAE = \frac{1}{m} \sum_{Q_i \in Q} |Q_i(D) - Q_i(D)|, \quad (4.5.1)
\]
where $Q_i(D)$ is the true aggregation result for one query, and $\hat{Q}(\hat{D})$ is the perturbed aggregation result that calculated through our aggregation framework. A lower MAE represents a higher accuracy.

**Comparison.** Within our proposed aggregation framework, multifunctional aggregation could be achieved very simply using a traditional *Laplace* differential privacy method (LapDP). The fog node could be used as regional storage and to release the query results used in the aggregation function calculations. Hence, we compared our machine learning-based method (MLDP) to the traditional LapDP method.

**Parameters.** Table 4.3 lists the parameter settings for our experiment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Size of training set</td>
<td>$1 - 500$</td>
<td>200</td>
</tr>
<tr>
<td>$Q_s$</td>
<td>Size of query set</td>
<td>$1 - 100$</td>
<td>100</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Privacy budget</td>
<td>$0.1 - 1$</td>
<td>1</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Percentile query parameter</td>
<td>$-$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**4.5.2 Experiment Results**

To compare the performance of the proposed method with LapDP, we assessed the results of several aggregation functions - *sum*, *max*, *min*, *σ-percentile* and *average* - in terms MAE with a number of different conditions. These were:
Figure 4.3: Performance with different sizes of query set
Figure 4.4: Performance with different sizes of query set
Performance by varying size of query set

The query set is used to calculate all aggregation functions. This experiment examined the performance of the proposed method on both the REDD and MHEALTH datasets with query sets $Q_s$ ranging from 1 to 500.

Fig. 4.4 shows the impact of the size of the query set on the performance of both methods in terms of MAE. With all aggregation functions on all the datasets, LapDP’s MAE linearly increased as the size of query set grew, while MLDP remained stable. This is because, given a fixed privacy budget, the sensitivity in LapDP increases linearly with the growth of query set and, in turn, the amount of noise added to the query result also increases linearly. However, because MLDP satisfies differential privacy during the training process, the size of the query set has no effect on performance with a fixed privacy budget.

We also observed that LapDP showed better performance than MLDP with a small enough query set. But MLDP significantly outperformed LapDP as the size of the query set grew. For example, Fig. 4.4h shows the performance results for the max function on the REDD dataset. At $Q_s < 20$, MLDP has a higher MAE than LapDP, whereas at $Q_s > 20$, MLDP’s MAE is lower than LapDP. Similarly, Fig. 4.3d shows MLDP with a higher MAE than LapDP up to $Q_s \approx 18$, at which point it starts to perform better than LapDP. We find the same results for other aggregation functions on both REDD (Figs. 4.4g-4.3f) and the MHEALTH dataset (Figs. 4.6g-4.6l). For example, LapDP performed 50% better than MLDP with the min function on the MHEALTH dataset, with an MAE of 42.2 compared to MLDP’s 94.1 at $Q_s = 5$. However, at $Q_s = 14$, LapDP and MLDP show similar performance, with an MAE of 96.4 and 94.3, respectively, and at $Q_s > 20$, MLDP significantly outperforms LapDP.
Figure 4.5: Performance with different privacy budgets
Figure 4.6: Performance with different privacy budgets
These results indicate that MLDP performs well, and significantly outperforms the traditional *Laplace* method, when calculating aggregation functions on large datasets.

**Varying the levels of privacy budget**

The privacy budget determines the amount of noise that is added to the training set and the query results. To determine how the privacy budget contributes to the final aggregation results, we changed the budget from 0.1 to 1 in steps of 0.1 for both datasets and fixed the training and query sets.

Fig. 4.6 shows the variations in the tendencies of all aggregation functions for the REDD and MHEALTH datasets along with the privacy budget $\epsilon$. We observe that the MAE decreased as the privacy budget $\epsilon$ increased with both MLDP and LapDP. This is because a smaller privacy budget $\epsilon$ means more noise needs to be added. Correspondingly, as the privacy budget increases, less noise needs to be added, which means the results are less perturbed, leading to higher accuracy and a smaller MAE. In addition, we observed that our method consistently outperformed LapDP, with a lower MAE for all aggregation functions. As shown in Figs. 4.5b and 4.5c, when $\epsilon = 0.2$, LapDP scored an MAE of 1156 and 2941 for the max and min functions, respectively, while MLDP scored 47.61 and 48.17 - a significant improvement. When $\epsilon = 0.8$, LapDP resulted in an MAE of 236 for the max function and 667 for the min function, which is much larger than the MAE values of 34.24 and 60.00 for our method. We observed similar results for the other aggregation functions, as shown in Figs. 4.5a, 4.5d, 4.5e, and 4.5f. In addition, we observed that varying the privacy budget had a tremendous impact on LapDP’s performance, while MLDP only showed small changes in performance. For example, in Fig. 4.5e, when $\epsilon = 1$, LapDP’s MAE
was 190 for average aggregation, yet at $\epsilon = 0.1$, LapDP’s MAE rose to 1781 - an increase of around 90%. In contrast, MLDP’s MAE rose from 171 to 180 - an increase of only around 6%. Figs. 4.6g - 4.6l show the results for the MHEALTH dataset with similar observations. This is because LapDP has a much higher sensitivity than MLDP to begin with, which means it adds much more noise to the original data. Hence, when the privacy budget is halved, the amount of noise doubles. In LapDP’s case, this doubling results in a huge amount of noise which significantly impacts accuracy, while for MLDP, doubling the small level of initial noise does not result in nearly as great a drop in accuracy.

**Performance by varying size of training set**

Our theoretical analysis indicated that the size of the training set would play a vital role in the accuracy of the aggregation result. To observe the change in performance with different sized training sets, we increased the number of instances from 1 to 500 and tested all the aggregation functions using MLDP on both datasets. We then compared the results to the MAEs for the LapDP method with both a fixed privacy budget and fixed query set size.

Fig. 4.8 shows the results for the REDD and MHEALTH datasets, illustrating that the performance of the proposed method is greatly improved by increasing the size of the training set, initially, but once the training set reaches a certain value, the MAE reaches its nadir and become stable. As shown in Fig. 4.7e, the MAE continues to decrease until the training set contains 120 record where, at $MAE = 4.0$, the MAE reaches its lowest point. Subsequent increases in the size of the training set result in an MAE that fluctuates around 4. Fig. 4.7f shows the results for the $\sigma$ - percentile
Figure 4.7: Performance with different sizes of training set
Figure 4.8: Performance with different sizes of training set
aggregation. When the size of the training set is below 100, the MAE is very high but decreases significantly as the size of the training set increases, but at $T > 150$, the MAE no longer decreases. Similar results were observed on the MHEALTH dataset, as shown in Figs. 4.8g - 4.8l.

Given MLDP’s performance is impacted by a mixture of noise and model errors, when the size of the training set is small, the sensitivity and noise levels are small, so the model errors play a more dominant role. Hence, the MAE decreases significantly with an increase in the size of the training set. However, beyond a certain threshold, a large training set carries too much sensitivity and noise to offset the increase in accuracy size brings. At this point, noise reduces the utility of the model and the MAE stops decreasing.

### 4.6 Related Work

Existing data aggregation methods typically use homomorphic encryption when aggregating data to ensure privacy [52, 78, 170, 174–176]. Zhang et al. [176] proposed a solution based on peer-to-peer protocols, called VPA, to preserve privacy in people-centric urban sensing systems. VPA supports a wide range of statistical additive and non-additive aggregations, but cannot defend against the differential attacks common to most data aggregation scenarios. Zhang et al. [170] proposed a priority-based aggregation solution for health data (PHDA), which includes privacy protection and also improves the cloud aggregation efficiency of the cloud service and the privacy of data privacy in WBANs. PHDA uses the relationships between its users and fixed social spots to choose the best relay for providing reliable data aggregation. In addition, PHDA can also withstand both internal and external forgery attacks, but
it does not handle differential attack very well. Li et al. [78] proposed an efficient privacy-preserving protocol, called EPADA, which calculates sum aggregations from time-series data. The protocol uses additive homomorphic encryption and a novel key management technique to support a large plain-text space. Although the proposed method is easily extendable to min aggregations with just one round of communication, it is more difficult to adapt to compute multifunctional aggregations, especially non-additive aggregate functions, such as percentile and average. Han et al. [52] proposed a privacy-preserving multifunctional aggregation mechanism, also for health data. The cloud server is able to calculate multiple statistical functions and provides a range of services, each with privacy protection. This method supports both additive and non-additive aggregation functions.

However, all these schemes using encryption technology to protect the user’s data and, since encryption usually results in a significant computational overhead, they are not practical for use with energy-limited sensors like smartphones. In addition, the computational burden on the cloud server is heavy, especially when aggregating data that is reported with high frequency. A fog computing architecture allows computing services to reside at the edge of the network. Hence, a local aggregation device can be used to calculate the query results, which reduces the communication and computation overheads on the cloud server. Several papers have already explored privacy problems related to data aggregation in fog computing [44, 60, 88, 102]. For example, Huang et al. [60] proposed a model that filters multiple encrypted XML streams and performs aggregation operations without decryption in a fog node. Lu et al. [88] proposed a lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT devices. However, most also include homomorphic encryption schemes,
which does not solve the problem of sensors with limited energy resources.

Different from the existing method, the proposed method applied differential privacy technology, which significantly reduced the computation overhead compared with encryption-based method. In addition, we transfer the sensory data to a learning model, which allows multiple types of query while preventing differential attack.

### 4.7 Summary

In this chapter, we proposed a privacy-preserving multifunctional data aggregation method based on machine learning. Within the method, a training dataset comprising the aggregation queries is used to train a machine learning model, which in turn predicts the aggregation results. The method allows for multiple aggregation functions without disclosing a user’s privacy. The framework operates within a fog computing architecture, which means the computationally heavy aggregation tasks are distributed to the edge of the network, alleviating this burden from the cloud server. Additionally, only the aggregation results are sent to the server rather than all the sensory data, which significantly improves communication efficiency. Experimental results prove that the proposed method answers various aggregation queries with high accuracy.
5.1 Introduction

The pervasive diffusion of GPS-enabled devices has provided tremendous opportunities for the development of location-based services (LBSs). A typical example is providing recommendations about nearby points of interest (POIs). A user queries an LBS with their current location, and the LBS returns the corresponding POIs. Even though LBSs provide great benefits, they come at the cost of exposing a user’s location. Where a person is, is sensitive information. It can easily be linked to highly confidential details, such as their home address, and their religious practices. Therefore, devising a solution that allows users to benefit from LBSs while guaranteeing the privacy of their location is highly desirable.

Numerous location privacy protection methods have been proposed in the last decade. Most solutions proposed in the literature are based on location obfuscation. The basic idea is to transform the user’s exact location into a region large enough to thwart attacks, also known as cloaking region [75]. Unfortunately, most location
obfuscation techniques proposed rely on syntactic approaches such as k-anonymity, which cannot provide rigorous privacy [159]. For example, Hashem et al. [?] identified an overlapping rectangle attack based on the obfuscated location information. In addition, Dewri et al. [28] highlighted the inadequacy of cloaking regions in preventing location privacy breaches when the adversary grasps some approximate location knowledge about the user. Another class of technique is Private Information Retrieval (PIR), which uses cryptography to protect the user’s location information [36]. This technique allows a user to query POIs without revealing any information about the query. However, while LBS queries based on PIR provides strong cryptographic guarantees, they are often computationally and communicationally expensive and not practical in addition to requiring different query plans to be designed for different query types [159].

Differential privacy, a powerful privacy model, is widely accepted for providing rigorous privacy guarantees for aggregate data analysis [35]. It ensures that one individual cannot significantly affect the output of a query. Differential privacy is normally achieved by injecting random noise to the result of the query. Applying differential privacy for location protection is still at its early stage [159]. Dewri [28] proposed a differential location perturbation method that added Laplace noise to the \( x \) and \( y \) coordinate separately. Andres [18] introduced a generalized privacy notation geo-indistinguishability to formalize the problem of location privacy preserving. Palia and Tandon [109] further considered the impact of prior information about POIs on the utility. All of them need to add significant noise to hide the user’s exact location in the safe region, which reduces the accuracy of returned POIs. In addition, as all existing methods are designed to operate in client server structures, shown in Fig.
users inevitably expose partial information to the service provider in exchange for usable services. Once the attacker has acquired appropriate background knowledge, a user’s location privacy will be violated. For example, Huo et al. [?] inferred the users’ location information by using their check-in history. Therefore, any method based on a client-server structure is highly unlikely to provide strict location privacy preservation.

In this chapter, we propose a new Johnson Lindenstrauss transform based location privacy protection method. The Johnson Lindenstrauss lemma states that a small set of points in a high dimensional space can be embedded into a much lower dimension that the Euclidean distances between the points can be nearly preserved [27]. Therefore, the basic idea of the proposed method is that transfer the user’s exact location together with the map into another dimension in such a way that the adversary have no idea the user’s exact location but the relative distances between POIs are maintained that helps to find nearby POIs.

There are several challenges in applying Johnson Lindenstrauss transform in the location privacy protection. First, how to evaluate the privacy level protected by the
Johnson Lindenstrauss transform? Jeremias et al. [16] proved that the Johnson-Lindenstrauss transform preserves edge differential privacy in graph sanitization. Based on this finding, we show that it also guarantees differential privacy in a location dataset. Second, how to return the accurate POIs without disclosing it to the service provider? We solve this challenge by introducing a semi-trusted third party who in charge of transferring the map and return anonymized encrypted POIs. In such way, the service provider only access the transformed dataset, while the third party access the original POI information, the exact queried POI information can be returned to the user by third party and service provider intersection without disclosing location privacy to either party.

Overall, our contributions are summarized as follows:

1. We propose a new privacy framework that introduces a semi-trusted third party to protect user locations regardless of adversaries background knowledge.

2. We present a location perturbation method based on the Johnson-Lindenstrauss transform that satisfies differential privacy. The proposed method not only guarantees rigorous privacy preservation but also allows LBS providers to provide accurate services.

3. We systematically analyze how the proposed method can defend against various background knowledge attacks while providing high-quality services.

The rest of the chapter is organized as follows. We propose our privacy preservation framework and apply it to two basic POI queries in Section 5.2 and Section 5.3 respectively. Section 5.5 details the results of the experiments. Section VII discusses the related work and Section 6.7 concludes the chapter.
5.2 Location Privacy Preservation Framework based on a Third Party

5.2.1 Notations

Let $U = \{u_1, u_2, ..., u_n\}$ be a set of users. Each user has a true location and a perturbed location. The true location coordinates are denoted as $l_t(x, y)$, the perturbed location is denoted as an $m$ dimension location vector $l_p(c_1, c_2, ..., c_m)$. $M$ is the map matrix, whose records are POI location coordinates, Assume there are $t$ POIs in the map, then $M \in \mathbb{R}^{t \times 2}$. $\hat{M} \in \mathbb{R}^{t \times m}$ is the perturbed map matrix. Important symbols used in this section and following parts of the chapter are listed in Table 5.1 for reference.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_t$</td>
<td>User’s true location</td>
<td>$l_p$</td>
<td>Perturbed location</td>
</tr>
<tr>
<td>$X$</td>
<td>Transition matrix</td>
<td>$m$</td>
<td>Dimension of $X$</td>
</tr>
<tr>
<td>$M$</td>
<td>Map’s location matrix</td>
<td>$\hat{M}$</td>
<td>Perturbed map matrix</td>
</tr>
<tr>
<td>$POIs$</td>
<td>Set of POIs</td>
<td>$POIs_a$</td>
<td>Anonymized POI set</td>
</tr>
<tr>
<td>$r$</td>
<td>Radius</td>
<td>$\hat{r}$</td>
<td>Perturbed radius</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of POI</td>
<td>$\hat{M}_a$</td>
<td>Anonymized $\hat{M}$</td>
</tr>
</tbody>
</table>

5.2.2 Problem Definition and Assumption

*Problem Definition.* In this chapter, we consider the location privacy problem in the popular location-based service where the user queries the LBS server for nearby POIs.
However, the service provider may be untrusted and attacked by outside attackers. Therefore, we define the problem as follows:

**Problem 1.** Given a user who has a location \( l_t(x, y) \), \( P\{p_1, p_2, ..., p_k\} \) is the nearby POIs of the user. Design a location privacy protection method, through which, the service provider has no idea the user’s exact location, while the returned POI set \( P' \) should have the following character: the value of \( |P \triangle P'| \) should as small as possible.

**Assumptions.** To make the problem clear, we can make a few reasonable assumptions.

1. *The third party will hold the map information.*

   This is reasonable as many LBSs use public maps, e.g., Google Maps. The third party could even be a map provider.

2. *The third party is semi-trusted.*

   We assume the third party in our privacy framework is semi-trusted with following characters:
   - It curious about the user’s location information;
   - It follows the process of POI queries;
   - It does not collude with the service provider.

### 5.2.3 Framework

The proposed location privacy preservation framework contains three components as illustrated in Fig. 5.2.
Figure 5.2: Location privacy framework

**Users:** Users are people with GPS-enabled devices who ask for location-based services. They prefer to mask their exact locations. Therefore, location information will be perturbed before sending from the device. User devices are assumed to be trusted. Any malicious software would not be able to access the position sensor [45].

**LBS Server (Service Provider):** Service providers are application platforms that provide location-based services, such as *Google Maps*, *Foursquare*, and *Yelp*. These providers require a user’s location to deliver high-quality services to an individual. Service providers may be untrusted. In our proposed framework, a service provider receives a query directly from a user and returns the encrypted POIs received from a third party.

**Third Party:** The third party is semi-trusted and does not collude with the service provider. It holds a portion of the location information and acts as a bridge between the user and the service provider. The third party may acquire the map used by the
service provider to provide location-based services, and it also has the essential tasks of perturbing the map and helping the service provider return highly accurate POIs.

As a basic outline of the entire process: (a) a user perturbs their location locally using the transition matrix $X$; (b) the user query the service provider for nearby POI and query the third party for map sanitization; (c) the third party perturbs the map accordingly; (d) the service provider searches for anonymized POIs, asks the third party for exact POI information and forwards them back to the user.

5.2.4 Privacy Protection Scheme based on Johnson-Lindenstrauss Transform

In this section, we introduce a perturbation method based on the Johnson-Lindenstrauss transform that operates within the proposed privacy framework. Jeremias et al. [16] show that the Johnson-Lindenstrauss transform allows us to publish a sanitized graph that preserves edge differential privacy. Based on that, we prove that the Johnson-Lindenstrass transform preserves differential privacy for location dataset as well. According to the Johnson-Lindenstrauss lemma, the transformation can keep the relative distances between points. Therefore, it works well for answering POI queries, which is based on Euclidean distance. The details of the procedure are explained in the following parts.

a. A user has access to their location coordinates through their GPS-enabled devices, and these are perturbed by a linear transition matrix $X \in \mathbb{R}^m$. The details are shown in Algorithm 2.

Algorithm 2 shows the proposed location perturbation method. First, a random transition matrix $X$ is generated, which is constructed in Steps 1 to 5. $X$ is a $2 \times m$
Algorithm 2 Location perturbation

Require: user $u_i$’s location $l_t(x_i, y_i)$, projected dimension $m$.

Ensure: perturbed location $l_p$, transformation matrix $X \in \mathbb{R}^{2 \times m}$.

1: for $i = 1$ to 2 do
2:   for $j = 1$ to $m$ do
3:     Sample $X[i, j]$ from Gaussian distribution $N(0, 1)$;
4:   end for
5: end for
6: $l_p \leftarrow l_t X$;
7: return $l_p, X$.

matrix and the entries for $X$ are sampled from a 0 mean Gaussian distribution. We perturb the user’s location by multiplying the original location coordinate with $X$ at Step 6. Step 6 is the process of Johnson Lindenstrauss transform. After transformation, the user’s location information is changed from a location coordinate $(x, y)$ to a meaningless location vector $(c_1, c_2, ..., c_m)$. We prove that the transformation satisfies $\epsilon$- differential privacy in Section 5.4.1.

b. The user chooses one region $R$ that he/she feels comfortable with and generates a key pair $(sk, pk)$. Then, he/she queries the LBS nearby POIs using the perturbed location, and query the third party for map sanitization by sending the transition matrix $X$, region $R$, public key, and queries type to the third party. As the user’s location coordinates are perturbed by matrix $X$, to guarantee the utility, the map should also be perturbed by the same matrix, after which the relative distances between user and POIs can be preserved. Both the service provider and the third party hold partial location information about the user. None of them can obtain the
location of the user with only partial information.

c. Once the third party receives the map sanitization query, it transforms the region $R$'s map with queried POI type using $X$. The relative distances between the records and user can be maintained because they have been transferred using the same transition matrix. The perturbed map is anonymized, after which it can be sent to the service provider. Fig. 5.3 shows an example of the map sanitization results.

<table>
<thead>
<tr>
<th>Original map</th>
<th>Perturbed map with anonymization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Restaurant</strong></td>
<td><strong>Restaurant</strong></td>
</tr>
<tr>
<td>Sofia</td>
<td>1</td>
</tr>
<tr>
<td>(2,5)</td>
<td>(10.5019,6.2377)</td>
</tr>
<tr>
<td>Thai Yim</td>
<td>2</td>
</tr>
<tr>
<td>(4,3)</td>
<td>(11.3109,1.8529)</td>
</tr>
<tr>
<td>Taste dumpling</td>
<td>3</td>
</tr>
<tr>
<td>(1,7)</td>
<td>(11.4821,9.9476)</td>
</tr>
<tr>
<td>Geppetto's</td>
<td>4</td>
</tr>
<tr>
<td>(3,8)</td>
<td>(16.4452,10.1153)</td>
</tr>
<tr>
<td>Chpaati</td>
<td>5</td>
</tr>
<tr>
<td>(6,5)</td>
<td>(17.6587,3.5381)</td>
</tr>
</tbody>
</table>

Figure 5.3: Map sanitization

Assume the user queries the nearby restaurants. The table on the left shows the sampled original map. Each restaurant has a geographic location coordinate. Assume the transition matrix is 2-dimensional, denoted by $X = \begin{bmatrix} 1.7892 & -0.6749 \\ 1.3847 & 1.5175 \end{bmatrix}$, resulting in perturbed location vectors shown in the right-hand side table. The location of the restaurants are totally changed and the names are replace by the meaningless number.

d. The service provider searches for POIs based on the perturbed locations and send them to the third party together with another $K - 1$ sets of POIs within safe region, where $K$ is the number of POI set. As the POIs found by the service provider are the meaningless number with the perturbed location, the third party will need to find the corresponding POIs by searching a mapping list. To avoid the third party
gets the queried POIs by the user, the POIs calculated by the service provider are protected by $k$-anonymity technology shown in Fig. 5.4. The left hand side table is the searched POIs, denote as set 1. the right hand side table is the $K$-anonymized POI sets. There are totally $K$ sets. The right hand side figure shows that the $K - 1$ sets of POIs are randomly chosen from the safe region. Therefore, the third party has no idea which sets are the queried POIs. The value of $K$ is determined by the user. The probability of identify the user queried POIs is $1/K$. A bigger value of $K$ means harder to identify the returned POIs for the third party. Users can choose it according to their own privacy consideration. The upper bound of $K$ value is controlled under 20, which is big enough to prevent the POIs being identified. $K$ sets of real POIs information will be returned to the service provider. In order to prevent the service provider from obtaining the accurate POIs information, the returned $K$ sets of POIs are encrypted using user’s public key. And the queried POIs are filtered according to the POI ID and returned to the user by the service provider.

The service provider is ”blind” during the whole POIs query process. As shown in Table. 5.2, it knows nothing about the user’s query. The service provider only receives a location vector. Therefore, it has no idea the user’s exact location. During
the searching POIs process, the service provider just accesses to the anonymized meaningless records. Therefore, it has no idea what the type of these POIs and which POIs these records refer to. For the third party, it needs to transform the map according to the user’s requirement. Therefore, it knows the user’s query type, such as restaurants and cinema. However, the third party has no idea the user’s location and returned POIs as well, because the third party never receives any location information of the user and only access the $k$ anonymized POIs. Besides, using this privacy framework, the service provider can return very accurate POIs due to the Johnson-Lindenstrauss transform’s ability to maintain relative Euclidean distances between records.

Table 5.2: Information access

<table>
<thead>
<tr>
<th></th>
<th>Queried POI type</th>
<th>User’s location</th>
<th>Returned POIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service provider</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Third party</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>User</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

In the proposed privacy preservation method, the third party needs to encrypt the received POIs. Assume the number of queried POI is $k$, the third party needs to compute $Kk$ encryptions, where $K$ is the number of sets of POIs. In addition, the user needs to compute 1 decryption. Therefore, the total computation complexity for encryption is $O(Kk + 1)$ exp. As the number $k$ is small, the computation complexity caused by the encryption part is acceptable. The proposed privacy framework supports the two most popular spatial queries: k-nearest POIs queries and range queries. In Section 5.3, we present the details of how these queries are processed in our privacy model.

111
5.3 Privacy Protection Algorithms for Two Basic POI queries

5.3.1 K-nearest POI Queries

Consider an application that the user wants to query the $k$-nearest POIs around his location. The query can be 'where is the $k$ nearest restaurants'.

$k$-nearest POIs query generation

It has two main operations for user: location perturbation and query generation. The specific steps are shown in Algorithm 3.

Algorithm 3 k-nearest POIs query (User)

Require: user ID, $u_i$’s location $l_t$, POI number $k$, POI set number $K$, dimension $m$.

Ensure: $k$-nearest POIs $POIs$.

1: Perturb user’s location information using location perturbation method, get the perturbed location $l_p$.
2: Query LBS provider POIs with perturbed location $l_p$.
3: Generate a key pair $sk$ and $pk$ using RSA algorithm.
4: Choose a safe region $R$.
5: Query the third party map sanitization with user ID, transformation matrix $X$, safe region, queried POI type, and public key.
6: return Obtain $POIs$ from the service provider
• **Location perturbation.** The user’s location $l_t$ can be transferred into a meaningless location vector $l_p$ using a Johnson transformation matrix (Step 1).

$$l_p \leftarrow f(l_t, m),$$  \hspace{1cm} (5.3.1)

where $m$ is the dimension of the new location vector, which is specified by users. $f$ is the function of the transformation.

• **Query generation.** The query sent to the service provider is in the following form:

$$\text{query} \leftarrow \langle \text{userID}, k, l_p, K \rangle,$$  \hspace{1cm} (5.3.2)

where $k$ is the number of queried POIs, and $K$ refers to the number of POIs sets.

User query k-nearest POIs using perturbed location $l_p$ (Step 2). As $l_p$ is a $1 \times m$ vector, the service provider cannot find any relationship between this vector and the user’s location without the transition matrix $X$.

The user generates a key pair in Step 3, which is used to encrypt the POI information. A safe region $R$ is chosen by the user in Step 4. The safe region means the user does not mind other people knows that he/she is in region $R$. While defining a safe region helps the third party transformed less data, which increase the algorithm efficiency. The query sent to the third party is in the following form:

$$\text{query} \leftarrow \langle \text{userID}, X, R, \text{POI-type}, pk \rangle.$$  \hspace{1cm} (5.3.3)

The public key together with the transition matrix, safe region, and POI type are sent to the third party for map sanitization in Step 5. The user ID is also
included in the query, such that the service provider can find the corresponding perturbed map for each specific query. At the end, the user can get the queried POIs from service provider shown in Step 6.

To guarantee utility, the map should be perturbed by the same transition matrix as the user’s location. However, if the service provider knows the perturbed location vector and the transition matrix at the same time, the true location coordinates of the user would be disclosed. Therefore, this task is assigned to the third party.

Map sanitization

After the third party gets a query from the user, it starts to sanitize the map. Algorithm 4 shows the process.

First, the third party samples a small map according to the received user’s safe region $R$ and the queried POI type in Step 1 and perturbs it using received transition matrix $X$ in Step 2. To avoid the service provider inferring the real POIs information, the transformed POIs are anonymized by replacing the identifiers by meaningless numbers in Step 3. The sanitized map is sent to the service provider for $k$-nearest POIs calculation in Step 4. After that, the third party will get $K$ sets of $k$ nearest POIs from the service provider. In step 5, the service provider finds the POIs real information according to the mapping function $f$, and encrypt it using encrypt algorithm in Step 6. The encrypted POIs are in the format of $<ID, Enc(POIs)>$. At the end, the encrypted real POIs information is sent back to the service provider.
Algorithm 4 Map sanitization (Third Party)

Require: map $\mathcal{M}$, user ID, transformation matrix $X$, queried POI type, safe region $R$

Ensure: POIs.

1: Sample a small map $M$ with same POI type required in the query in region $R$ from the map $\mathcal{M}$.
2: Perturb the map $M$. $f : \hat{M} \leftarrow MX$,
3: Anonymize the perturbed map $\hat{M}$. $\hat{M}_a \leftarrow \hat{M}$.
4: Send the map $\hat{M}_a$ to LBS provider with the user ID.

After getting the $k$ anonymized POI sets $POIs_a$ from service provider:

5: Find the real POIs information according to the mapping function $f$.
6: Encrypt the POIs information: $Enc(POIs) \leftarrow POIs$, and send the encrypted POIs back to the service provider.
Search nearby POIs

After receives the user’s POI query and the sanitized map, the service provider searches the nearby POIs blindly.

**Algorithm 5 Response POI Queries (LBS Server)**

**Require:** User ID, $u_i$’s perturbed location $l_p$, user’ query $Q$ anonymized perturbed map $\hat{M}_a$.

**Ensure:** $POIs_a$.

1. Find the corresponding map $\hat{M}_a$ according to the user ID.
2. for $i = 1$ to length($\hat{M}_a$) do
3. \hspace{1em} $dis(i) \leftarrow$ Euclidean distance between $l_p$ and $M_a(i)$;
4. end for
5. $POIs_a \leftarrow$ find the POIs corresponding to $k$ smallest $dis$;
6. Send $POIs_a$ to the third party;
7. $POIs_a \leftarrow$ find other $K$ sets of $k$ closest POIs randomly within the map.
8. Send $POIs_a$ to the third party.

*After getting the encrypted accurate POI information from the third party*

9. Filter out the queried POI information according to the ID and return them back to the user.
10. return $POIs_a$

Algorithm 5 shows how to find the k-nearest POIs based on the perturbed information. First, the perturbed map $\hat{M}_a$ is searched according to the user’s ID if there are multiple query users at the same time. Then, the distances between the perturbed
locations of the POIs and the user’s perturbed location are calculated in Steps 2 to Step 4. The $k$-closest POIs and another $K$ sets of $k$ closest POIs are chosen in Step 5 and 6 separately. All of them are sent to the third party in such way, the third party has no idea the returned POIs to the user and cannot infer the user’s exact location. After getting the encrypted accurate POI information, the LBS server will filter out the records corresponding to the POIs obtained in Step 5 and return them to the user. The user can get the accurate POI information by decrypting it using the private key.

Although the server has no idea where the user is, the $k$-nearest POIs can still be found accurately. Fig. 5.5 provides an example. The blue dots are the true locations of restaurants around user $u$. The red dots are the perturbed restaurant locations. The different restaurants are denoted as $r_i$ for simplicity. Although the geographic positions are changed after the transformation, the relative distances are essentially unchanged. For example, in the original map, the two nearest restaurants are $r_3$ and $r_4$, while in the perturbed map, we reach the same conclusion.
5.3.2 Range Queries

An example of a range query is 'List all restaurants within 100m of the user.' In this section, we present how the proposed privacy framework to process range query.

Range query generation

A privacy-preserving range query process is similar to $k$-NN query. However, the queried radius $r$ cannot be used on the perturbed map directly because even though the relative distance is essentially the same after the transformation, the actual distance between any two locations has changed. Therefore, directly using the radius $r$ would introduce a large error. Fig. 5.6 shows an example.

![Figure 5.6: JL transform for range queries with radius $r$](image)

Assume a user queries restaurants within a radius $r$. In the original dataset, there are four restaurants $(r_1,r_3,r_4,r_5)$. If the server calculates the POIs using $r$ on the perturbed dataset, we can see only $r_4$ would be returned to the user, which is wildly inaccurate. Therefore we need to find the mapping $f : r \rightarrow \hat{r}$, to make
the relationship between $\hat{r}$ and the perturbed locations relatively consistent with the relationship between $r$ and the original locations.

Algorithm 6 Range queries (user)

Require: user ID, $u_i$’s location $l_t(x,y)$, range $r$, dimension $m$, transformation matrix $X$, queried POI type, safe region $R$, $K$.

Ensure: POIs in range $r$ $POI_a$.

1: $l_p \leftarrow f(l_t, m)$;
2: $s_i = (x + xcos\theta_i, y + ysin\theta_i)$;
3: $\hat{s}_i \leftarrow f(s_i, m)$;
4: $\hat{r} = \frac{\sum d_{s_i,l_p}}{|s_i|}$;
5: Query$_1 \leftarrow \langle userID, \hat{r}, l_p, K \rangle$;
6: Generate a key pair $sk$ and $pk$;
7: Query$_2 \leftarrow \langle userID, X, R, POI-type, pk \rangle$
8: LBS Provider $\leftarrow$ Query$_1$;
9: Third Party $\leftarrow$ Query$_2$;

Algorithm 6 shows the details. First, similar to the $k$-nearest POI queries, the user’s location coordinates $l_t(x,y)$ are perturbed to a location vector $l_p(c_1, c_2, ..., c_m)$ using location perturbation method in Step 1. To construct the effective radius $\hat{r}$ for the perturbed location, few points are chosen randomly from the edge of the queried range in Step 2. Where $\theta \in [0, 2\pi]$, to make the result more accurate, we choose more than 4 points. Identically, we get the perturbed value set $\hat{s}$ by applying location perturbation algorithm again in Step 3. In Step 4 calculates the average distance between the perturbed location $l_p$ and the points in set $\hat{s}$, which is the perturbed radius $\hat{r}$. The first query is generated in Step 5, which includes the user’s
ID, perturbed radius $\hat{r}$, and perturbed location. The user sends the range query with this perturbed radius $\hat{r}$ to the service provider in Step 6. Step 6 generates a key pair for POI information encryption and the second query is generated in Steps 7. The second query is a map sanitization query, which includes user ID, transition matrix, safe region, POI type, public key. These two queries are sent to the service provider and the third party separately in Step 8 and Step 9 respectively.

**Map sanitization**

The third party perturbs the map using the same method as in Algorithm 4 for the $k$-nearest POI queries. A smaller map is sampled according to the user’s safe region and query type. Then, perturb it using received transition matrix and send it back to the service provider for POI searching. The specific steps are shown in Section 5.3.1.

**Search nearby POIs**

The service provider processes the range query with Algorithm 7.

As shown in Algorithm 7, the distances between the user and all the POIs are calculated in Steps 1 to 2. As long as the distance $d(i) \leq \hat{r}$, the location is added to the set $POIs_a$ in Step 4. Another $K - 1$ POIs are randomly selected within perturbed map $\hat{M}_a(i)$ in Step 7, and the corresponding nearby POIs are calculated in Step 8. All of the POIs are sent to the third party. After that, the user can get the queried POIs by the same way for $k$-nearest queries. Fig. 5.7 shows an example of utility maintenance. After perturbation, the radius $\hat{r}$ accurately includes the POIs queried by the user.
Algorithm 7 Response range query (LBS server)

Require: User ID, \( u_i \)'s perturbed location \( l_p \), radius \( \hat{r} \) anonymized perturbed map \( \hat{M}_a \).

Ensure: anonymized POI \( POIs_a \).

1: \textbf{for } \( i = 1 \) to length\( (\hat{M}_a) \) \textbf{do}
2: \( dis(i) \leftarrow \text{Euclidean distance between } l_p \text{ and } \hat{M}_a(i); \)
3: \textbf{if } \( d(i) < \hat{r} \) \textbf{then}
4: \( POIs_a = POIs_a \cup ID(\hat{M}_a(i)); \)
5: \textbf{end if}
6: \textbf{end for}
7: \( P_s = \{P_1, P_2, ..., P_K\}, \text{ where } P_i \in \hat{M}_a; \)
8: \text{Repeat step 1 to 5 for each } P_i;
9: \text{Third Party } \leftarrow \text{POIs}_a;
10: \textbf{return } POIs_a

Figure 5.7: JL transform for range queries with radius \( \hat{r} \)
5.4 Privacy and Utility

5.4.1 Privacy Analysis

Theorem 9. Algorithm 2 guarantees $\epsilon$-differential privacy.

Proof. We prove that the output of the perturbation is indistinguishable regardless of the input location. That is

$$Pr[\mathcal{M}(L) \in \Omega] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(L') \in \Omega].$$ (5.4.1)

We observe that the perturbed location vector $L_p$ is composed of $m$ identically distributed variable. Each variable is created by multiplying the true location coordinate $L_t$ with a vector $M_c \in \mathbb{R}^2$. Therefore, we proof theorem 9 by showing that each variable of the output satisfies 5.4.1.

As the entries of vector $M_c$ are sampled from iid Gaussian distribution $N(0, \sigma^2)$, vector $M_c$ follows the multi-dimensional Gaussian distribution $N(0, \Sigma)$, where $\Sigma = \begin{bmatrix} \sigma^2 & \sigma^2 \\ \sigma^2 & \sigma^2 \end{bmatrix}$. According to the linear combination property of multi-dimensional Gaussian distribution, the transformed variable $L_{pc} \sim N(0, L_t\Sigma L_t^T)$. Let $L_t = (x, y)$, therefore, $L_t\Sigma L_t^T = (x^2 + y^2)\sigma^2$. Denoted by $\lambda^2 = L_t\Sigma L_t^T$, then, $L_{pc} \sim N(0, \lambda^2)$. Let $L_t(x', y')$ be the neighbouring dataset, $L'_{pc} \sim N(0, \lambda^2)$ We have

$$\frac{Pr[\mathcal{M}(L) \in \Omega]}{Pr[\mathcal{M}(L') \in \Omega]} = \frac{PDF_{L}(x)}{PDF_{L'}(x)} = \frac{\frac{1}{\sqrt{2\pi}\lambda} \exp\left(-\frac{x^2}{2\lambda^2}\right)}{\frac{1}{\sqrt{2\pi}\lambda'} \exp\left(-\frac{x^2}{2\lambda'^2}\right)} = \frac{\lambda'}{\lambda} \exp\left(\frac{x^2}{2}\left(\frac{1}{\lambda'^2} - \frac{1}{\lambda^2}\right)\right).$$ (5.4.2)
When variance $\lambda^2 > \lambda'^2$,

$$\frac{Pr[\mathcal{M}(L) \in \Omega]}{Pr[\mathcal{M}(L') \in \Omega]} \geq \frac{\lambda'}{\lambda} = \frac{\lambda - \Delta}{\lambda} = 1 - \frac{\Delta}{\lambda} . \quad (5.4.3)$$

$\epsilon_0 = -\ln(1 - \frac{\Delta}{\lambda})$.

When variance $\lambda^2 < \lambda'^2$,

$$\frac{Pr[\mathcal{M}(L) \in \Omega]}{Pr[\mathcal{M}(L') \in \Omega]} \geq \frac{\lambda'}{\lambda} = \frac{\lambda + \Delta}{\lambda} = 1 + \frac{\Delta}{\lambda} . \quad (5.4.4)$$

$\epsilon_1 = \ln(1 + \frac{\Delta}{\lambda})$.

As

$$\epsilon_1 - \epsilon_0 = \ln(1 + \frac{\Delta}{\lambda}) + \ln(1 - \frac{\Delta}{\lambda})$$

$$= \ln(\frac{\lambda^2 - \Delta^2}{\lambda^2}) < 0 . \quad (5.4.5)$$

$\epsilon_0 > \epsilon_1$, Therefore,

$$e^{-\epsilon_0} \leq \frac{Pr[\mathcal{M}(L) \in \Omega]}{Pr[\mathcal{M}(L') \in \Omega]} \leq e^{\epsilon_0} . \quad (5.4.6)$$

Let $\epsilon = \epsilon_0$, therefore, the proposed method satisfies $\epsilon$-differential privacy, where $\epsilon = -\ln(1 - \frac{\Delta}{\lambda})$.

\[\square\]

### 5.4.2 Utility Analysis

For POI queries, the returned POIs are based on the relative distance between the users and the nearby POIs. Therefore, the utility is evaluated by comparing whether the relative distances between the users and the POIs have changed.

As shown in Fig. 5.8, given a user $u$, and two POIs $a$ and $b$ around $u$, assume $\| u - b \| = t \| u - a \|$ and $t > 1$, where $\| u - a \|$ is the Euclidean distance $d_{u,a}$
between $u$ and $a$. After the Johnson Lindenstrauss transform, the distances between $a$ and $u$ and $b$ and $u$ are $\| f(u) - f(a) \|$ and $\| f(u) - f(b) \|$ separately. Then the probability of causing an error is $Pr[\text{error}] = Pr[\| f(u) - f(a) \| \geq \| f(u) - f(b) \|]$.

According to Lemma 3.3.1,

$$Pr[\| f(u) - f(a) \| \geq \| f(u) - f(b) \|]$$

$$\leq Pr[\| u - a \| \sqrt{1 + \delta} \geq \| u - b \| \sqrt{1 - \delta}]$$

$$= Pr[\| u - b \| \leq \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}}]$$

$$= Pr[t \leq \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}}] \quad (5.4.7)$$

$$= \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}} - 1$$

$$= \sqrt{3}$$

As $0 < \delta < \frac{1}{2}$, therefore, $1 < \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}} < \sqrt{3}$. Variable $\delta$ is defined by the user, therefore, the value of $\delta$ can be chosen uniformly from $(1, \sqrt{3})$. So, we assume the variable $\delta$ obeys uniform distribution.

Case 1: For a given $\delta$, $1 < \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}} < \sqrt{3}$. As $t > 1$, when $t \geq \sqrt{3}$, $Pr[t \leq \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}}] = 0$. When $t < \sqrt{3}$, we assume $t$ obey uniform distribution in $(1, \sqrt{3})$, which is reasonable, because $t$ is the ratio of distances between two POIs to the user. It can be an arbitrary value in the range of $(1, \sqrt{3})$ according to different POI locations selection. Therefore, $Pr[t \leq \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}}] = \frac{\sqrt{3}}{\sqrt{\delta}}$. Equation 5.4.7 can be written as

$$Pr[t \leq \frac{\sqrt{1 + \delta}}{\sqrt{1 - \delta}}] = \frac{\sqrt{3}}{\sqrt{\delta}}.$$
\[
Pr[error] \leq \begin{cases} 
0 & t \geq \sqrt{3} \\
\frac{\sqrt{1+\delta}}{\sqrt{3}\sqrt{1-\delta}} - \frac{\sqrt{3}}{3} & t < \sqrt{3}
\end{cases}
\]

As
\[
\frac{\partial\left(\frac{\sqrt{1+\delta}}{\sqrt{3}\sqrt{1-\delta}} - \frac{\sqrt{3}}{3}\right)}{\partial \delta} = \frac{\sqrt{3}}{6} \frac{\sqrt{1+\delta} + \sqrt{1-\delta}}{1-\delta} > 0, \quad (5.4.9)
\]

Therefore, for a given \( \delta \), when \( t \geq \sqrt{3} \), the probability of cause an error is 0; when \( t < \sqrt{3} \), the greater the \( \delta \) is, the higher the error probability is. According to the Johnson Lindenstrauss Lemma, \( m = \Omega(\log(n)/\delta^2) \). We can infer that for a fixed \( n \), \( m \sim \frac{1}{\delta^2} \). Therefore, we can draw the conclusion that greater value of \( m \) helps to reduce the error probability.

Case 2: For a given \( t \), where \( t > 1 \). Equation 5.4.10 can be written as

\[
Pr[\| f(u) - f(a) \| \geq \| f(u) - f(b) \|]
\leq Pr[t \leq \frac{\sqrt{1+\delta}}{\sqrt{1-\delta}}]
= Pr[\delta \geq \frac{t^2 - 1}{t^2 + 1}], \quad (5.4.10)
\]

As \( 0 < \delta < \frac{1}{2} \), when \( \frac{t^2 - 1}{t^2 + 1} \geq \frac{1}{2} \), that is \( t \geq \sqrt{3} \), \( Pr[\delta \geq \frac{t^2 - 1}{t^2 + 1}] = 0 \). When \( 1 < t < \sqrt{3} \), \( Pr[\delta \geq \frac{t^2 - 1}{t^2 + 1}] = \frac{3-t^2}{t^2+1} \). Equation 5.4.10 can be written as

\[
Pr[error] \leq \begin{cases} 
0 & t \geq \sqrt{3} \\
\frac{3-t^2}{t^2+1} & t < \sqrt{3}
\end{cases}
\]

As \( \frac{\partial^2 \frac{3-t^2}{t^2+1}}{\partial t} = -\frac{8t}{(t^2+1)^2} < 0 \), when \( t < \sqrt{3} \), the error probability is monotonically decreasing with the increasing of \( t \). Which means the larger the proportional distance between two POIs, the higher the accuracy achieved.
Overall, when $t \geq \sqrt{3}$, the Johnson-Lindenstrauss transform cause no error. When $t \leq \sqrt{3}$, the greater the $m$ and the greater the $t$, the smaller the error rate.

5.5 Evaluation and Discussion

We evaluated the performance of our privacy framework through an extensive set of experiments. First, we present the experiment settings, and then discuss the experimental results.

5.5.1 Experimental Setup

Datasets. We used two real-world datasets. SimpleGeo Places dataset [1] and Yelp business dataset [2]. We extracted 8275 business entries in the area of Sydney from SimpleGeo dataset, and 22830 business entries in Las Vegas from Yelp dataset.

Metrics. The effectiveness of the proposed method was evaluated by comparing the similarities between the result sets. We evaluated the accuracy of our method in terms of displacement, resemblance [72] and recall.

1. Resemblance. Let $P = \{p_1, p_2, ... p_k\}$ be the POI set retrieved by the POI query, relative to the true location $l_t$ of the user $u$, and $P' = \{p'_1, p'_2, ... p'_k\}$ be the retrieved POI set based on the perturbed location. Resemblance measures the fraction of POIs in the actual result set that is included in the approximated
result set.

\[ \text{Resemblance} = \frac{|P \cap P'|}{|P|}, \quad (5.5.1) \]

where \(|P|\) is the size of the set \(P\).

2. **Displacement.** Displacement measures how closely \(P\) is measured by \(P'\) on average. It shows the average difference between the real POIs distance across the mismatched POIs on \(k - NN\) query.

\[ \text{Displacement} = \frac{\sum_{i=1}^{k} \| l_t - p_i \| - \sum_{i=1}^{k} \| l_t - p'_i \|}{|P|}, \quad (5.5.2) \]

where \(\|\cdot\|\) is the Euclidean distance between the true location of the user and the location of the POI.

3. **Recall.** We use recall to evaluate the proportion of relevant POIs retrieved on range query.

\[ \text{Recall} = \text{Resemblance} = \frac{|P \cap P'|}{|P'|}, \quad (5.5.3) \]

where the length of \(P\) can be different from \(P'\).

**Experimental Setup.** We used 1000 users to retrieve the \(k\)-nearest POIs and POIs in a region with an \(r\) radius corresponding to the true and perturbed locations. For the SimpleGeo dataset, we choose Restaurant and Shopping locations as interesting POIs and 1000 users were chosen randomly from the Professionals attribute. For the Yelp dataset, we choose Nightlife and Beauty & Spas as interesting POIs and 1000 users were chosen randomly from Restaurants. All of these query strings reflect different POI densities. Given the proposed method is based on a random mapping, we repeat each experiment 20 times and used the average to ensure accuracy. All algorithms
Table 5.3: Parameter settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>dimension of transition matrix</td>
<td>$2 \sim 50$</td>
<td>10</td>
</tr>
<tr>
<td>$k$</td>
<td>number of queried POIs</td>
<td>$1 \sim 50$</td>
<td>20</td>
</tr>
<tr>
<td>$r$</td>
<td>radius of range query</td>
<td>$100 \sim 1000$</td>
<td>400</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>privacy budget</td>
<td>–</td>
<td>0.5</td>
</tr>
<tr>
<td>$R$</td>
<td>privacy protection radius</td>
<td>–</td>
<td>2km</td>
</tr>
</tbody>
</table>

were implemented in Matlab on a PC with 2.7 GHz Intel Core i5 Processor and 8 GB Memory. Table 6.2 shows the parameters used.

We compared our method to two other differential privacy-based methods with a client-server structure, as no other similar technique applying a semi-trusted third party.

1. **The Geo-indistinguishability method.** This concept was proposed by E. A. Miguel et al. [11] in 2013. It states that for any $x$ and $x'$ within radius $r$, the distance $d(K(x), K(x'))$ between the corresponding distributions should at most be $l$, where the mechanism $K$ is a probabilistic function for selecting a reported value, and $l = \epsilon r$. One idea proposed by K. Chatzikokolakis et al. [21] for achieving geo-indistinguishability is that whenever the actual location is $x_0$, instead, a point $x$ is randomly generated instead, according to the planar Laplace noise function.

2. **A simple Laplace method.** In this approach, the user’s location is perturbed by adding Laplace noise to the $x$ and $y$ coordinates independently. The noise is determined by the privacy parameter $\epsilon$ and sensitivity $s$, where the sensitivity
equals the radius $r$, so that the user cannot be distinguished with from other users within $r$.

To make these methods comparable, we assumed the neighbor location $l'(x', y')$ in our method was within the radius $r$ of true location $l(x, y)$. Then, $\Delta = |\sqrt{x'^2 + y'^2} - \sqrt{x^2 + y^2}| \ll \lambda$. Therefore, $\epsilon \ll 1$. We made $\epsilon = 0.5$ for all methods. The maximum radius calculated in paper [21] is $2km$; therefore, $r = 2km$ in the simple differential privacy method. Our method and the two other methods are denoted as JL, GEO, and DP, respectively.

5.5.2 Performance of the Proposed Method

$k$-NN query

We examined the performance of the proposed method in relation to the number of queried POIs $k$ for $k$-NN queries in terms of resemblance and displacement. We varied the number of queried POIs between 1 and 50 in Step 1 on both datasets.

The resemblance values corresponding to different values of $k$ for both the SimpleGeo and Yelp datasets are shown in Figs. 5.9a, 5.9b, 5.9c, and 5.9d. It is clear that JL significantly outperformed GEO and DP in all configurations. As shown in Fig. 5.9a, when $k = 5$ and nearby Restaurants were queried from the SimpleGeo dataset, GEO achieved a resemblance of 0.2560, DP achieved a resemblance of 0.0850, and JL achieved a resemblance of 0.8582, which outperformed GEO by 60% and DP by 77%. A similar result was also observed when measuring resemblance on the Yelp dataset. Obviously, GEO and DP performed very badly when $k < 10$ on both datasets, but they were greatly improved by increasing the value of $k$. However, the number of queried POIs had little effect on JL in terms of the resemblance metric. The JL
Figure 5.9: $k$-NN query performance
method always performed well, even when $k$ was small, because its utility is related to the dimensions $m$ and $t$, as shown in Section 5.4.2. For a fixed $m$, a larger $t$ helped maintain a higher accuracy and was not affected by the size of $k$.

The displacement values corresponding to different values of $k$ on both the SimpleGeo and Yelp datasets are shown in Fig. 5.9e, 5.9f, 5.10g, and 5.10h. We observed that JL had a much lower displacement value than both GEO and DP when querying all attributes on the two datasets. Additionally, we observed that the size of $k$ did not affect the performance of JL significantly in terms of the displacement metric, while GEO and DP increased faster when $k$ decreased from 20 to 1. For example, in Fig. 5.13a, the displacement JL achieved across all values of $k$ was under $20m$. However, when $k = 5$, GEO and DP achieved displacements of 181.4295 and 642.0554, respectively, much larger than the displacement values of $83.3879m$ for GEO and $367.0430m$ for DP when $k = 20$. A similar observation was found in Fig. 5.9f, 5.10g, and 5.10h. This indicates that a smaller $k$ means a larger difference between the returned POIs based on the true location and the false location for both the GEO
Figure 5.11: Range query performance
Figure 5.12: Range query performance
and DP methods. JL had no such problem. The same observation was made when considering performance in terms of the resemblance metric.

The proposed JL method has such excellent performance because it simultaneously perturbs the user’s location and the POI locations by mapping them to different dimensions instead of reporting false locations. And the relative distances between records are nearly preserved. However, any method that reports a false location will have a large margin of error when querying only a few nearby POIs, especially when the distribution of the queried POIs is compact. It is easy to deduce that the closest POI to the true location is not the closest one to the false location.

**Range query**

We further examined the performance of the proposed method in relation to the radius \( r \) for range queries in terms of resemblance and recall. We varied the radius \( r \) between 100\( m \) and 1000\( m \) in steps of 100 on both datasets.

The resemblance results of the three methods on the SimpleGeo and Yelp datasets are shown in Fig. 5.11a, 5.11b, 5.11c, and 5.11d. Obviously, JL had a higher resemblance value than the other two methods-irrespective of the value of \( r \). For the JL method, the resemblance metric was stable and close to 1 even with an increasing \( r \), which indicates the dataset’s utility was not overly sacrificed, and the accuracy of the returned POIs was mostly enhanced. However, as shown in Fig. 5.11a, when \( r < 600 \), neither GEO nor DP maintained good utility. Specifically, when \( r = 800 \), GEO and DP achieved resemblances of 0.8291 and 0.2712, respectively. JL achieved a resemblance of 0.9569, an improvement of 13% and 68%, respectively. When \( r = 300 \), JL achieved a resemblance of 0.9001, outperforming GEO by 60% and DP by 84%.
These improvements by JL were observed on the Yelp dataset. As shown in Fig. 5.11d, when $r = 400$, GEO and DP achieved resemblances of 0.2560 and 0.2347, respectively. JL outperformed them by 70% and 72% with a resemblance of 0.9548. When $r = 800$, JL achieved a resemblance of 0.9521, which is an improvement of 43% and 66% over GEO and DP’s resemblances of 0.5291 and 0.2935, respectively. Figs. 5.12g, 5.12h, 5.11c, and 5.11d show the results for the SimpleGeo dataset in terms of recall. We observed that the performance of the three methods was similar to the results of the resemblance metric. JL outperformed GEO and DP across all $r$ values. JL was relatively stable compared to GEO and DP - they changed significantly when varying the radius $r$.

We attribute the reason JL performed so well to the properties of the Johnson-Lindenstrauss transform. The distances between the locations were almost fully preserved after the transformation, regardless of the queried radius $r$. However, in the GEO and DP methods, if the queried range is very small, it is possible for there to be no overlap between the true POIs and the perturbed POIs. So, the JL method outperforms GEO and DP significantly when $r$ is small.

5.5.3 The Impact of POI Density

To examine the impact of POI density on the three methods for both $k$-NN queries and range queries. We choose three attributes that reflect different POI densities in the SimpleGeo and Yelp datasets.
Figure 5.13: The effect of density on the \(k\)-NN and range queries
Figure 5.14: The effect of density on the $k$-NN and range queries

$k$-NN queries

The resemblance and displacement values corresponding to the three methods in two datasets by querying three different $k$-NN POIs when $k = 20$ are shown in Figs. 5.13a - 5.13d. The POI density had little effect on JL; however, it affected the GEO and DP significantly. As shown in Fig. 5.13a, JL’s resemblance value increased slightly when increasing the POI density, while the resemblance value decreased significantly for the GEO and DP methods. Specifically, in Fig. 5.13a, JL achieved approximately 0.83 resemblance for all POI queries on the SimpleGeo dataset. While, when querying Restaurant, the resemblance achieved by GEO was 0.6730. When querying nearby Shopping and Health Service, GEO achieved resemblances of 0.1960 and 0.1159, respectively, a decrease of 48% and 8%. The resemblance achieved by DP decreased by approximately 30% and 5%. The resemblance results on the Yelp dataset closely resembled those on SimpleGeo, shown in Fig. 5.13c. Accordingly, the displacement decreased slightly for JL and increased significantly for both GEO and DP. As shown in
Fig. 5.13b, JL achieved displacements of 20.0431, 6.0760, and 3.9532 for Restaurant, Shopping, and Health Service, respectively. While GEO achieved 83.3879, 199.6278, and 236.7437, an increase of around 116 and 27 when increasing the POI density. DP achieved 367.0430, 670.1029, and 689.0549, an increase of 303 and 19.

Because for GEO and DP methods, when the density increased, more mismatched POIs reduced the accuracy. However, for JL method, the $k$ nearest POIs were much closer to the user as the density increased. Therefore, the proportion of the distances became slightly larger compared to low-density distribution. According to our utility analysis, accuracy increases as the distance proportionally increases between two POIs and the user.

**Range Query**

The effects of the POI density on the range query are shown in Figs. 5.13e - 5.14h. It is clear that the POI density had no obvious impacts on JL for both datasets. For example, JL achieved a resemblance of approximately 0.89 for all three different POI queries in Fig. 5.13e on SimpleGeo dataset, and achieved a recall of approximately 0.86 in Fig. 5.13f. Similar results can be observed in Fig. 5.14g and Fig. 5.14h on Yelp dataset. However, there was a different result for GEO and DP. Both resemblance and recall slight changed on SimpleGeo dataset, but had a significant reduction in the performance when increasing the POI density of the queries on Yelp dataset. For instance, the GEO method achieved a resemblance of 0.5600 when querying nearby Nightlife; however and reduced to 0.2560 when querying Beauty & and Spas. The performance was much worse when querying Restaurants, which has the largest density. The DP method had similar results. Fig. 5.14h shows a similar trend in terms
of recall.

As we mentioned, for a given $m$, the performance of JL is only related to the ratio of distances between the user and POIs $t$. The change of POI density cannot affect the JL performance significantly for range query.

Figure 5.15: The effect of $m$ on $k$-NN and range queries
5.5.4 The Impact of Transition Matrix $M$

In the proposed method, $m$ is an important parameter that determines the length of the perturbed location vector. To determine how $m$ contributed to the results, we changed the value of $m$ from 2 to 50 in steps of 4 on both the SimpleGeo and Yelp datasets.

The results on SimpleGeo and Yelp for the $k$-NN queries are shown in Fig. 5.15. JL’s performance was greatly improved by increasing the value of $m$. When the dimension of the transition matrix increased, resemblance showed an upward trend on both datasets. As shown in Fig. 5.15a, when $m = 2$ for SimpleGeo querying nearby POIs resulted in a resemblance of 0.7. When $m$ is increased, the resemblance increases rapidly. When $m = 14$, the resemblance = 0.9. Performance started to level off with an $m$ higher than 14. A similar result is observed when querying nearby POIs on the Yelp dataset. As shown in Fig. 5.15b, when $m$ is in the range of [2, 14], the changes in resemblance are significant. The query result is much more accurate when the dimensions of the transition matrix are much higher. When it reaches a threshold, performance is not expected to improve dramatically. Fig. 5.15d and Fig. 5.15d show the results of the range queries on two datasets. We can observe that the increasing parameter $m$ has a positive effect on performance. For instance, in Fig. 5.15c, when $m = 2$, resemblance is around 0.82. When $m = 10$, a resemblance of 0.91 is achieved.

The relationship between the two fixed POIs did not change, which means $t$ is invariable. According to the analysis in Section 5.4.2, when $t$ is fixed, the probability of an error decreases with the increase in the value of $m$. Therefore, when $m$ becomes larger, the error probability is reduced, and better performance is achieved.
5.5.5 Overhead Analyses

In this section, we analyse the computational cost, query process delay and transmission overhead induced by the proposed privacy framework. There are three processes may cause the delay: map sanitization, POI encryption, and POI decryption. Because in the map sanitization process, only the queried POIs in the safe region are transformed by the transition matrix, the time cost is very short that can be ignored. Therefore, we tested the computational cost of POI encryption and POI decryption under the proposed privacy framework by varying the number of queried POIs and number of dummy POI sets.

![Graph](a) Computational cost by varying k

![Graph](b) Computational cost by varying K

Figure 5.16: Computational cost of the proposed method

Fig. 5.16 shows the results. The dark blue regions in the bars indicate the computational cost at the third party for encrypting the POI information. The yellow regions in the bars indicate the computation burden at the user side for decrypting the received POI information. Therefore, the whole bars indicates the overall query response time, that is the delay induced by the proposed privacy framework.
Fig. 5.16a shows the computation time cost by varying the number of queried POIs, where $K = 3$. We can observe that the time cost is increasing with the increase of the number $k$, no matter the encrypt time or decrypt time. This is because when the number of queried POIs increases, the number of POIs needed to be encrypted and decrypted increases. Therefore, both time costs are increasing. Fig. 5.16b shows the computation time cost by varying the number of dummy POI sets, where $k = 5$. We observe that the encrypt time increases with the increase of number $K$, however, the decrypted time does not change. This is because the user only needs to decrypt the queried POIs. When the number of queried POIs does not change, the time cost caused by decryption does not change.

It’s clear that the whole query response time is very short that was controlled within few seconds. The delay induced by the proposed privacy framework is negligible compared with tradition encryption-based method. This is because in the proposed method, the encryption technology is only used to hide the POI information. It was never used to calculate or search the POI records. In addition, only few POI records are encrypted instead of the whole region. Also, there is no big transmission overhead, because only the queried POIs in a safe region and limited encrypted POIs are transmitted between the service provider and the third party.

5.6 Related Work

Numerous techniques have been provided to protect a user’s location privacy [40, 53, 83, 109]. The current main techniques for preserving privacy in LBSs are described in the following review.

Most proposed methods are designed to operate in client-server structures. In
dummy location methods, multiple false locations along with the user’s true location are sent to the LBS server such that the true location cannot be distinguished from the false locations [53]. Obfuscation techniques try to protect a user’s location by reducing the precision of the position information, reporting a region to the LBS server instead of the precise user location [164]. Beyond the cloaking method, Gutscher et al. [50] proposed a coordinate transformation method. The user performs some basic geometric operations over their positions, such as shifting and rotating, before sending it to the LBS server. Lin et al. [?] proposed the so-called Policy-Embedded Bx-tree (PEB-tree), which organizes objects based on both spatial proximity and privacy policy compatibility. Differential privacy-based perturbation methods have also been proposed. Dewri [28] proposed a method that perturbs the user’s location by adding Laplace noise to the $x$ and $y$ coordinates independently. Andre et al. [11] introduced the notion of geo-indistinguishability to formalize the problem of protecting a single user’s position. It achieves privacy by adding controlled noise to the user’s location making the user’s location indistinguishable within a radius $r$ [21].

Private information retrieval-based protocols [167] were also proposed for POI queries. This technique allows a user to retrieve a record from a database server without revealing any information about the query. However, such cryptography-based approaches rely on heavy cryptographic mechanisms, which are often computationally and communicatively expensive.

A few privacy-preserving techniques have attempted to use trusted third parties (TTP) for location-based services. The most commonly used TTP approaches rely on an anonymizer to create a spatial region that includes at least $k - 1$ other users to hide the true location [40]. Because $k$-anonymity is achieved, an adversary can only
identify a user’s true location with probability no higher than $1/k$. Papadopoulos et al. [110] employed trusted hardware to perform PIR for LBS queries. Their hardware-aided PIR technique relies on a trusted third party to set a secret key and permutate the database. However, the third party is aware of the precise user positions and is, therefore, vulnerable to misbehavior by the fully-trusted third party. Recently, Schlegel et al. [126] proposed a dynamic grid system to provide privacy-preserving continuous LBS. They introduced a semi-trusted third party, responsible for simple matching operations. However, their method is based on complicated cryptographic functions, which is, as previously mentioned, carries a heavy computational burden.

5.7 Summary

In this chapter, we have proposed a new privacy preservation framework that includes three elements: a user, a third party, and the LBS server. The third party is a semi-trusted entity based on a weak assumption that it does not collude with the LBS server. In addition, we propose a novel privacy protection scheme, based on the Johnson-Lindenstrauss transform. Moreover, the user’s exact location is perturbed by Johnson-Lindenstrauss which satisfies the definition of differential privacy, and maintains high level application utility at the same time. The combination of privacy preservation method and client-bridge-server structure ensure that none of the party is aware of the user’s exact location. Our method supports two very popular queries: k-nearest-neighbour queries and range queries. Performance is evaluated through extensive experiments, and our proposed method is further compared to two representative differential privacy-based methods. The results demonstrate that our framework provides better privacy guarantees and is more efficient in terms of
resemblance, displacement, and recall.
Chapter 6

Privacy-preserving in Crowdsensing System

6.1 Introduction

Crowdsensing as a new trend of development in the Internet of Things (IoT) takes advantage of pervasive sensor-equipped mobile devices to collect and share data. The phenomenon has given rise to numerous large scale, real-world applications, which have the power to create awareness about a specific large-scale phenomena and to ignite crowd intelligence [12], such as environment monitoring [121], traffic condition detection [97], and point-of-interest characterization [23]. In a typical crowdsensing platform, participants are registered as candidate workers. A centralized server (hereafter, the Server) selects workers to complete a data-collection task, and they are paid a reward for doing so. The selected workers then travel to a predefined location to collect the required data. However, to be able to assign tasks more efficiently, workers need to submit their exact location to the Server. Disclosing one’s location raises serious privacy concerns as the Server may not be trusted. Given a lack of privacy protection may affect worker uptake of such systems, ensuring the privacy of
the worker locations is highly desirable.

Numerous techniques have been proposed to protect the privacy of user locations, such as dummy locations, k-anonymity, obfuscation methods, and differential privacy. Of these methods, differential privacy has been widely accepted because of its ability to provide rigorous privacy protection. Differential privacy ensures that no single individual, whether included or excluded from the dataset, can significantly affect the output of a query. It is normally achieved by injecting random noise into the query results. The process has already been well-implemented for location-based queries; however, many weaknesses remain in its application to spatial crowdsensing.

The current typical solution of differential privacy protects location privacy by introducing a trusted third party [141]. The third party partitions the domain of worker locations into small cells and hides each worker in a cell. The method is based on the assumption that workers are uniformly distributed within the domain. We argue this assumption is unreasonable, unless the cell size is very small. If the cell size was large, workers would likely be seen as clusters, as aggregation is a basic feature of human society; worker locations would be distributed more realistically as communities. However, this uneven distribution would cause significant errors during the task assignment process. In addition, this existing method means the partitioning process needs to satisfy differential privacy, known as privacy spatial decomposition. Yet adding Laplace noise to each cell at each level consumes too much of the privacy budget and generates a significant volume of noise. Therefore, applying differential privacy introduces two challenges as discussed next.

The first challenge is how to more accurately measure the distance between the workers and the task. This factor is crucial for an efficient task-matching system. To
preserve privacy in current crowdsensing systems, only a noisy count of the workers in each cell can be released. Therefore, the distance between a worker and a task is normally assumed to be equal to the average distance between the task and each of the four corners of the cell. If the workers are distributed uniformly in the cell, the distance measurement would be closer to reality. Hence, we propose a privacy data release method that partitions the domain of worker locations based on worker density and ensures the distribution of workers within a cell is as uniform as possible.

The second challenge is guaranteeing the success rate of task assignment while reducing system overhead. In this chapter, system overhead refers to the distance workers must travel to complete a task and the number of workers who are notified. Under the veil of differential privacy, an exact count of workers in each cell cannot be released to the Server, so the Server cannot be sure of the exact number of workers that are notified about a task. In fact, it is possible for there to be no workers in a cell. As a result, the Server needs to allocate tasks to a large number of workers within a cell to guarantee a task assignment success rate, and this can increase system overheads. To solve this problem, we propose a two-pronged approach. First, a privacy budget is assigned to each cell when releasing the data. This reduces the noise and increases the accuracy of the released data. Second, to balance the task assignment success rate and the system overhead, the model is constructed by solving a geocast region optimization problem. The proposed method takes both travel distance and the number of notified workers into account, balancing the task assignment success rate and system overhead very well.

Overall, this chapter makes the following contributions:

1. We propose a privacy protection data release method based on worker density
that achieves differential privacy. The sanitized data is able to accurately repre-
represent the original distribution of the data, which contributes to a high task
assignment success rate.

2. We introduce a geocast region selection method, which ensures highly efficient
task assignments and adequately balances task assignment success rates with
system overheads.

The rest of the chapter is organized as follows. In Section 6.2, we introduce the
fundamentals of crowdsensing. We propose our privacy crowdsensing method and the
theoretically analyze privacy and utility in Sections 6.3 and 6.4, respectively. Section
6.5 details the results of the experiments. Section 6.6 discusses related work, and
Section 6.7 concludes the chapter.

6.2 Fundamentals of Crowdsensing

In this section, we present the basic concepts of crowdsensing, followed by a typical
privacy framework for mobile crowdsensing

6.2.1 Crowdsensing

Crowdsensing is a technique where a large group of individuals with mobile devices
equipped with sensors collectively share sensory data to measure, analyze, or infer any
any processes of common interest. Specifically, the requester posts the sensing tasks,
the workers finish the task and send it to the requester for rewards. Crowdsensing has
two models of task assignment, *Worker Selected Tasks* (WST) and *Server Assigned
Tasks* (SAT).
In WST, the server publishes the tasks and the workers autonomously select their favourite tasks. The advantage of this model is that the workers do not need to reveal their exact location information and the server does not know which tasks the workers take. That is, the server has no idea where the worker is (the worker’s exact location) and where the worker is going to (the location of the assigned task). The drawback of this model is that the server doesn’t have any control over the allocation of the tasks. The workers prefer to choose tasks based on their own objectives (e.g., choose $k$ closest tasks to reduce the travel cost), which may result in low task assignment success rate.

In SAT, the workers report their location information to the server, and the server assigns tasks to the worker according to their locations. The advantage of this model is that the server can control the process of task assignment. Not only assign the nearby tasks to the worker, but also maximize the task assignment success rate. The drawback is that the server knows both the worker’s location and the tasks assigned to the worker, which brings strong privacy concern.

### 6.2.2 Framework

Fig. 6.1 shows the private framework for spatial crowdsensing, which includes three entities: the workers, the cell service provider (CSP) and the Server.

**Workers:** The workers are the participants who are actively involved in collecting and contributing data. Workers must submit their location to the CSP, travel to the location designated for the task and collect data using their sensor-equipped device.

**CSP:** The CSP collects locations from workers and releases data in sanitized form to the Server for task assignment. The CSP has a signed agreement with the workers
Figure 6.1: A framework for private spatial crowdsensing.

through a service contract, so a trust relationship exists between the CSP and the workers.

The Server: The Server queries the CSP for a sanitized dataset once it receives a task. It then assigns the task to suitable workers, through the CSP, according to a task assignment algorithm. The algorithm helps the Server choose appropriate workers, balancing a high task assignment success rate with a low system overhead.

6.3 Mobile Crowd sensing under Differential Privacy Protection

6.3.1 Problem Definition and Assumptions

Notations

Let $D$ denote the domain of all worker location. $SD = \{c_1, c_2, ..., c_m\}$ is the spatial decomposition result of $D$. $SSD = \{r_1, r_2, ..., r_m\}$ is the sanitized version of $C$. Given a task $t$, $d_{w,t}$ represents the distance between a worker and the task, then the $p_{w,t}$ is
Further notations are detailed in Table 6.2:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{w,t}$</td>
<td>Distance between a worker and a task</td>
</tr>
<tr>
<td>$d_{c,t}$</td>
<td>Distance between a cell and a task</td>
</tr>
<tr>
<td>$d_{mtd}$</td>
<td>Maximum travel distance</td>
</tr>
<tr>
<td>$p_w$</td>
<td>Acceptance probability of a worker</td>
</tr>
<tr>
<td>$p_c$</td>
<td>Task assignment success rate with a cell</td>
</tr>
<tr>
<td>$p_{ar}$</td>
<td>Acceptance rate</td>
</tr>
<tr>
<td>$ESR$</td>
<td>Expected success rate</td>
</tr>
</tbody>
</table>

Problem definition

Consider a location privacy problem in a crowdsensing system during the process of task assignment, and we consider the server assigned tasks model where the workers need to report their locations to the server, and the server will assign the task to appropriate workers. However, the Server may be untrusted. In typical privacy-preserving crowdsensing architectures, as shown in Fig. 6.1, workers submit their location to the CSP, the CSP applies an appropriate privacy protection method, and releases sanitized statistical data to the Server. Our goal is to design a data release method that accurately represents the distribution of the workers and helps the Server efficiently match workers with tasks without compromising the privacy of their locations. In addition, we need to develop a geocast region construction method that
allows the Server choose appropriate workers based on a sanitized dataset, resulting in high task assignment success rate and a low system overhead.

Assumptions

To clarify the problem, a few assumptions are necessary. First, we assume the Server is malicious; the participants do not trust the Server. Second, we assume the CSP is trusted and will not disclose worker location information.

6.3.2 Sanitized Data Release

The basic idea of private data release is that the domain of worker locations is partitioned into small cells and *Laplace* noise is added to the count of workers in each cell to achieve a differential privacy guarantee.

Density-based partition

Previous literature assumes the worker locations are distributed uniformly, and the workers in each cell have the same acceptance rate, which is not the case in real-world scenarios. Partitioning the data domain into a uniform grid would result in sizeable errors. Therefore, we propose a recursive partitioning process based on worker density. The aim is to identify dense regions and sparse regions and make the distribution of the workers in each smaller region as near to uniform as possible. Multiple space-partitioning data structures can assist with this process. For the purposes of this chapter, we used quadtree for the partitioning as it makes a good trade off between utility and efficiency.

Traditional quadtrees recursively subdivide cells into four equal-sized subcells until
the cell satisfies a stop condition. A cell becomes a leaf node if it can no longer be divided. In a data domain, this represents a region. Fig. 6.2a shows the traditional quadtree method. Note that the midpoint is always chosen to partition the parent cell. The drawback of this method is that the partition is data-independent. Workers may be clustered together in a small area of the cell, which could reduce the accuracy of the data release. However, that problem can be solved by applying the quadtree technique in a data-dependent way, i.e., by partitioning the cells according to the density of the workers as shown in Fig. 6.2b.

First, several initial partition points in the location domain need to be selected. The differences in density between the subcells partitioned by each partition point are calculated, and the subcells with the biggest differences in density are then chosen as partitions. This process is repeated for each subcell until the stop condition is met. Algorithm 8 presents the details of this density-based partitioning process.

Algorithm 8 Density based partitioning

**Require:** Dataset $W$
**Ensure:** Spatial decomposition $SD$.

1. $SD = \phi$;
2. $cell = W$;
3. $m = \sqrt{S_{cell}/\alpha}$;
4. $n \leftarrow$ number of workers in cell;
5. if $n < 1 \parallel m <= 1$ then
6. $SD = SD \cup cell$;
7. else
8. Generate $m$ partition points randomly within domain;
9. for $i = 1$ to $m$ do
10. Subcells set $C \leftarrow$ partition cell;
11. for each cell $c_j \in C$ do
12. Calculate the workers density in cell $c_j$;
13. end for
14. Calculate $\Delta d_i = max\{den(c_j)\} - min\{den(c_j)\}$;
15. end for
16. if $max\{\Delta d\} > \beta$ then
17. Partition the cell at the point with biggest $\Delta d$ into four subcells $C = \{c_1, c_2, c_3, c_4\}$;
18. for $c_i \in C$ do
19. $cell = c_i$;
20. Go to step 3;
21. end for
22. else
23: Determine whether the cell needs to be partitioned further by calculating
   \[ m' = \lceil \sqrt{\frac{n}{\sqrt{2}}} \rceil; \]
24: if \( m' > 1 \) then
25:   Partition cell to \( m' \times m' \) subcells \( c_i, 1 \leq i \leq m; \)
26:   \[ SD = SD \cup c_i, 1 \leq i \leq m; \]
27: end if
28: end if
29: end if
30: return \( SD \)

Stop condition  The stop conditions are very important in the partitioning process, as they have an important effect on the assignment success rate. Traditional quadtrees require the data publisher to specify the height of the partitioning. It is difficult to calculate an effective height with non-uniform partitioning, so we defined three stop conditions for this scenario to improve efficiency and utility.

- If no workers exist in the cell, no further partitioning is needed as that cell cannot contribute to the task. Therefore, the cell is marked as a leaf node, as shown in Steps 3 to 6.

- If a cell is too small to be further partitioned, a stop condition is met. The parameter \( \alpha \) in Step 2 controls the area of this cell \( S_{cell} \leq \alpha^2 \). The smaller the cell, the more uniform the distribution of workers within it.

- If the distribution of workers in a cell is relatively uniform, a stop condition is also met. We use maximum density difference \( \Delta d \) to measure whether the worker location distribution is uniform.
Partition point  The selection of the partition point directly affects the results of partitioning. It decides whether the distribution of workers in each cell is uniform. Therefore, \( m \) initial partition points are randomly generated within the cell. The parameter \( m \) is decided by the area of the cell being partitioned. Step 3 shows the calculation method. The intent is to find the best partitioning point that can divide the cell into four subcells with maximum density difference from the initial partition points. Therefore, more initial partitioning points mean more accurate segmentation.

All initial partition points are denoted as \( p \in \{p_1,p_2,...,p_m\} \). The score function for selecting each partition point is evaluated by the density difference, which is calculated as follow:

\[
\Delta d(\text{cell},p) = \max_{c_i \in C} \{\text{den}(c_i)\} - \min_{c_i \in C} \{\text{den}(c_i)\}.
\]  

(6.3.1)

Steps 9 to 15 calculate all the density differences based on the partition points. The partition point \( p \) with biggest density difference is chosen as a candidate. If the biggest density difference \( \Delta d_p \) is greater than the threshold \( \beta \), the cell is partitioned at point \( p \) (Steps 16 and 17). The entire process is repeated for the partitioned subcells until no further cells can be partitioned (Steps 18 to 21). Otherwise, the cell will not be partitioned as the distribution of worker in the cell is already close to uniform. Yet even after this process, the number of workers in a cell may still be large, adding to the system overhead. Therefore, Step 23 determines whether the cells need to be further partitioned into smaller cells with fewer workers. If \( m' > 1 \), the cell is partitioned into a smaller one of equal size in Step 25 and add them to SD in Step 26.
Differential privacy data release

As previously mentioned, a noisy count of the number of workers in each cell is released to protect the privacy of worker locations. Algorithm 9 shows the details of this release.

Algorithm 9 Differential privacy data release

Require: Spatial decomposition SD
Ensure: Sanitized data SSD

1: for $c_i \in SD$ do
2:   $n_i \leftarrow$ number of workers in $c_i$;
3:   $N_i = n + \text{Laplace}(\frac{\epsilon}{\delta})$;
4: end for
5: return $SSD = \{r_1, r_2, ..., r_m\}$

First, Step 2 calculates the number of workers in each cell, then Laplace noise is added to the count in Step 3. In Step 5, the sanitized SD, say SSD is released. $r_i$ represents a region with a sanitized count of workers in the cell. According to the definition of differential privacy, whether or not a worker within a specific cell cannot be identified. Therefore, worker location privacy is preserved.

6.3.3 Task Assignment

When the server receives the sanitized data, it determines a geocast region GR to disseminate the task to the workers in GR. The goal is to reach an expected task assignment success rate, while reducing system overhead at the same time, such as the distance workers need to travel and the number of workers notified of the task.
Worker acceptance probability

The distance a worker has to travel to complete a task is an important issue to consider in task allocation because it has a significant impact on both worker acceptance probability and the task assignment success rate. Not only may workers be unwilling to accept tasks with long travel times but organizers might also have to pay higher incentives to workers who are further away. Therefore, worker acceptance probability $p_w$ as is modeled as a function of distance $d_{w,t}$, as follows:

$$p_w = f(d_{w,t}).$$

(6.3.2)

Two cases are considered. In the first case, a worker’s acceptance probability decreases linearly with an increase in the distance between her location to the task location, as shown in 6.3.3.

$$f(d_{w,t}) = \begin{cases} 
\frac{d_{mtd} - d_{w,t}}{d_{mtd}}, & d_{w,t} \leq d_{mtd} \\
0, & d_{w,t} > d_{mtd}
\end{cases}.$$  

(6.3.3)

where $d_{mtd}$ is the maximum distance that most workers will travel.

In the second case, we use the nonlinear hyperbolic tangent function [10], a non-linear function.

$$y(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$  

(6.3.4)

with the property $y \in [0, 1)$, when $x \geq 0$. The acceptance probability is defined as:

$$f(d_{w,t}) = \begin{cases} 
y(\frac{c}{d_{w,t}}), & d_{w,t} \leq d_{mtd} \\
0, & d_{w,t} > d_{mtd}
\end{cases}.$$  

(6.3.5)

where $c$ is the parameter that regulates drops in the acceptance rate with increase in the travel distance.
Assuming there are $n$ workers in a cell, the probability that workers in the cell will accept a task is

$$p_c = 1 - (1 - p_w)^n.$$  \hfill (6.3.6)

**Geocast region selection**

There are two important standards when selecting the geocast region. First, the worker’s travel distance should be short. Second, the worker acceptance rate over the geocast region should achieve the expected task assignment success rate. Although the acceptance probability is based on the travel distance, we cannot say the cell with higher acceptance probability has a shorter distance to the task location. Assume there are two split cells $A$ and $B$, and the distances between these two cells and the task location $l_t$ are $d_{a,t}$ and $d_{b,t}$, also $d_{a,t} > d_{b,t}$. If cells $A$ and $B$ contain the same number of workers, cell $B$ has a greater acceptance probability. However, if there are more workers in cell $A$, it is possible that the acceptance probability of $A$ is greater than $B$. Therefore, the problem of geocast region selection can be formalized as follows:

- The number of notified workers should be as small as possible, and the worker’s travel distance should be as short as possible.

- The acceptance probability of the geocast region should reach the expected task assignment success rate.

- The distance between the selected cell and the task should be within the maximum travel distance of the workers.
To achieve our objective, we propose the geocast region selection method shown in Algorithm 10.

As shown in Step 1, the partitioned cells are sorted in increasing order according to the distance to the task. Initially, the closest cell to the task is chosen as the first GR in Step 2. If the acceptance probability does not reach expectations, the GR continues to expand by adding the closest cell to the task from the remaining cells until the acceptance probability reaches the expected goal or the cell’s distance is beyond the maximum travel distance $d_{mtd}$, as shown in Steps 3 to 5. This method ensures the worker’s travel distance is short. However, reducing the number of notified workers requires some exploration. A cell that can satisfy the expected task assignment success rate with the best balance between distance with worker numbers needs to be found. As it is not known which users will accept the task at this stage, Step 6 estimates the travel distance as an expectation of the distance of the selected cells, while Step 7 calculates the number of notified workers. Step 8 locates the cells with an acceptance probability higher than the expected acceptance probability and with fewer workers. If such cells exist, and their distance is under the threshold $d_{mtd}$, the cell $r_i \in S$ with the shortest distance is chosen as the candidate. If the expected travel distance of $p_{r_i}d_{r_i,t} < Ed$, the candidate $r_i$ is chosen as the geocast region (Steps 9 to 16).

### 6.4 Privacy and Utility Analysis

#### 6.4.1 Privacy Analysis

User location information is preserved by hiding it in partitioned cells, and the sanitized data is released by adding $Laplace$ noise to the statistical results of each cell.
Algorithm 10 Geocast region selection

Require: SSD

Ensure: GR

1: Order $SSD = \{r_1, r_2, ..., r_n\}$, where $d_{r_1,t} \leq d_{r_2,t} \leq ... \leq d_{r_n,t}$
2: Choose $r_1$ as the initial geocast region $GR$.
3: repeat
4: Expanding $GR$ by adding the closest cell in the remaining cells one by one;
5: until $p_{or} < ES$ or $d_{r_i,t} > d_{mtd}$
6: Calculate the expectation of travel distance: $Ed = \sum_{i=1}^{m} p_{r_i} d_{r_i,t}$;
7: Calculate the number of workers in $GR$: $N = \sum_{i=1}^{m} n_{r_i}$;
8: $S \leftarrow$ find $c_i$ that $p_{r_i} > ES$, $n_{r_i} \leq N$ and $d_{r_i} \leq d_{mta}$;
9: if $S \neq \emptyset$ then
10: for $r_i \in S$ do
11: find the cell $r_i$ has the shortest distance to $l_i$;
12: end for
13: if $p_{r_i} d_{r_i,t} < Ed$ then
14: $GR = r_i$;
15: end if
16: end if
17: return $GR$
Theorem 10 shows that the proposed data release method satisfies \( \epsilon \)-differential privacy.

Theorem 10. For a given dataset \( D \), each record represents a user’s location information, and the records are independent of each other. The proposed privacy preserving method can provide \( \epsilon \)-differential privacy.

Proof. Assume the proposed method partitions the map into \( m \) disjoint cells. A set of Laplace mechanisms \( \{M_1, M_2, ..., M_m\} \) are performed on each cell, and the assigned privacy parameter for each cell is \( \epsilon_i \). Each cell satisfies \( \epsilon_i \)-differential privacy. The composite properties of the privacy budget are applied to the whole dataset to analyze the privacy guarantee, which is defined below.

Theorem 11 (Parallel Composition [93]). Suppose we have a set of privacy mechanisms \( M = \{M_1, M_2, ..., M_m\} \), and each \( M_i \) provides \( \epsilon_i \) privacy guarantee on a disjoint subset of the entire dataset, \( M \) provides \( \max(\epsilon_i) \)-differential privacy.

Theorem 11 can be used to directly analyze the privacy guarantee of the proposed method. As mentioned earlier, assume the assigned privacy parameter for each cell is \( \epsilon_i \), and the cells are disjoint and independent of each other. According to Theorem 11, the set of privacy mechanisms \( \{M_1, M_2, ..., M_m\} \) will consume the \( \max\{\epsilon_1, \epsilon_2, ..., \epsilon_m\} \) privacy budget. In the proposed method, we assign each cell the same privacy budget \( \epsilon \); therefore, the proposed method preserves \( \epsilon \)-differential privacy.

\( \square \)

6.4.2 Utility Analysis

In this section, we apply a well-known utility definition suggested by Blum et al. [17] to measure prediction accuracy.
Definition 17 ((\(\alpha, \beta\))-useful). A database access mechanism \(M\) is \((\alpha, \beta)\)-useful with respect to count query, if for every database \(D\), with a probability of at least \(1 - \beta\), the output of the mechanism \(M\) satisfies

\[
Pr[\max|\hat{M}(\text{cell}_i) - M(\text{cell}_i)| \leq \alpha] \geq 1 - \beta.
\] (6.4.1)

Theorem 12. The output error of the count query on each cell caused by the proposed method is less than \(\alpha\) with a probability of at least \(1 - \beta\). The proposed method is satisfied with \((\alpha, \beta)\)-useful when \(\alpha \leq -\frac{\text{sln}2\beta}{\epsilon}\).

Proof. The error caused by the proposed method is only the noise, and is denoted as \(\lambda\), and \(\lambda \sim \text{Laplace}(\frac{s}{\epsilon})\).

Therefore,

\[
Pr[\max|\hat{M}(\text{cell}_i) - M(\text{cell}_i)| > \alpha] = Pr[\text{Laplace}(\frac{s}{\epsilon}) > \alpha] = \int_{\alpha}^{\infty} \frac{1}{2b}e^{-\frac{x}{b}}dx.
\] (6.4.2)

Let \(\int_{\alpha}^{\infty} \frac{1}{2b}e^{-\frac{x}{b}}dx = \beta\), we have

\[
\int_{\alpha}^{\infty} e^{-\frac{x}{b}}dx = 2b\beta \\
\Rightarrow -be^{-\frac{\alpha}{b}}|_{\alpha}^{\infty} = 2b\beta \\
\Rightarrow be^{-\frac{\alpha}{b}} = 2b\beta \\
\Rightarrow -\frac{\alpha}{b} = \ln2\beta \\
\Rightarrow \alpha = -b\ln2\beta.
\] (6.4.3)

As \(b = \frac{s}{\epsilon}\), therefore, \(\alpha = -\frac{\text{sln}2\beta}{\epsilon}\). That is, when \(\alpha \leq -\frac{\text{sln}2\beta}{\epsilon}\), the error introduced by the privacy operation is controlled within \(\alpha\) with a high probability. \(\square\)
6.5 Experiment Evaluation

We evaluated the performance of our method through an extensive set of experiments. First, the experimental settings are presented, followed by a discussion of the results.

6.5.1 Experimental Setup

Dataset. We used two real-world datasets.

1. SimpleGeo Places dataset [1]. This dataset contains information on more than 20 million places in 63 countries around the world. We extracted 8275 business entries for the most populous city in Australia, Sydney. We randomly chose 1000 locations as tasks, and the rest of the locations were used as workers.

2. Yelp dataset [2]. The Yelp dataset includes a business dataset, a check-in dataset, a user dataset, and so on. We used the business dataset, which includes information about local businesses in 11 cities across 4 countries. We chose the businesses located in Las Vegas, using the restaurant locations as workers and 1000 random shopping locations as the tasks.

Metrics. The effectiveness of the proposed method can be evaluated by the success rate and efficiency of the tasks assignment. Therefore, we use the following metrics:

1. Task assignment success rate. Let $T = \{t_1, t_2, ..., t_m\}$ be a set of tasks. Each task was assigned to a group of workers to be accepted with a specific probability. Assume there are $n$ tasks to be confirmed by the workers; the task assignment success rate can be represented as follows

$$TASR = \frac{n}{|T|},$$

(6.5.1)
where $|T|$ is the number of tasks.

2. **Average travel distance.** Assume $T_s = \{t_1, t_2, ..., t_n\}$ is a successfully allocated task set, and the set $W = \{w_1, w_2, ..., w_n\}$ are the corresponding workers who performed the task. Then

$$\text{ATD} = \frac{\sum_{t_i \in T_s, w_i \in W} d(t_i, w_i)}{|T_s|},$$

(6.5.2)

where $d(t_i, w_i)$ is the Euclidean distance between the task and the worker, $|T_s|$ is the number of tasks that assigned successfully.

3. **Average notified workers.** For each task $t_i \in T$, there were $n_i$ workers are notified about the task. We calculated the average number of notified workers as follows:

$$\text{ANW} = \frac{\sum_{i=1}^{\|T\|} n_i}{\|T\|},$$

(6.5.3)

where $|T|$ is the number of tasks.

**Comparison.** We compared our method with the uniform partition method, which was proposed by To et.al. [141]. They proposed partitioning the space in sparse regions using a two-level grid by modifying the state-of-the-art adaptive grid method. All the partition processes were uniform across the data domain. We considered two scenarios within a uniform partition during the geocast region selection process. First, the cell with the closest distance to the task was preferred when expanding the geocast region. Second, the cell with the highest acceptance rate was chosen at each step in the construction of the geocast region. In the experiment, we considered both linear and nonlinear acceptance probabilities.

**Parameters.** Table 6.2 shows the parameter settings of our experiment; the default values are highlighted.
Table 6.2: Parameter Setting

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>Privacy budget</td>
<td>$0.1 - 1.0$</td>
</tr>
<tr>
<td>$MTD$</td>
<td>Maximum travel distance</td>
<td>$1km - 5km$</td>
</tr>
<tr>
<td>$ESR$</td>
<td>Expected success rate</td>
<td>$0.3 - 0.9$</td>
</tr>
</tbody>
</table>

The privacy budget is a very important parameter. It determines how much noise is added to the released dataset, which affects utility. We set the privacy budget to $\epsilon \in [0.1, 1.0]$, and show the changes in performance. The default value was 1.0. The worker’s maximum travel distance was changed from 1km to 5km, and the expected task assignment success rate was varied from 0.3 to 0.9.

6.5.2 Experiment Results

Performance by varying $\epsilon$

We examined the performance of the three methods in relation to the different privacy budgets $\epsilon$ for an assigned task in terms of ANW, ATD, and TASR. We varied the privacy budget $\epsilon$ between 0.1 and 1 on both datasets using the linear and nonlinear acceptance rates. DP-GRB refers to the proposed method. UP-GRB represents the method with uniformed partitions and a balanced geocast region construction. UP-GRS gives priority to the task assignment success rate.

Performance on ANW  Figs. 6.3a-6.3d show the results in terms of ANW. We observed that ANW decreased as the privacy budget $\epsilon$ increased for both DP-GRB and UP-GRB with a reverse trend for UP-GRS. This is because a smaller privacy
Figure 6.3: Performance by varying $\epsilon$. 
Figure 6.4: Performance by varying $\epsilon$. 
budget $\epsilon$ means more noise needs to be added to each cell; therefore, more workers need to be notified to achieve the expected success rate for both DP-GRB and UP-GRB. Correspondingly, when the privacy budget $\epsilon$ is increased, less noise needs to be added to each cell, which means more workers need to be selected to achieve a higher task assignment success rate. In addition, we observed that our method always outperformed the other two methods, which have lower ANWs in all configurations. Specifically, as shown in Figs. 6.3a and 6.3b, when $\epsilon = 0.3$, our method, with linear acceptance rates, achieved an ANW of 5.3431 and 9.2615 for the Yelp and SimpleGeo datasets, respectively, UP-GRB achieved 7.7044 and 12.5312, an increase of around 50% and 30%, respectively and UP-GRS achieved 6.3847 and 16.7064, an increase of around 20% and 80%, respectively. When $\epsilon = 0.8$, UP-GRB and UP-GRS achieved ANWs of 4.8993 and 7.7132 for the Yelp dataset and 11.0441 and 20.0814 for the SimpleGeo dataset, which are much larger than the ANW values of 3.4231 and 8.7352 of our method. A similar observation was found in Fig. 6.3c and Fig. 6.3d. This is because the distribution of workers in each cell is not uniform in the UP-GRB method; however, their acceptance rates are considered to be the same, which causes some errors. Conversely, the proposed method partitions the worker domain based on worker density, which makes worker distribution in each cell close to uniform. That helps to choose more accurate cells for task assignment. Because UP-GRS always chooses the cell that produces the highest success rate at each step, more workers are needed to achieve a higher success rate.

**Performance on ATD** Figs. 6.3e-6.4h show the change in ATD with a varied privacy budget. We observed that the ATD does not significantly increase with a reduced privacy budget in either our method or UP-GRB. However, the privacy
budget had a significant effect on ATD for UP-GRS when $\epsilon < 0.4$. This proves that the proposed GR construction method did a good job in selecting which cells to balance the assignment of tasks and system overhead. Additionally, the added noise had a significant effect on the cell selection when achieving a high ASR. The UP-GRS method had a greater ATD compared to the other two methods under all configurations. This is because the construction of UP-GRS prefers to choose the cells with a higher utility regardless of the distance to the task, as long as the task is within the worker’s maximum travel distance. In addition, we observed that the ATD value of our method was always lower than UP-GRB, which means tasks can be completed within shorter distance using our method. Specifically, our method achieved an ATD of around 1km for the Yelp dataset with a linear acceptance rate, as shown in Fig. 6.3e, and it outperformed the UP-GRB method by approximately 300m. Fig. 6.4g shows the results for the Yelp dataset with a nonlinear acceptance rate. Our method achieved an ATD of around 0.12km, while UP-GRB achieved around 0.145km, which is an increase of 250m. Figs. 6.3f and 6.3f show the results on the SimpleGeo dataset. The performance of the three methods are similar to the results for the Yelp dataset.

**Performance on TASR** The TASR values corresponding to the different methods used on both the Yelp and SimpleGeo datasets are shown in Figs. 6.4i, 6.4j, 6.4k, and 6.4l. It is clear that the TASR values achieved by our method are basically around 0.8, which is the expected success rate ($ESR$). UP-GRB achieved a TASR of around 0.9, 10% higher than expected. The TASR achieved by UP-GRS significantly increased as the privacy budget increased. Specifically, as shown in Fig. 6.4i, when $\epsilon = 0.1$, our method achieved a TASR of 0.7734, only 3% below the $ESR$, while
UP-GRB achieved a TASR of 0.9190, approximately 11% higher than the ESR. UP-GRS achieved a TASR of 0.5858, which is a decrease of about 22% compared to the ESR. When $\epsilon = 0.8$, our method and UP-GRB achieved a TASR of 0.8119 and 0.9335, respectively, which is slightly greater than the TASR achieved by both methods when $\epsilon = 0.1$. This indicates that a greater privacy budget means a higher task assignment success rate. UP-GRS achieved a TASR of 0.9690, which is much higher than the other two methods. The performance of the three methods in terms of TASR with nonlinear acceptance rates is shown in Fig. 6.4k. Similar to the result shown in Fig. 6.4i, the TASR achieved by our method was around 0.8, and UP-GRB achieved a TASR of around 0.9, which was much higher than expected. The TASR value changed significantly when the privacy budget $\epsilon$ varied. Figs. 6.4j and 6.4l show the results for the SimplGeo dataset. We were able to observe that the higher TASR was at the cost of increased overhead more notified workers and a longer travel distance to the task destination. Our method achieved a good trade off, achieving the ESR while reducing the number of notified workers and their travel distance.

**Performance by varying MTD**

We evaluated the performance of the proposed method on both datasets by varying the maximum travel distance (MTD). Fig. 6.5 shows the results when the acceptance rate has a linear distribution. The results with a nonlinear distribution show similar performance. We observed that when the MTD was small, more workers were required to achieve the ESR. For example, as shown in Fig. 6.6a, with $\epsilon = 0.5$, when $MTD = 1km$, more than 8 workers were needed to guarantee an 80% success rate on the Yelp dataset. While only around 4 workers were sufficient when the maximum travel
Figure 6.5: The performance by varying MTD
distance was increased to 5 km. This is because a worker has a higher probability of accepting a task at a fixed distance when the maximum travel distance is longer. Meaning, fewer workers are needed to achieve the ESR. Fig. 6.6b shows a similar trend when increasing the MTD for the SimpleGeo dataset. We also observed that changing the MTD had little effect on the ATD, irrespective of the dataset, which is shown in Figs. 6.6c and 6.6d, respectively. The value of ATD basically remained the same, especially, when the added noise was smaller. This trend affects the MTD’s influence on ATD.

Figure 6.6: The performance by varying ESR
Performance by varying ESR

The variations in the tendencies of ANW and ATD for the Yelp and SimpleGeo datasets along with the parameter’s ESRs are shown in Fig. 6.6. We observed that a higher ESR results in both higher ANW and ATD. Fig. 6.6a shows the effect of ESR on ANW for the Yelp dataset with linear acceptance rates. We can observe that when $\epsilon = 0.5$, and $ESR = 0.7$, 3.6 workers are enough to achieve a 0.7 success rate. To achieve a higher ESR, more workers need to be notified. Fig. 6.6b shows similar results for the SimpleGeo dataset in terms of ANW. Figs. 6.6c and 6.6d indicate the impact of increasing the ESR for ATD. When the $ESR = 0.3$, $\epsilon = 1$, a travel distance of 0.085km was needed to finish the task in Fig. 6.6c. However, when the $ESR = 0.9$, the travel distance was increased to 0.096km. This is because obtaining a higher task assignment success rate requires more cells to construct a larger geocast region, which leads to an increased travel distance as well as an increase in the number of notified workers.

6.6 Related Work

Location privacy has been studied extensively. For example, dummy locations [53] were proposed to protect user locations by adding false positions to the true locations; cloaking region techniques [13] transform the exact location to a sufficiently larger region to reduce location precision; the transformation method [50] performs some basic geometric operations over a user’s location; private information retrieval [167] uses encryption to protect a user’s location; and differential privacy-based perturbation methods [11] have also been proposed.
These techniques have largely been used and studied in location-based services. However, only a few studies focus on crowdsensing [84]. Kazemi et al. [69] presented a privacy framework in which each participant forms their own cloaked region by computing a Voronoi cell in a distributed fashion. Then, a voting mechanism is devised to select the set of representative participants and send their cloaked regions to the server. The query results are subsequently shared with the rest of the participants. Similar to the method proposed by Kazemi et al., Hu et al. [58] employed a peer-to-peer cloaking technique to cloak worker locations among $k - 1$ other workers. In addition, Bin et al. [180] presented a clustering method in which the location of the virtual cluster center is reported to the server by the cluster head. Once the cluster head receives the task from the server, tasks are assigned to the chosen cluster member according to their exact location. However, none of these obfuscation-based techniques provides a rigorous privacy guarantee. Their reliability is highly dependent on an adversary’s background knowledge. Once the attacker obtains a key piece of background knowledge, such as a location the user visits frequently, the user’s location can easily be inferred. Shen et al. [128] applied an encryption technique and proposed a privacy framework that performs worker task matching in an encrypted domain. In particular, they introduced a semi-trusted third party to provide privacy functionality and collect encrypted data from workers. The server communicates with the third party in the encrypted domain to find workers at a minimum cost. The advantage is that it can provide a strong privacy guarantee. However, encryption-based technology is often computationally and communicationally expensive.

Differential privacy is a powerful privacy model that satisfies privacy regardless of the attacker’s background knowledge. It is also less computationally expensive. With
this privacy definition, To et al. [141] proposed a privacy-aware framework to protect user’s location information in spatial crowd-sourcing by introducing a cellular service provider (CSP) as a trusted third party. The CSP collects workers’ locations and then partitions the entire spatial region into a grid of indexed cells by applying the CSP’s partition algorithm. Laplace noise is added to the count of each cell, and the sanitized data are released to the service provider. Latter, they also presented a tool box [142] to display the framework in a visual, interactive environment. Their recent work [140] was extended in a further solution, addressing dynamic worker datasets [141] that investigate privacy budget allocation techniques across consecutive releases and employ post-processing based on Kalman filters to improve the accuracy. Yanmin et al. [46] proposed a similar differential privacy framework for task assignment in ad hoc mobile clouds. They not only consider location privacy but also service quality, which considers the mobile servers’ reputation. In addition, Wang et al. [151] propose a location privacy-preserving task allocation framework with geo-obfuscation to protect users’ locations during task assignments, which make participants obfuscate their reported locations under the guarantee of differential privacy. Ping et al. [162] presented a differentially private allocation mechanism for reward-based spatial crowdsourcing. They presented a contour plot to characterize location distribution and proposed an optimized-reward allocation method to achieve a specified probability of assignment success.

6.7 Summary

In this chapter, we proposed a privacy-preserving data release method based on worker density. This method satisfies differential privacy and enables workers to participate
in crowdsensing platforms without disclosing their location. In addition, the proposed method improves the accuracy of the released data. We also proposed an optimal geocast region selection strategy that considers the distance workers must travel and the number of workers that are notified of available tasks. The proposed geocast region selection strategy not only achieves the expected task assignment success rate but also reduces system overhead. We evaluated the performance through extensive experiments, and the results prove that our method achieves a better balance between task assignment success rate and system overhead with the same privacy guarantee.
Chapter 7

Conclusion and Future Work

The research presented in this thesis mainly discusses the privacy problems in various fields. It consists of four parts: the first part discusses the privacy-preserving problem in the recommendation system, mainly focuses on the privacy disclosure in neighbourhood-based collaborative filtering; the second part focuses on the privacy problem of aggregation on the Internet of Things; the third part concentrates on the privacy problems in location-based services; the fourth part discusses the privacy problems in crowdsensing system. The privacy problems are identified in each chapter and the corresponding solutions are proposed. The proposed solutions aim to protect the data privacy with guaranteed privacy level, while enhancing the data utility. This chapter summarizes the main contributions of this thesis, and several open issues for future research have also been identified.

7.1 Contributions

We summaries the main contributions of this thesis as follows:
We proposed a privacy-preserving collaborative filtering method, which protects the users’ rating records without compromising the recommendation accuracy. Specifically, we introduced a randomization method, called Johnson-Lindenstrauss transform, which project the records in a high dimension to a lower dimensional space while the distances between records are preserved. Though this randomization transform, the data utility is enhanced, especially for the sparse rating datasets. In addition, we prove that the transformation on the rating dataset satisfies differential privacy, which provides provable and rigorous privacy protection.

We proposed a privacy-preserving data aggregation method under fog computing architecture. To allow the flexible aggregation queries to meet the diversified...
aggregation goals, we propose using the machine learning algorithm to train the
learning model to predict the query results, and allocate the aggregator at the
fog center to report the aggregation results to the cloud server. In addition, we
make the training process satisfy differential privacy by adding \textit{Laplace} noise
to the query results in the training dataset. Therefore, the proposed method
can effectively defend the differential attack, which often appears in most ag-
gregation functions. The experiments show that the more queries, the better
performance the proposed method presented compared with traditional \textit{Laplace}
mechanism.

• We proposed a location privacy-preserving framework for location-based services
(Point of Interest query). The main idea is applying the Johnson-Lindenstrass
transform to the location coordinates. Through the random transformation, the
user’s and PoIs’ location coordinates become high dimensional location vectors,
but the relative distances are maintained. Therefore, the user’s location privacy
is protected locally, nobody knows the user’s exact location. Also, we proved
that the process satisfies differential privacy, which not only guarantees rigorous
privacy preservation but also allows LBS providers to provide accurate services.

• We proposed a privacy-preserving data release method for crowdsensing system.
The proposed method includes three phases, worker density-based partition, dif-
ferential privacy data release and geocast region selection. The proposed worker
density-based partition method considers the workers’ real location distribution,
which contributes to a high task assignment success rate. In the differential pri-
vacy data release process, the \textit{Laplace} noise was added to the statistical result
of each grid to hide the worker’s location information. The proposed geocast
region selection method ensures the balance between task assignment success rates and system overhead. The systematic theoretical analysis and extensive experimental results show that the proposed method has better performance with the same privacy-preserving level.

To make it much clearer, we summaries the key contribution of the proposed methods in each scenario as shown in Fig. 7.1.

7.2 Future Works

Although the proposed methods solved some privacy issues in various application scenarios, there are still some problems that need to be addressed. This section discusses the open issues as the extensions of the presented work in the thesis.

**Recommendation system.** We discussed the privacy problem in the recommendation system, and proposed a Johnson-Lindenstrass transform based privacy-preserving method to protect the user’s rating history. The proposed method only concentrates on neighbourhood-based collaborative filtering. However, other recommendation techniques, such as matrix factorization, still suffer from privacy disclosure. Therefore, the future work should consider the privacy-preserving problem in other recommendation techniques. In addition, one limitation of the proposed method is that it assumes the service provider is trusted, which is a very common assumption in existing works. It cannot defend the malicious information disclosure by service provider and server attack. Therefore, for the future work, we would like to develop the privacy framework under the untrusted server setting. Make it robust to malicious users as well as untrusted service provider.
**Location-based services.** For location-based services, the most existing privacy-preserving methods are designed under Client-Server structure, users inevitably expose partial information to the service provider in exchange for usable services. Once the attacker gets appropriate background knowledge, users’ location privacy would be violated. For example, if the attacker knows that the user is a staff member in Deakin University at Burwood campus, he can confidently infer that the user is in Deakin at working time when the user is at Burwood region. Therefore, to defend such strong background knowledge attack, even the user located region should not be disclosed.

In chapter 5, to protect the user’s location information, we introduced a semi-trusted third party and combined with multiple privacy-preserving techniques. Though the server has no idea the user’s location information, the third party knows some clues about the user’s location. However, the John-Lindenstrass transform gives a good start for blind matching that allows distance calculation with perturbed location information. For the future work, we would like to develop a more strict location privacy protection strategy, which can prevent the strong background knowledge attack.

**Crowdsensing system.** In chapter 6, we proposed a location privacy-preserving method in the crowdsensing system. Though the proposed the method improves the task assignment success rate significantly, the proposed method is still based on the trusted third party. The main weakness of the trusted third party is the single point of failure. Therefore, we would like to explore the new location privacy preserving method that can protect the worker’s location locally without consuming the utility. In addition, the proposed method only consider the privacy disclosure during the location reporting and task assignment process, the payment process would disclose the worker’s location as well. As the payment information is associated with the real
identity of the worker. The server can infer in which task a particular worker is by observing the payment. Therefore, the location privacy disclosure in the payment process is another research direction in the future.
Bibliography


[23] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In Proceedings of


