Differentially Private Data Analytics

Zhigang Lu

School of Computer Science

The University of Adelaide

This thesis is submitted for the degree of

Doctor of Philosophy

Supervisors: Prof Hong Shen and Prof Michael Sheng

January 2021

To my mother, father and wife,

my late paternal aunt,

and all the people

who encouraged me over the past years.

YNWA

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Zhigang Lu

January 2021

Acknowledgements

This thesis would not have been possible without the support I recieved from many people. I would like to take this opportunity to acknowledge all the people who have supported and helped me during my PhD candidature.

First of all, I would like to express my sincere gratitude to my supervisors for their continuous support and guidance over the past years. I am grateful to my principal supervisor, Prof. Hong Shen, for the constructive discussions and arguments, through which I learnt the skills to do independent research and the qualities to embrace my academic career. I also appreciate my co-supervisor, Prof. Michael Sheng, for his patience and encouragement. His insightful suggestions and positive thinking always help me to overcome the difficulties during my PhD candidature.

Secondly, I would like to thank my colleagues for creating a friendly and inclusive atmosphere, which makes everthing easier. Thanks to Prof Longkun Guo, Dr Kewen Liao, Dr Wei Emma Zhang and Mr Shuichi Sakai for their career suggestions; to Ms Nicola Macdonald, Dr Feng Shi, Mr Ping Mao and Dr Peng Fu for their personal support. Thanks also go to the school staff, Ms Hilary Brookes, Ms Melissa Smithen, Ms Julie Mayo and Ms Sharyn Liersch, for managing all the daily challenges.

I also thank the Liverpool Football Club for the spirit of never giving up, which keeps encouraging me when exploring the uncharted research world. I salute this spirit by quoting the club's anthem here: "When you walk through a storm, hold your head up high, and don't be afraid of the dark. At the end of a storm, there's a golden sky, and the sweet silver song of a lark. Walk on through the wind, walk on through the rain, though your dreams be tossed and blown. Walk on, walk on, with hope in your heart, and you'll never walk alone. You'll never walk alone. Walk on, walk on, with hope in your heart, and you'll never walk alone. You'll never walk alone." [49]

From the bottom of my heart, I would like to say big thank you to my family and friends. I am indebted to my parents and my wife for their unconditional support and endless love. They always prioritise me even sacrificing themselves. They are the true heroes in my life. Furthermore, I am fortunate for the friendship I have from Zheng, Meng, Javier, Ana, Catherine and Kym in the past years. The insightful talks with them always inspired me from some of the challenging moments during my PhD candidature.

Finally, I thank the anonymous examiners for their valuable comments to improve the quality of this thesis. I also thank School of Computer Science for financially supporting my conference travels. I acknowledge that this thesis was supported by Australian Government Research Training Program Scholarship and computing resources were supported with supercomputing resources provided by the Phoenix HPC service at the University of Adelaide. This thesis was edited by Elite Editing, and editorial intervention was restriced to Standards *D* and *E* of the Australian Standards for Editing Practice.

Abstract

With the emergence of smart devices and data-driven applications, personal data are being dramatically generated, gathered and used by modern systems for data analytics in a wide range of customised service applications. Despite the advantages of data analytics, potential risks arise that allow adversaries to infer individuals' private information by using some auxiliary information. Therefore, it is crucial to develop new methods and techniques for privacy-preserving data analytics that ensure acceptable trade-offs between privacy and utility.

Over the last decade, differential privacy (DP) has been viewed as a promising notion of privacy because it mathematically bounds the trade-off between privacy and utility against adversaries' strong inference attacks ($n-1$ out of n items in the input). By exploring the latest results of differentially private data analytics, this thesis concentrates on four sub-topics: *differentially private data aggregation with security guarantees*, *privacy budget guarantees in distributed systems*, *differentially private single-path publishing* and *differentially private k-means clustering*.

For differentially private data aggregation with security guarantees, we propose a two-layer data aggregation approach against semi-honest but colluding data aggregators, where DP noise is randomly injected. We address the problems of *the collusion of data curators and data aggregators*. The key idea of our approach is injecting DP noise randomly to prevent privacy disclosure from collusion, while maintaining a high degree of utility and splitting and sharing data pieces to guarantee security. The experimental evaluations over synthetic datasets confirm our mathematical analysis results and show that our approach achieves enhanced aggregation utility.

For privacy budget guarantees in distributed systems, we study the parallel composition of privacy budget in differentially private data aggregation scenarios. We propose a new lower bound of the parallel composition of privacy budgets. We address two problems of the state-of-the-art: *poor utility when using global sensitivity for both data curators and data aggregators* and *unknown privacy guarantees with conditions on the local sensitivities between data curators and data aggregators*. The key is taking a property of the summation function: the local sensitivity of summation for a dataset is equal to the maximum summation of any sub-dataset. The experimental results over a real-life dataset support our theoretical results of the proposed lower bound of the unconditional parallel composition of privacy budget.

For differentially private single-path publishing, we propose a graph-based single-path publishing approach with DP guarantees. We address two problems in existing work: *information loss regarding the exact path for genuine users* and *no privacy guarantees for edges when adversaries have information about all vertices, yet one edge is missing*. The main idea is adding fake vertices and edges into the real path by DP, and then hiding connections in the perturbed path in the topology of the published graph so that only the trusted path users with full knowledge about the map can recover the real path. The experimental evaluations of synthetic datasets corroborate the theoretical properties of our approach.

For differentially private k-means clustering, we propose a convergent differentially private k-means algorithm that addresses *the non-convergence problem of existing work*. The key idea is that, at each iteration, we sample a centroid for the next iteration from a specially defined area in each cluster with a selected orientation to guarantee both convergence and convergence rates. Such an orientation is determined by either past centroid movements or past plus future centroid movements. Both mathematical and experimental evaluations show that, because of the convergence, our approach achieves enhanced clustering quality over the state-of-the-art of DP *k*-means clustering, while having the same DP guarantees.

Table of Contents

List of Figures

List of Tables

Nomenclature

Greek Symbols

Δf	The function sensitivity in differential privacy
ε	The privacy budget of differential privacy

Acronyms / Abbreviations

DDP	Distributed Differential Privacy
DP	Differential Privacy
EMD	Earth Mover's Distance
ExpDP	Exponential mechanism of Differential Privacy
GA	Genetic Algorithm
HBC	Honest-but-Curious
IoT	Internet of Things
LapDP	Laplace mechanism of Differential Privacy
LBS	Location-based Service
MSE	Mean Square Error

NBN	National Broadband Network
OT	Oblivious Transfer
PDF	Probability Density Function
POI	Points-of-Interest
RDP	Rényi Differential Privacy
SDD	Sampling Distance and Direction
SFE	Secure Function Evaluation
$SMC(MPC)$	Secure Multi-party Computation
WHO	World Health Organisation

Chapter 1

Introduction

Privacy is a fundamental human right [109]. Achieving acceptable trade-offs between data utility and data privacy is a continuous research challenge for data analytics. This thesis studies highly utilised privacy-preserving solutions to some computing tasks of data analytics, including data aggregation/collection, data publishing/sharing and data mining/machine learning.

In this chapter, we first briefly introduce the research background of this thesis, including the overview of non-private data analytics, privacy issues of the non-private data analytics and the privacy-preserving techniques (the notion of privacy) used in this thesis. Afterwards, we state the research scope of this thesis, discuss the research challenges of the state-of-the-art and summarise our contributions.

1.1 Research Background

1.1.1 Brief Overview of Data Analytics

Over the last decade, data, especially personal data, have been explosively produced, collected and consumed because of the proliferation of smart devices and the Internet of Things (IoT). It has been reported that, by 2020, there will be about 40 zettabytes of data in the world [67, 30]. Based on the latest maximum download speed of Australia's National Broadband Network (NBN), say 60 Mbps [111], it would take around 170 million years to download these data. Note that 170 million years ago was the Jurassic Period.

It is commonly accepted that such a data explosion has been caused by the emergence of data-driven applications. By taking advantage of insights learnt from customers' data, data-driven applications can offer significant financial benefits to businesses and convenient personalised services to customers. For example, modern recommender systems of online shopping (e.g., Amazon), social networks (e.g., Facebook) and streaming music/video providers (e.g., Spotify and YouTube) collect and analyse customers' profile information and visit records every single minute to provide reliable and personalised suggestions for customers and their potential interests [29].

To gain insights from customers' raw data, there are many computing tasks in the process of data analytics. In this thesis, we focus on three key computing components: data aggregation/collection, data publishing/sharing and data mining/machine learning [169]. Specifically, data aggregation/collection component gathers raw data from distributed data curators/contributors, such as customers of recommender systems or smart electricity meters [146, 15]. Data publishers broadcast the data (without explicit identifiers of an individual) to authorised organisations and/or persons wishing to further study the data. Data publishing/sharing may occur without provision of secure channels or data encryption [46]. Finally, suitable data mining/machine learning algorithms, according to the computing tasks, gain knowledge from the collected data.

1.1.2 Privacy Disclosure against Data Analytics

Despite the potential benefits and insights attained by such a data analytics process, privacy disclosure (data breaches) can arise and generate huge financial loss [8], as well as discouraging people's willingness to share personal data [123]. Since 2004, there have been a large number of severe data breaches (privacy disclosure of at least $30,000$ records) reported worldwide involving all types of organisations, including government, finance, academia and high-tech firms, to name a few [164]. Most of the privacy breaches listed in [164] were caused by hacking, inappropriate permission for use of data and theft by system insider(s). For example, in 2018, it was revealed that around 50 million Facebook users' profile information and social connections were used for political opinion predictions without the users' permission [165].

To prevent private information disclosure as a result of hacking, inappropriate permission and system insider theft, a normal and useful approach involves keeping the system firewall up to date, and reviewing and refreshing the database permission/authority list regularly [117, 139, 135]. However, there is a group of attacks, often referred to as *inference attacks* [77, 110, 141, 98] that are more challenging for privacy preservation. Some famous inference attacks include linkage attacks [151], sybil attacks [31], continuous observation attacks [16] and shilling attacks (including nuke attacks and push attacks) [55]. Inference attacks aim to infer partial or all input data by considering auxiliary knowledge held by adversaries against a given computing outcome. Figure 1.1 depicts an overview of such inference attacks (leveraging the computing outputs and adversaries' auxiliary knowledge) against data analytics.

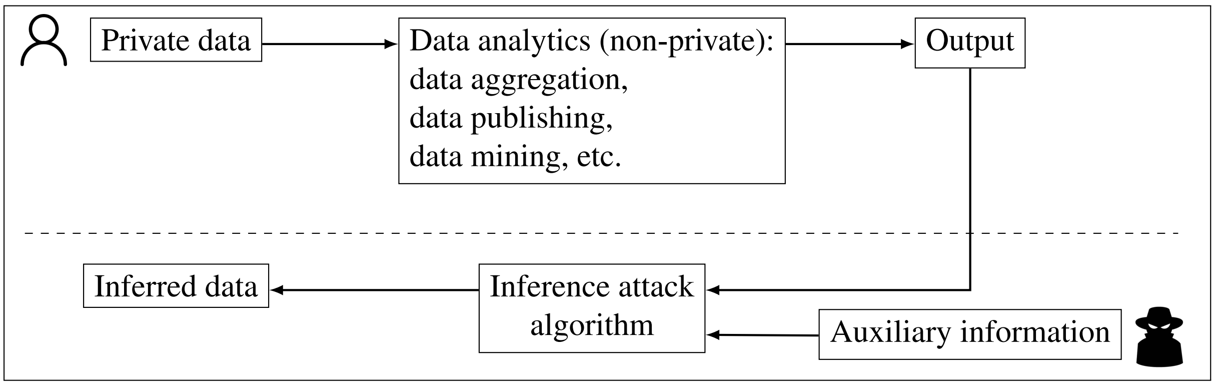


Fig. 1.1 Overview of Inference Attacks.

We formally define the inference attacks for this thesis in Definition 1.1.

Definition 1.1 (Inference Attack)**.** *Given a function f, a private dataset $X = \{x_1, \ldots, x_n\}$, an adversary's inference function g and the adversary's auxiliary information set A, where $A \cap X = \{a_i, \ldots, a_j\}$, $\exists x_i \in X$ and $a_i \cap x_i \neq \emptyset$, an inference attack successfully infers privacy if and only if $(Y \setminus A) \cap X \neq \emptyset$, where Y is the inferred data, defined by $Y = g(f(X), f(A))$.*

We have the following remarks for the inference attack in Definition 1.1:

- f could be any computing component involved in the data analytics. In this thesis, we specify them as data aggregation/collection, data publishing/sharing and data mining/machine learning.
- In the best case, where the adversary has no background knowledge, A is empty.
- In the worst case, where the adversary has all except one arbitrary item of the private dataset, $A = X \setminus \{x_i\}$. That is, when the adversary has one more data point x_i, such that $A = X$, the adversary does not need to infer anything about X.
- $Y \setminus A$ is the privacy inferred by a potential adversary, that is the difference between any subset of the private dataset and the adversary's auxiliary information. In the worst case, $Y = X$.

Such inference attacks could successfully access additional private data, with sufficient auxiliary information, from any computing component of the data analytics process. For instance, let f be a counting function and X be a list of patients' human immunodeficiency virus (HIV) test results, where the number of positive/negative cases is 1/0, $f(X) = n$. If $f(A) = f(X) = n$, then the adversaries infer that the unknown patients in $X \setminus A$ are all HIV negative. Detailed running examples of the way inference attacks work against each component of data analytics are presented in Chapter 4 (for data aggregation/collection), Chapter 6 (for data publishing/sharing) and Chapter 7 (for data mining/machine learning).

1.1.3 Privacy-preserving Techniques for Data Analytics against the Inference Attacks

Generally, the goal of privacy-preserving techniques for data analytics is to produce privacy-preserved computing outcomes, which minimise the opportunity of inferring the privacy. Achieving indistinguishability during computation is a common direction in current research. Indistinguishability guarantees that the computing outcomes over the original private dataset and adversaries' auxiliary information set are similar (or the same) with a given similarity measurement.

Currently, there are three main techniques that implement the concepts of indistinguishability for security or privacy-preservation purposes: secure multi-party computation (SMC) [174], anonymity [150] and differential privacy (DP) [33]. Briefly, SMC ensures that a real protocol, π, for multi-party computation achieves the indistinguishable behaviour of a securely trusted third party, T, which takes the inputs from each party and produces a computing output without compromising any input. This way, SMC defines the security of π by the security of T, that is, the value of a private input X to a joint computation cannot be inferred by the computing outcome, $f(X)$. However, when adversaries have some auxiliary information, SMC is vulnerable to inference attacks. The anonymity requires that each item, $x_i \in X$, is indistinguishable from a group of items, such that adversaries cannot infer a specific item even with preassumed auxiliary information. However, when adversaries have different auxiliary information, anonymity may be vulnerable. DP guarantees that the outcomes over the original private dataset X and a dataset X', which differs in one item to X, are indistinguishable. In DP, X' is assumed to be within the adversaries' auxiliary information, which is the maximum information the adversaries could have, thus DP is the most promising notion of privacy for privacy-preserving computation at this stage. Table 1.1

Table 1.1 Brief Comparisons of SMC, Anonymity and DP.

	Aux Info	Indistinguishability	Vulnerability
SMC	$\emptyset$	Security of $f(X)$ run by π = security of $f(X)$ run by T	When $A \neq \emptyset$, no privacy guarantee
Anonymity	A predefined A	$f(X) = f(A)$	When $\exists A'$ that $f(X) \not\approx f(A')$, no privacy guarantee
DP	X', $X \setminus X' = \{x_i\}$	$f(X) \approx f(X')$	At this stage, no significant vulnerability

presents a brief comparison of SMC, anonymity and DP for indistinguishability against inference attacks.

1.2 Research Scope, Challenges and Contributions

Given the strong privacy guarantees of DP, we use DP as the basis for implementing privacy-preserving data analytics. That is, this thesis presents research on differentially private data analytics preserving individual privacy while retrieving insights from raw data against the strongest adversary's auxiliary information across the computing components specified in Section 1.1 - *differentially private data aggregation*, *differentially private data publishing* and *differentially private data mining/machine learning*.

1.2.1 Research Scope

In this thesis, we particularly study the above three research topics through the following specific research problems:

- For *differentially private data aggregation*, we focus on two problems. First, we consider how a data aggregation system guarantees both the (output) privacy and (computing) security when delivering aggregation results, that is, differentially private data aggregation with security guarantees. Second, we consider the privacy performance of the whole data aggregation system (aggregating data from distributed data curators) through a summation/counting function when injecting DP noise at each data curator locally, that is, the privacy budget guarantees of distributed systems.

- For *differentially private data publishing*, we study this via differentially private path publishing. This is triggered by the fact that, because of the prevalence of global positioning system (GPS)–enabled devices, location-based service mobile applications, such as WhatsApp, Facebook, Google Maps and Uber, have perhaps been the most downloaded and used apps in the world and offer the greatest privacy issues. Protecting personal privacy (relationships between individuals in a path) when publishing a path of connected individuals (e.g., trajectory) has become an urgent need.
- For *differentially private data mining/machine learning*, we focus on differentially private k-means clustering, which preserves membership privacy of the input dataset.

1.2.2 Research Challenges and Contributions

Given this research scope, the research challenges and thesis contributions are listed below:

- For *differentially private data aggregation with security guarantees*, the *research challenges* of existing work [130, 140, 1, 38, 52, 39, 122] include the collusion of data curators/contributors, the collusion of data aggregators, the vulnerability of a single data aggregator and the high communication cost when the network topology is a complete graph without a data aggregator. To address these challenges, we propose that DP noise is randomly injected into two-layer data aggregation against semi-honest yet colluding data curators and colluding data aggregators [88]. Our *main contributions* here are guaranteeing both DP and security with low community cost, while maintaining high data utility against both collusion of data curators and collusion of data aggregators. Our approach involves the following steps. First, each data curator randomly injects DP noise into their data, and then splits and shares each piece of the differentially private data with semi-honest data aggregators. Second, each data aggregator aggregates the received pieces and then randomly injects DP noise into the aggregated data. Finally, the data aggregators securely compute the aggregation output. Both mathematical and experimental analyses[1] (on synthetic datasets) show that our approach achieves enhanced data utility, while maintaining the same DP and security guarantees as existing work.

- For *privacy budget guarantees of distributed systems*, we study the parallel composition of the privacy budgets of differentially private data aggregation for the summation/counting function. To address the *research challenges* with existing work [99, 140]: low overall data utility through global sensitivity for both aggregators and curators and no privacy guarantees with unconditional local sensitivities for curators and aggregators, we provide a new lower bound of the parallel composition of privacy budget for differentially private data aggregation for the summation/counting function [89]. Our *main contributions* are ensuring acceptable overall data utility with unconditional local sensitivities between curators and aggregators. The key idea here is taking advantage of a property of the summation/counting function when computing local sensitivity, where the local sensitivity of summation/counting for the aggregation system is equal to the maximum summation of any individual data curator. We also confirm the theoretical results by running experiments[2] over real-life datasets.

- For *differentially private single-path publishing*, to address the *research challenges* of existing work [71, 171, 61, 62, 113, 158, 157, 173, 54, 20], we consider the case, where trusted path users should recover the exact path from the published path while preserving the privacy of the edges (i.e., the relationships between individuals), if adversaries' auxiliary information includes all vertices, instead of partial vertices as

[1]Source code (Chapter 4): https://github.com/suluz/sec_dp_aggregation
[2]Source code (Chapter 5): https://github.com/suluz/dist_DP

assumed in existing work. We develop a graph-based path publishing scheme in [92]. Our *main contributions* are preserving the privacy of the existence of any edge in a map against adversaries that have all information about vertices and $m-1$ out of m edges, while, with high probability, guaranteeing full utility for trusted path users. Specifically, we publish a graph where each node represents a vertex (that could be either fake or real). The connections in the path for publishing are concealed in the topology of the graph. By reading such a topology, only trusted path users that have correct/full information about the existence of an edge can recover the exact path from the published graph, while adversaries with partial information cannot. The experimental results[3] on synthetic datasets match the theoretical properties of the published graphs.

- For *differentially private k-means clustering*, to address the *research challenge* of the non-convergence in existing work [12, 34, 104, 176, 112, 147, 148, 118], we present an approach guaranteeing the convergence in [90, 91]. Our *main contribution* is providing enhanced clustering quality (because of the convergence guarantee), while achieving the same DP requirement as existing work. In particular, to ensure convergence, we differentially privately update the centroid of a cluster at each iteration from a specially designed area in the cluster. Further, to enable a good convergence rate, we control the orientation of the centroids update through knowledge of past centroid movements or knowledge of past plus future centroid movements. We prove that, in the expected/average case, our algorithm converges (to the same clusters as Lloyd's k-means algorithm [87]) in at most twice as many iterations as Lloyd's algorithm. The experimental (on real-world datasets) results[4] show that our algorithm outperforms the state-of-the-art methods with guaranteed convergence and enhanced clustering quality, while meeting the same DP requirement.

[3]Source code (Chapter 6): https://github.com/suluz/dp_path
[4]Source code (Chapter 7): https://github.com/suluz/diffpriv_clustering

1.3 Thesis Publications

During my PhD study at the University of Adelaide, I, as the first author, produced five papers related to this thesis. Three conference papers were published and two manuscripts that are currently under review as journal articles (one of them recieved a major revision from the first-round review). This thesis is largely based on the content presented in the following papers.

Conference Papers

- **Zhigang Lu**, Hong Shen. 2019. "A Convergent Differentially Private k-Means Clustering Algorithm", In *Proceedings of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2019).* Springer, pp. 612-624. [90]
  CORE rank A

- **Zhigang Lu**, Hong Shen. 2017. "A new lower bound of privacy budget for distributed differential privacy", In *Proceedings of the 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT-2017).* IEEE, pp. 25-32. [89]
  CORE rank B

- **Zhigang Lu**, Hong Shen. 2017. "Secured Privacy Preserving Data Aggregation with Semi-honest Servers", In *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2017).* Springer, pp. 300-312. [88]
  CORE rank A

Journal Articles

- **Zhigang Lu**, Hong Shen. 2020. "Differentially Private k-Means Clustering with Convergence Guarantee". IEEE Transactions on Dependable and Secure Computing

(TDSC). Early Access. [91]

CORE rank A

- **Zhigang Lu**, Hong Shen. 2020. "Protect Edge Privacy in Path Publishing with Differential Privacy". Submitted to IEEE Transactions on Knowledge and Data Engineering (TKDE). [92]

  CORE rank A*

1.4 Thesis Organisation

The remainder of this thesis is organised as follows:

- **Chapter 2** presents a brief introduction of the most commonly used notions of privacy, including secure multi-party computation (SMC), anonymity and differential privacy (DP). In particular, when reviewing each notion of privacy, we describe its motivation scenarios first, and then present the basic concepts and formal definitions through running examples. Finally, we discuss their possible vulnerabilities against given adversaries with background knowledge. By comparing the notions of privacy, we choose DP as the basis for privacy in this thesis because of its mathematical bound of privacy preservation against adversaries with maximum auxiliary information.

- **Chapter 3** reviews the literature of differentially private data analytics, including differentially private data aggregation with security guarantees, parallel composition of privacy budgets of differentially private data aggregation for summation/counting, differentially private single-path publishing and differentially private k-means clustering. We analyse both the advantages and disadvantages of existing work on the above topics. Such analyses help us discover the research gaps, which lead us to propose our approaches and contributions.

- **Chapter 4** focuses on differentially private data aggregations with security guarantee. In this chapter, we propose a differentially private data aggregation framework where malicious data curators and semi-honest data aggregators inject DP noise randomly. We mathematically prove the DP and security of the framework. We also use synthetic datasets to experimentally compare the utility of our approach with existing work, while achieving the same DP guarantees.

- **Chapter 5** studies the privacy budget guarantees of distributed systems. Specifically, we consider the parallel composition of the privacy budgets of differentially private data aggregation during summation/counting. In particular, we provide a new lower bound of privacy budget for distributed differentially private summation/counting. Our bound of privacy budget is tighter than existing summation/counting functions, without any conditions on the function sensitivity. We evaluate this bound experimentally on real-life datasets, which confirm the theoretical performance of our lower bound on the privacy budget.

- **Chapter 6** considers the existence of a single edge in a graph as the privacy when publishing a path, rather than preserving the privacy of vertices, as considered in existing work. We propose a graph-based path publishing algorithm that conceals the real path in the topology of the published graph. DP is implemented by adding fake edges and fake vertices to the published graph. Both mathematical and experimental evaluations are provided to study the performance of our proposed algorithm.

- **Chapter 7** considers the convergence of a differentially private k-means clustering algorithm. In Chapter 7, we first discover a convergent zone for each cluster to ensure convergence, and then take care of the convergence orientation to achieve an acceptable convergence rate. We apply an exponential mechanism of DP to perturb and update centroids within the given convergent zone and the given convergence orientation. We analyse the properties related to convergence mathematically and experimentally over real-world datasets.

- **Chapter 8** summarises this thesis and identifies several possible extensions and areas for future work.

1.5 Summary

To conclude, this PhD thesis aims to develop new methods and techniques for differentially private data analytics, where the entire process of data analytics (data aggregation/collection, data publishing/sharing and data mining/machine learning) is guaranteed based on DP to achieve the most acceptable trade-offs between data utility and privacy against adversaries with maximum background knowledge. In particular, we develop methods from the following four parts: differentially private data aggregation with security guarantees, the privacy budget guarantees of distributed systems, differentially private single-path publishing and differentially private k-means clustering. The research challenge in this thesis is improving the data utility of the above computing tasks while maintaining the same DP guarantees as existing work. To do so, for each part, we consider the research gap of the state-of-the-art works, propose our framework/algorithms to address the problems with existing work, mathematically evaluate the key features or properties of our proposed approaches and compare the performance of our work and existing work with real-life (or synthetic) datasets.

Chapter 2

Background on the Notion of Privacy

The notion of privacy is important to privacy-preserving algorithms, as it is the basis to evaluate whether a framework, a scheme or an algorithm is privacy preserved (or secure) under a given set of assumptions of adversaries. In this chapter, we briefly summarise three famous notions of privacy that are widely used in most of the state-of-the-art privacy-preserving research fields: secure multi-party computation (SMC) [174, 175], anonymity [150, 95, 80] and differential privacy (DP) [37, 33].

Indistinguishability is a core concept used to define security and privacy. SMC defines security by ensuring that the outputs of a joint computation from a secure protocol and a fully trusted middle party are indistinguishable. Anonymity publishes privacy-preserved data, where each individual belongs to a group of indistinguishable individuals. DP guarantees that slightly different datasets (in the worst case, different by one record) produce indistinguishable outputs with a given probability bound.

SMC, as a paradigm, aims to formally define the security of a jointly computed protocol/function without a trusted third party. In SMC, a protocol/function is secure, if and only if, multiple parties jointly compute a function over their own private data, while ensuring that no party learns the others' private data from the outputs of the joint computation. As a result, SMC guarantees full utility and security of the outputs of a computing protocol against adversaries without auxiliary information. Unfortunately, adversaries with some background knowledge could infer private information from the secure outputs of an SMC function. In this case, given utility requirements and adversaries' background knowledge (to form an inference attack), an anonymity scheme processes the raw data by removing or generalising the precise information that is vulnerable to adversaries' background knowledge, whilist retaining specific statistical features of the data according to the utility requirements. However, as anonymity schemes do not have a clear measurement of an adversary's auxiliary information, such schemes only produce sound results against adversaries with pre-assumed background knowledge. To address this, DP provides a formal metric to measure the trade-off between utility and privacy in the case where the adversaries have maximum background knowledge: $n-1$ out of n items of a dataset. In particular, DP bounds the probability to distinguish whether a differentially private output is produced by the dataset with n items or $n-1$ items, so that any arbitrary item in the dataset is concealed against adversaries with maximum auxiliary information. Figure 2.1 depicts how SMC, anonymity and DP achieve the concept of indistinguishability from different aspects, while considering an adversary's background knowledge.

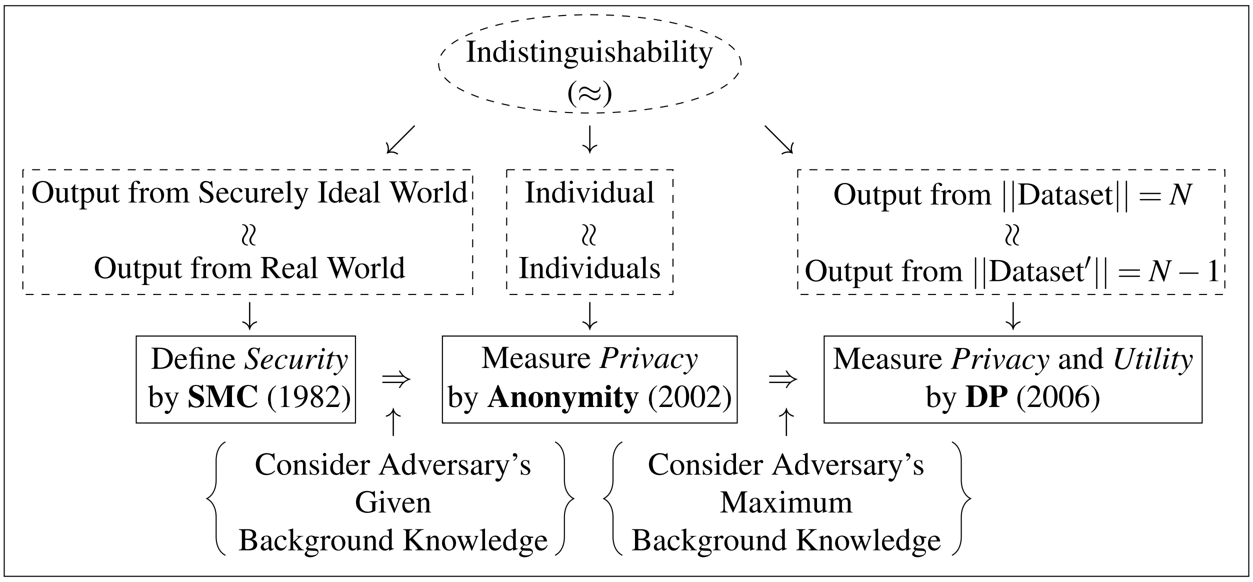

Indistinguishability (≈)
Output from Securely Ideal World ≈ Output from Real World
Individual ≈ Individuals
Output from ||Dataset|| = N ≈ Output from ||Dataset′|| = N − 1
Define Security by SMC (1982)
Measure Privacy by Anonymity (2002)
Measure Privacy and Utility by DP (2006)
Consider Adversary's Given Background Knowledge
Consider Adversary's Maximum Background Knowledge

Fig. 2.1 Development of Notion of Privacy.

2.1 Secure Multi-party Computation

Secure multi-party computation (SMC or MPC) is also known as secure function evaluation, and was first introduced by Andrew Chi-Chih Yao [174, 175]. SMC is a paradigm to define the security of a multi-party protocol/function without a trusted third party. As a result of the limited space in this thesis, we only provide a brief introduction to some of the important concepts and implementations of SMC here. For more details on SMC protocols and recent developments in both academia and industry, please refer to surveys, such as [24, 11, 40].

2.1.1 Toy Example and Informal Concepts

Prior to introducing the SMC paradigm, in this section, we provide a toy example that is a special case of SMC, where two parties are involved in the computation: Yao's *Millionaires Problem* [174], also known as secure two-party computation.

Problem description: Two millionaires, Alice and Bob, want to know who is wealthier, without revealing their actual wealth.

A simple solution: There are a huge number of solutions to this problem [174, 175, 68, 121, 83, 178, 86, 85] from different aspects. In this section, for simplicity, we apply Yao's original solution [174], where we have the following assumptions. The wealth (in millions) of Alice and Bob is represented by its integer part in a range of $[1,9]$, say Alice has I millions and Bob has J millions. A solid encryption scheme, such as, RSA [132], is applied in the communications between Alice and Bob, where Bob's public/encryption key is (e,n) and private/decryption key is (d,n). In particular, this solution contains three steps:

1. Alice chooses a random N-bit integer x, calculates $C = RSA_{(e,n)}(x)$, $m = C - I + 1$, and sends m to Bob.

2. Bob calculates a list Y, where $Y_i = RSA^{-1}_{(d,n)}(m + i - 1)$, $i \in [1,9]$, generates a list Z, where $Z_i = Y_i$, for $i \leq J$, $Z_i = Y_i + 1$; otherwise, chooses a random $N/2$-bit prime p for a

new list W, where $W_i = Z_i \mod p$, and sends W and p to Alice.

Notes:

- $Y_I = C$.
- Given that Bob does not know the value of I (Alice's wealth), C and x; hence, Bob cannot identify the value of I by $C = Y_I$.
- Bob does not send Z to Alice; otherwise, Alice will immediately know the value of J by the following operations. Alice calculates a list V, where $V_i = RSA_{(e,n)}(Z_i)$, and finds an index i that $V_i - V_{i-1} = 2$, then $J = i - 1$.
- Alice cannot have the value of Z by W and p, according to how Bob generates W.

3. Alice calculates $G = C \mod p$; if $G = W_I$, $I \leq J$; otherwise $I > J$. She shares the results with Bob.

Toy example summary. Clearly, the above operations and communications provide a positive answer to Yao's *Millionaires Problem*, that is, without a trusted third party, all that Alice and Bob receive in the communications only reveals the output about who is wealthier, yet cannot be used to deduce or infer the real value of their wealth, unless one of them does not faithfully follow the protocol.

From the above, we provide a more general description of SMC. SMC considers the following problem. Suppose m parties want to compute the value of function $f(x_1, x_2, \ldots, x_m)$, where x_i is a private integer number, owned by party P_i, within a given bound. Communications between each party are allowed. Is it possible for each P_i to have the value of $f(x_1, x_2, \ldots, x_m)$, yet does not know anything about x_j, $\forall j \neq i$? Roughly, SMC requires that a secure computation leaks nothing that can be learnt from the output of the computation. Once a protocol or function meets this requirement, the protocol or function is *secure*.

2.1.2 Security Definition via Ideal-World vs. Real-World Approach

In this section, we first introduce an ideal-world versus real-world approach [51, 17, 84], which provides clear security requirements for SMC. Then, based on the ideal-world versus real-world approach, we provide a formal definition regarding the *security* of a function in SMC.

Generally, a straightforward method to define *security* involving making a checklist containing items that violate our *security* requirements. However, it is difficult to confirm whether we enumerate all possible security violations, so such a method would not be reliable to define the *security*. To overcome this difficulty, people apply an ideal-world versus real-world approach, where we define *security* by making the outputs from a secure ideal world and the real world indistinguishable. Namely, in the ideal world, everything perfectly guarantees the *security*, so that we can define *security* in the real world with regard to this ideal world.

In short, the philosophy of defining *security* through an ideal-world versus real-world approach is two-fold. First, we create an ideal world for a multi-party computation. In this ideal world, there is a trusted mechanism (or third party) that receives input from each party, yet does not release those private inputs to anyone else. Then, this trusted third party produces and broadcasts the output of the joint computation. At the end of the above process, all parties only have their own input and final output; therefore, such joint computation in the ideal world is *secure*. Second, when having the same joint computation in the real world without the trusted third party, if a protocol produces an output that is equivalent (or indistinguishable) to the one from the ideal world, then such a protocol is *secure*. Figure 2.2 depicts the ideal-world versus real-world approach. On the left side of Figure 2.2, security is guaranteed by the existence of a trusted third party, T, in an ideal world. We require this trusted party T to only report the output of computation, yet not collude with any other parties. As a result, each P_i only learns the output of the joint computation $F_T(x_1, \ldots, x_n)$, without knowing anything about x_j, $\forall j \neq i$. In SMC, such a computation is *secure*. On the right side of Figure 2.2, there is no such trusted third party, T, in the real world. However, if we could have a joint computation protocol π, such that $F_\pi(x_1,\dots,x_n)$ achieves indistinguishable results as $F_T(x_1,\dots,x_n)$, then we say that such a protocol π is as *secure* as the trusted party T for the joint computation F.

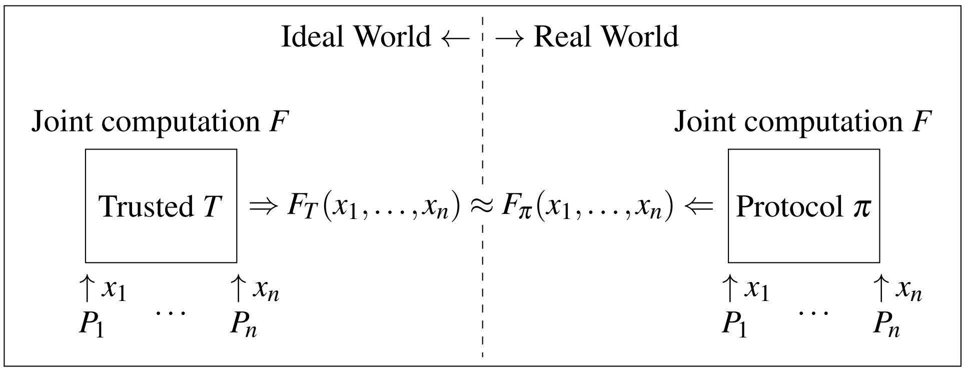

Ideal World ← → Real World
Joint computation F
Joint computation F
Trusted T
$\Rightarrow F_T(x_1,\dots,x_n) \approx F_\pi(x_1,\dots,x_n) \Leftarrow$
Protocol π
$\uparrow x_1$ P_1 $\cdots$ $\uparrow x_n$ P_n
$\uparrow x_1$ P_1 $\cdots$ $\uparrow x_n$ P_n

Fig. 2.2 Ideal World versus Real World.

Wensley et al. [161] further prove that a joint computation in the real world is as *secure* as the ideal world, if and only if less than one-third of all parties are colluding *semi-honest* adversaries (a.k.a Byzantine faults in distributed computing [161, 78]). Here, a *semi-honest* adversary is also known as an *honest-but-curious* (HBC) or *passive* adversary, which always follows the protocol honestly, yet also tries to infer the privacy from both the process and the output of the protocol. Colluding adversaries may collude to exchange information to learn more.

We now go to the formal definition of *security* based on the "indistinguishable" concept from the ideal-world versus real-world approach. We call such security **semi-honest security** since it is only secure against *semi-honest* adversaries.

Definition 2.1 (Security by Indistinguishability [51, 17, 9, 24, 40])**.** *A protocol π securely realises T against semi-honest adversaries in every possible subset Z, if there exists a* *polynomial time simulator S, such that, for any inputs* $(x_1,\ldots,x_n)$*, we have that:*

$$|\Pr[Real_\pi(k,Z,x_1,\ldots,x_n)=0]-\Pr[Ideal_{T,S}(k,Z,x_1,\ldots,x_n)=0]|$$

is negligible in k.

In Definition 2.1, we use the indistinguishable probability distribution to indicate the indistinguishability between the outputs from the real world and ideal world. Specifically, in the ideal world, T is the trusted third party who never colludes with any other parties, and S is a polynomial time simulator that simulates all possible attacks in the presence of a group of semi-honest adversaries Z. In the real world, we use the same inputs vector $(k,Z,x_1,\ldots,x_n)$ as the ideal world without the trusted third party, but only a protocol for the secure multi-party computing.

In addition, besides SMC, there are several classic security techniques: secret sharing [137, 119, 23], homomorphic encryption [116, 48, 156], oblivious transfer [128, 107, 69] and zero-knowledge proof [127, 50, 129]. Shamir et al. [137] first invented a secret-sharing scheme, which refers to methods for distributing a secret among a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of partial shares is combined; individual shares are of no use on their own. Homomorphic encryption [116, 48, 156] is a form of encryption that allows computations to be undertaken on cipher text, thereby generating an encrypted result that, when decrypted, matches the result of the operations performed on the plain text. In particular, additive homomorphic encryption and multiplicative homomorphic encryption are two famous techniques [48]. An oblivious transfer (OT) [128, 107, 69] protocol is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver, yet remains oblivious as to what piece (if any) has been transferred. A zero-knowledge proof [127, 50, 129] is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true.

2.1.3 Vulnerability of SMC to Adversaries with Auxiliary Information

Clearly, SMC is quite a strong security guarantee for data publishing/sharing in a distributed environment against *semi-honest* adversaries; however, it may fail to protect individuals' private information as a result of adversaries' background knowledge combining the published output of the computation. Here is a simple counterexample for the solution to Yao's *Millionaires Problem*. Suppose in the *Millionaires Problem* that Bob knows a smaller range of Alice's wealth, say, Alice's wealth $I \in [a,b]$. Bob takes values from $[a,b]$ linearly, as his value, J, to the protocol. Bob determines that Alice's wealth $I = J$ once Bob has $I \geq J$ and $I < J+1$. In this example, Bob, with some background knowledge, successfully infers Alice's wealth by providing serial inputs to the protocol. In summary, based on this counterexample, to preserve privacy against adversaries with auxiliary information, SMC is not a robust privacy-preserving solution.

2.2 Anonymity

From the previous section, we see that SMC is vulnerable to adversaries with some background knowledge. In this section, we briefly introduce another branch of privacy notion, anonymity, which addresses the weaknesses of SMC. Overall, anonymity retains the statistics of a set of data records, while guaranteeing that each record is indistinguishable from a number of other records by generalising and/or suppressing individual data to remove the explicit linkage to a specified individual, with minimal data modification. This way, anonymity approaches achieve a trade-off between individual privacy and data utility.

To date, researchers have proposed several approaches [150, 167, 95, 80, 170, 120, 153, 10] to implement anonymity according to the various background knowledges of adversaries. For example, in k-anonymity [150], (ρ, α)-anonymisation [10], incremental anonymisation [120] and m-invariance [170], the authors suppose that the adversaries have knowledge about an individual's quasi-identifier (i.e., non-sensitive information, such as postcode, gender and age). In the model of l-diversity [95] and (α, k)-anonymity [167], the researchers assume that the adversaries know the distribution of sensitive values (e.g., the morbidity of one disease in a given area). In the model of l-diversity [95] and (ρ, α)-anonymisation [10], the scholars presume that the opponents have knowledge about one person's appearance in the published data. In the model of t-closeness [80], Li et al. believe that the attackers hold the knowledge discovered from the published data. In the model of k^m-anonymity [153], Terrovitis et al. consider that the opponents are aware of m attributes of an individual.

In this section, we mainly focus on three key approaches for anonymity: k-anonymity [150], which is the first in the line of works; l-diversity [95], which is the first that considers the diversity of sensitive values in each anonymised block; and t-closeness [80], which is the first that considers the balance of distributions of sensitive values between an anonymised block and the whole dataset.

2.2.1 Fundamental Concepts in Anonymity

In this section, we introduce fundamental concepts and techniques that are widely used in the family of anonymity approaches, including relational table and generalisation and suppression techniques.

Relational Table. Originally, an anonymity approach was proposed for publishing a relational table, $T = T(A_1, \ldots, A_n)$, where A_i is the ith attribute of T. The table T is made up of finite tuples, for example, $t_i(A_1, \ldots, A_n)$ is the ith tuple in the table T. Each tuple represents an individual record. $t[A_i]$ indicates the value of attribute A_i in a tuple t.

For anonymous data publishing, the attributes of a table are important and can be split into three categories: *explicit identifiers*, *sensitive attributes* and *non-sensitive attributes*. Among these, explicit identifiers, such as name and ID number, are always removed prior to publishing because such attributes can identify a tuple from the table directly. Sensitive attributes should be kept in a published table for further data analysis, such as patients' disease or staff salary; however, the sensitive attributes must not be linked directly with a specific tuple without permission. To link sensitive attributes to an individual tuple, adversaries must have a *quasi-identifier* (first defined in [150]). A set of quasi-identifiers is a subset of non-sensitive attributes by which adversaries can identify one tuple uniquely without knowing the explicit identifier. Table 2.1 presents an example of a relational table.

Table 2.1 Example of Relational Table in Anonymous Data Publishing.

Explicit Identifier		**Non-sensitive Attributes**					**Sensitive Attributes**
		Quasi-identifier				Others	
Name	ID No.	Postcode	Age	Gender	Commitment Date	Charging	Salary
Alice	03071052	7500	19	F	5/May/2008	\$2	\$128 k
Bob	03071074	7100	27	M	1/Mar/2016	\$32	\$95 k

Generalisation and Suppression Techniques. Generalisation and suppression [136] are essential concepts used by many anonymous data publishing methods to achieve indistinguishability. Overall, generalisation changes a value $t[A_i]$ to be less specific yet still useful statistical information, while suppression removes a value/tuple from a dataset, which is equivalent to replacing a value with a symbol, such as a star. Figure 2.3 illustrates an example of generalisation and suppression used in anonymous data publishing, where Z_0 contains the raw value of attribute Z; Z_1 and Z_2 contain less specific values for attribute Z (i.e., generalised values); and Z_3 shows the result of suppression, that is, all values of attribute Z are removed from the published relational table.

$$\begin{array}{rl}
\text{Generalisation:} \left\{ \begin{array}{l} Z_0 = \{02138, 02139, 02140, 02141\} \\ \quad\downarrow \\ Z_1 = \{0213*, 0214*\} \\ \quad\downarrow \\ Z_2 = \{021**\} \end{array} \right. \\
\quad\downarrow \\
\text{Suppression:} \quad Z_3 = \{*****\}
\end{array}$$

Fig. 2.3 Generalisation and Suppression.

2.2.2 Famous Anonymity Approaches

We now consider the three most famous anonymity approaches: k-anonymity, l-diversity and t-closeness. In this section, we introduce each of these approaches in two parts: the target problem (i.e., the attack model) and the main idea (i.e., the definition).

k-Anonymity

- The target problem is linkage attacks.

  A **linkage attack** [151] aims to discover more knowledge about a record by linking the quasi-identifier of that record over different relational tables where all explicit identifiers of that record have been removed. The success of a linkage attack is based on a simple statistical property: given a set of quasi-identifiers of a person as the background knowledge of an adversary, an adversary could link different relational tables by those quasi-identifiers, such that the adversary learns more information about that person. Figure 2.4 shows an example of a linkage attack, where the circle on the left contains personal medical information, and the circle on the right contains electoral enrolment data. By linking these two tables with the quasi-identifiers, the adversary can easily erode a person's privacy, such as accessing disease and party affiliation information.

- The main idea:

  k-anonymity decreases the probability of identifying a particular person from 1 to at

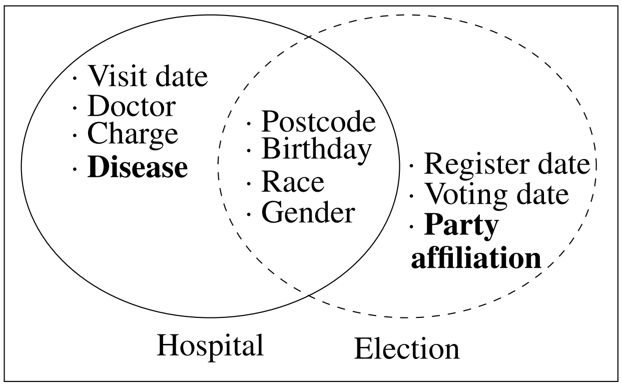


Fig. 2.4 Example of Linkage Attack.

most $1/k$ in linkage attacks. That is, for a give set of quasi-identifiers, there are at least k indistinguishable tuples for each tuple in a k-anonymised relational table.

Definition 2.2 (k-Anonymity [150]). *Given a relational table $T = \{tuples\}$, if T provides k-anonymity, then for each tuple $t_i \in T$, there exists at least $k-1$ tuples that have the same quasi-identifiers as t_i.*

Table 2.2 displays an example when processing a normal relational table to a 4-anonymity table. From Table 2.2, we see that the 4-anonymity table does not release any specific person's privacy against linkage attack, yet retains useful statistical information that can be used for further knowledge discovery, such as the relationship between people's age group and a given disease.

l-Diversity

- The target problem is that two attacks can increase the privacy inference probability by excluding some anonymised records in an anonymous block in a k-anonymity table with an adversary's background knowledge. This can include background knowledge attacks and homogeneity attacks.

  A **background knowledge attack** [94] identifies a record from a k-anonymity table using an adversary's background knowledge, such as one disease morbidity in a specific

Table 2.2 Example of 4-Anonymity.

(a) Original Table

No.	Quasi-identifier			Sensitive Attr.
	Postcode	Age	Ethnicity	Disease
1	1253	51	Chinese	Heart
2	1268	59	Japanese	Flu
3	1253	68	Indian	Flu
4	1268	80	British	Heart
5	1340	18	Spanish	HIV
6	1349	29	Chinese	HIV
7	1345	25	Korean	HIV
8	1343	2	Indian	HIV

(b) 4-Anonymity Table

No.	Quasi-identifier			Sensitive Attr.
	Postcode	Age	Ethnicity	Disease
1	12**	> 50	*	Heart
2	12**	> 50	*	Flu
3	12**	> 50	*	Flu
4	12**	> 50	*	Heart
5	13**	< 30	*	HIV
6	13**	< 30	*	HIV
7	13**	< 30	*	HIV
8	13**	< 30	*	HIV

ethnicity. For example, assume that Table 2.2b is published by hospital X. Alice knows that her Japanese friend is receiving medical treatment in that hospital. That is, Alice knows that her friend is in Table 2.2b. Note that Alice does not know which record of the table is her Japanese friend. By comparing information about her friend, such as postcode and age, with the anonymised information in Table 2.2b, she knows that her friend is in the block of $\{$Postcode: $12**$, Age $> 50\}$. Based on Alice's background knowledge that Japanese people have an extremely low morbidity of heart disease, Alice can further infer that her friend is possibly the second or the third record of that block. Since both records reflect someone with flu, Alice confirms that her Japanese friend has flu with higher probability from such background knowledge, which is a privacy breach.

A **homogeneity attack** [94] is a special case of background knowledge attack. With a homogeneity attack, the adversary only takes the quasi-identifier(s) as the background knowledge to infer the privacy of a record. For example, if the adversary knows that someone's quasi-identifier falls in the block of $\{$Postcode: $13**$, Age $< 30\}$ in Table 2.2b, then they can infer that this person must have HIV.

- The main idea:

  l-diversity introduces diversity in sensitive attribute(s) for each anonymous block of a k-anonymity table, so that we can further decrease the probability of privacy inference by background knowledge attack and homogeneity attack.

  Definition 2.3 (l-Diversity [95])**.** *Given a k-anonymity table $T = \{Blocks\}$, if T provides l-diversity, then, in each block, which is k-anonymity, there exist at least l kinds of sensitive identifiers in this block.*

  Table 2.3 presents an example of processing a normal relational table to a 3-diversity table (also a 4-anonymity table).

Table 2.3 Example of 3-Diversity.

(a) Original Table

No.	Quasi-identifier			Sensitive Attr.
	Postcode	Age	Ethnicity	Disease
1	1253	51	Chinese	Heart
2	1268	59	Japanese	Flu
3	1253	68	Indian	Flu
4	1268	80	British	Heart
5	1340	18	Spanish	HIV
6	1349	29	Chinese	HIV
7	1345	25	Korean	HIV
8	1343	2	Indian	HIV

(b) 3-Diversity Table

No.	Quasi-identifier			Sensitive Attr.
	Postcode	Age	Ethnicity	Disease
1	1***	$(20,60)$	Asian	Heart
2	1***	$(20,60)$	Asian	Flu
6	1***	$(20,60)$	Asian	HIV
7	1***	$(20,60)$	Asian	HIV
5	1***	$(0,20) \cup (60,90)$	Non-Asian	HIV
3	1***	$(0,20) \cup (60,90)$	Non-Asian	Flu
4	1***	$(0,20) \cup (60,90)$	Non-Asian	Heart
8	1***	$(0,20) \cup (60,90)$	Non-Asian	HIV

t-Closeness

- The target problem includes skewness attacks and similarity attacks.

  A **skewness attack** [81] infers privacy by the difference/skewness between the value distribution of sensitive attributes in an anonymous block and the whole anonymity table. For example, according to data from the World Health Organization (WHO), about 0.6% people worldwide are HIV positive [168]; however, in Table 2.3b, 50% of individuals are HIV positive. Such a big difference/skewness would provide more information to an adversary.

A **similarity attack** [81] infers privacy by the semantic closeness of sensitive values in an anonymous block. For example, an adversary could deduce that a person has eye problems if all the diseases in an anonymous block are eye-related diseases.

- The main idea:

  t-closeness maintains the balance between the distribution of sensitive values in each block and the distribution of sensitive values in the whole table. Namely, in t-closeness, the distribution of sensitive values in each block is as close as possible to the distribution of sensitive values in the whole table.

Definition 2.4 (t-Closeness [80])**.** *Given a k-anonymity table $T = \{Blocks\}$, if T provides t-closeness, then, for each k-anonymity block, the distance between the distribution of sensitive values in the block and the distribution of sensitive value in the whole table T is no more than a given t.*

In t-closeness, the threshold t is the key parameter. Li et al. [80] used the earth mover's distance (EMD) [134] for the distance calculation. The result of calculating EMD is the minimal amount needed to move from one distribution to another. Table 2.4 shows an example of building 0.278-closeness from 3-diversity.

Table 2.4 Example of 0.278-Closeness.

(a) 3-Diversity Table

No.	Postcode	Age	Disease
1	476**	2*	Gastric ulcer
2	476**	2*	Gastritis
3	476**	2*	Stomach cancer
4	4790*	> 40	Gastritis
5	4790*	> 40	Flu
6	4790*	> 40	Bronchitis
7	476**	3*	Bronchitis
8	476**	3*	Pneumonia
9	476**	3*	Stomach cancer

(b) 0.278-Closeness Table

No.	Postcode	Age	Disease
1	4767*	< 40	Gastric ulcer
8	4767*	< 40	Pneumonia
3	4767*	< 40	Stomach cancer
4	4790*	> 40	Gastritis
5	4790*	> 40	Flu
6	4790*	> 40	Bronchitis
7	4760*	< 40	Bronchitis
2	4760*	< 40	Gastritis
9	4760*	< 40	Stomach cancer

2.2.3 Vulnerability of Anonymity to Adversaries with Arbitrary Auxiliary Information

In summary, anonymity achieves a satisfying trade-off between individual privacy and published data utility against adversaries with specified background knowledge. However, there are two weaknesses in anonymity. First, each approach has pre-assumed background knowledge of adversaries in a very specific range, for example, k-anonymity only preserves privacy against linkage attacks, but not homogeneity attacks [95]. Second, although all anonymity approaches can measure privacy based on privacy disclosure probability, the utility of the published anonymous table cannot be quantified. For example, k-anonymity ensures that the probability of an adversary discovering private information about an individual is at least $1/k$; however, we do not have any utility guarantee when using a k-anonymity table for data analysis. To preserve privacy against adversaries with arbitrary auxiliary information, while measuring the trade-off between privacy and utility, anonymity is not a suitable notion of privacy.

2.3 Differential Privacy

To address the problems of anonymity approaches, Dwork et al. [37] proposed differential privacy (DP) to measure the trade-off between privacy and utility against adversaries with maximum auxiliary information. DP is one of the most widely adopted notions of privacy in the current research field [114, 96, 32, 57, 16, 5, 53, 60, 6, 115, 152, 28, 90]. The concept of DP was first introduced and defined by Dwork et at. [37] as ε-indistinguishability, then formalised in [33] as ε-differential privacy. Informally, DP is a scheme that minimises the sensitivity of output for a given function on two neighbouring datasets that differ by one record. In this manner, DP achieves indistinguishability between the outputs produced by two neighbouring datasets, so that the presence or absence of any record in a dataset, that is, the difference between the two neighbouring datasets, will be concealed to an adversary.

In this section, we introduce DP from three aspects. First, we consider a standard definition of DP and the fundamental concepts, including the privacy budget, composition properties of the privacy budget and function sensitivity. Second, we consider three important mechanisms or frameworks that implement DP for different computational tasks. Third, we discuss the most commonly used relaxation of DP.

2.3.1 Fundamentals of Differential Privacy

Given a trusted data curator who has a dataset X, and an adversary who has a dataset X', DP assumes that X' contains all records except an arbitrary one in X, that is, $|\, \|X\| - \|X'\| \,| = 1$, let $X \setminus X' = x_0$. We call such a pair of datasets, X and X', neighbouring datasets. With such a dataset X', the adversary has maximum background knowledge about the dataset X because the adversary only needs one more atomic record to have the full dataset X.

Standard Definition and the Privacy Budget

Basically, DP turns a function f to a randomised mechanism $\mathscr{T}$ by injecting random noise into either input(s) or output(s) or the computing process of f, so that the outcomes of $\mathscr{T}$ from X and X' are indistinguishable within a given bound based on a DP parameter: the privacy budget ε. The privacy budget measures the ratio of probabilities of returning the same outputs from both X and X'. We now provide the earliest formal and standard definition of DP by Dwork et al. [33].

Definition 2.5 (ε-Differential Privacy [33])**.** *A randomised mechanism $\mathscr{T}$ is ε-differentially private if and only if, for all neighbouring datasets X and X' and for all output sets $S \subseteq Range(\mathscr{T})$, $\mathscr{T}$ satisfies:*

$$\exp(-\varepsilon) \times \Pr[\mathscr{T}(X') \in S] \leq \Pr[\mathscr{T}(X) \in S] \leq \exp(\varepsilon) \times \Pr[\mathscr{T}(X') \in S], \tag{2.1}$$

where ε is the privacy budget.

According to Definition 2.5, it is clear that a small ε indicates a high level of indistinguishability, and hence good privacy-preserving performance; however, this is equivalent to bad output utility performance. In the extreme case, when $\varepsilon = 0$, $\mathscr{T}(X)$ and $\mathscr{T}(X')$ are completely indistinguishable. Therefore, in practice, the trusted data curator should set a predefined privacy budget according to the requirements for privacy and utility. Usually, the privacy budget is publicly available, so that the privacy performance is agreed upon by all parties involved in function f.

Composition of the Privacy Budget

When working with complicated cases, such as several differentially private mechanisms collaborating, there are two composition properties of the privacy budget used to analyse the overall privacy performance: *sequential composition* (Theorem 2.1) and *parallel composition* (Theorem 2.2) by McSherry et al. [99].

Theorem 2.1 (Sequential Composition of Privacy Budgets [99]). *Given a dataset X, a set of mechanisms where each $\mathscr{T}_i$ satisfies ε_i-DP, the sequential composition of $\mathscr{T}_i(X)$, $\mathscr{T}(X)$, is $\sum_i \varepsilon_i$-differentially private.*

Based on Theorem 2.1, when we sequentially compute multiple differentially private mechanisms over one dataset, the privacy-preserving performance of the sequential composition would be worse than an individual mechanism. To attain a satisfying overall privacy performance, we must set a higher privacy requirement for each component of the sequential composition.

Theorem 2.2 (Parallel Composition of Privacy Budgets [99]). *Given a dataset $X = \cup_i X_i$ where $X_i \cap X_j = \emptyset, \forall i, j$, a set of mechanisms where each $\mathscr{T}_i$ satisfies ε_i-DP, the parallel composition of $\mathscr{T}_i(X_i)$, $\mathscr{T}(X)$, is $\max_i\{\varepsilon_i\}$-differentially private.*

Based on Theorem 2.2, when we compute multiple functions in parallel over mutually disjoint datasets, the overall privacy-preserving performance is the same as the worst privacy-preserving mechanism. Thus, to attain a satisfying overall privacy performance, we only need to take care of the worst case among all parallel functions.

Sensitivity

Although the privacy budget bounds the DP guarantee, it is impossible to achieve the same privacy-preserving performance with the same privacy budget for all functions, even with the same input dataset X. Intuitively, a function on a given dataset X, which is more sensitive to an adversary with maximum background knowledge about X, needs more noise to conceal the privacy of the input data. To achieve this, Dwork et al. [37] defined the *sensitivity* of a function f, Δf, as the difference between $f(X)$ and $f(X')$. The sensitivity actually measures the effect of the adversary's auxiliary information used to infer privacy. The aim of injecting noise in DP is to make the sensitive result $f(X)$ insensitive and hide the difference between X and X' to preserve privacy. Two function sensitivities are widely used for DP, *global sensitivity* [37] and *local sensitivity* [114], according to how extensive the adversary's background knowledge could be.

Global sensitivity considers the globally greatest difference between $f(X)$ and $f(X')$, that is, the global worst-case of the adversary's auxiliary information, or the maximum difference between $f(X)$ and $f(X')$ over all possible $X \subset \mathit{Range}(X)$. Equation (2.2) [37] displays how to calculate the global sensitivity.

$$\Delta f_G(X) = \max_{\forall X, |\|X\| - \|X'\|| = 1} |f(X) - f(X')|. \tag{2.2}$$

In contrast, local sensitivity considers the locally greatest difference between $f(X)$ and $f(X')$, that is, the local worst-case of the adversary's auxiliary information, or the maximum difference between $f(X)$ and $f(X')$ over all possible X' as the neighbouring dataset of X.

Equation (2.3) [114] displays how to calculate the local sensitivity.

$$\Delta f_L(X) = \max_{\forall X', |\|X\| - \|X'\|| = 1} |f(X) - f(X')|. \tag{2.3}$$

Note that global sensitivity is also the maximum value among all the local sensitivities of a given pair of f and X. When the privacy-preserving requirement is not strict, to decrease the noise that needs to be injected, we can use Δf_L as the value for Δf.

$$\Delta f_G(X) = \max_{\forall X} \{\Delta f_L(X)\}. \tag{2.4}$$

We illustrate the difference between global sensitivity and local sensitivity by an example shown in Table 2.5. As we can see, with a given dataset, different functions would have different sensitivities. For function *Max*, $\Delta f_G = 9$, when $X = \{10, 1, 1, 1, 1\}$, $X' = \{1, 1, 1, 1\}$; $\Delta f_L = 1$, when $X' = \{1, 2, 3, 4\}$. For function *Sum*, $\Delta f_G = 10$, when $X = \{10, x_1, x_2, x_3, x_4\}$, $X' = \{x_1, x_2, x_3, x_4\}$; $\Delta f_L = 5$, when $X' = \{1, 2, 3, 4\}$.

Table 2.5 Example of Sensitivity in DP.

Dataset	Function	Δf_G	Δf_L
$X : \{1,2,3,4,5\}$	*Max*	9	1
$x_i \in [1, 10]$	*Sum*	10	5

In addition, from parallel composition from Theorem 2.2, we should be aware that the function sensitivity for each $\mathscr{T}_i$ is calculated with each disjoint dataset x_i, that is, $\Delta f(x_i)$; while the function sensitivity for the overall mechanism $\mathscr{T}$ is calculated with the whole dataset X, that is, $\Delta f(X)$.

Figure 2.5 depicts how DP changes the distribution of the function output (to guarantee DP as Definition 2.5) by injecting random noise, where $^a/_b = \exp(\varepsilon)$, that is the bound of DP.

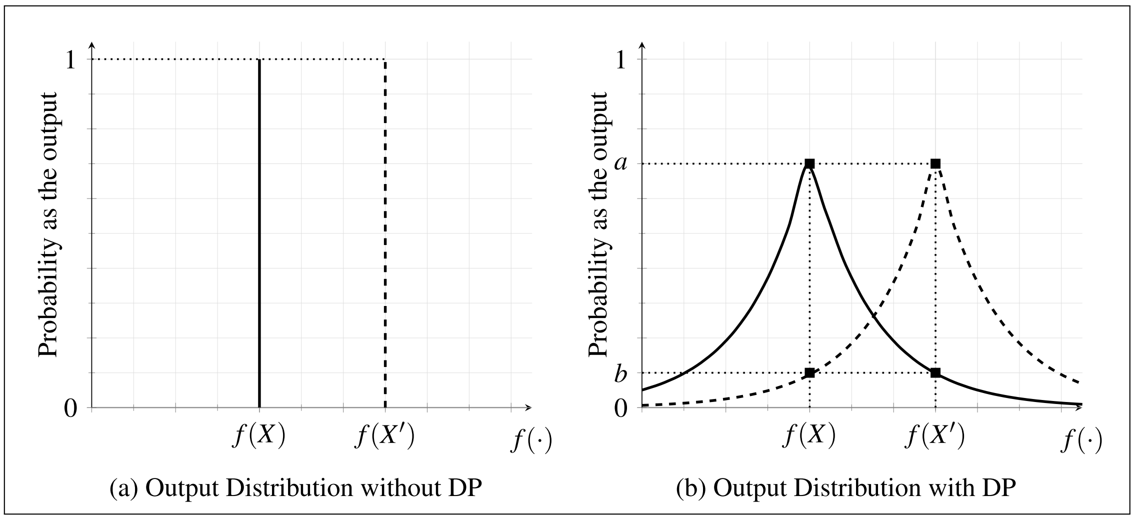


Fig. 2.5 General Idea of Differential Privacy.

2.3.2 Mechanisms of Differential Privacy

In this section, we present three mechanisms or frameworks of DP used in the state-of-the-art: the Laplace mechanism [33], the exponential mechanism [100] and the sample and aggregation framework [114].

Laplace Mechanism

The Laplace mechanism of DP (LapDP) [37] was the first mechanism to implement DP in a form suitable for numeric computing, such as Max/Min, Sum/Counting. The key idea of LapDP is adding random noise, following a Laplace distribution, to numeric values. Following the definition of probability density function (PDF), we cannot determine the probability of an arbitrary random variable x. Thus, to apply a Laplace distribution to LapDP, we use its probability density function (PDF, mean is zero) to calculate the sampling weight for a given noise value x:

$$\omega(x) = \frac{1}{2b}\exp(-\frac{|x|}{b}),$$

where b is the scale parameter in Laplace distribution [163]. In DP, $b = {}^{\Delta f}/_{\varepsilon}$. We define LapDP in Definition 2.6.

Definition 2.6 (LapDP [37]). *Given a function $f : X \to \mathbb{R}$, $\mathscr{T}(X)$ is ε-differentially private if*

$$\mathscr{T}(X) = f(X) + Lap^{-1}(\frac{\Delta f}{\varepsilon}), \tag{2.5}$$

where Δf is the sensitivity of function f and ε is the privacy budget.

Exponential Mechanism

The exponential mechanism of DP (ExpDP) [100] is an implementation of DP used for non-numeric computing, e.g., recommending an item from a set of items. The key idea of ExpDP is assigning a weight for each candidate according to its likelihood, and then weighted sampling the preferred items from the candidates. To assign weights, ExpDP uses a quality function, $q(X,x)$, which reflects how appealing the pair (X,x) is, where X denotes a dataset and x is the random response to a query function on the dataset X. Definition 2.7 formally defines ExpDP.

Definition 2.7 (ExpDP[100]). *Given a quality function of a dataset X, $q(X,x)$, which reflects the quality of a query response x, the exponential mechanism $\mathscr{T}$ provides ε-DP, if $\mathscr{T}(X) = \{\omega(x) = \exp(\frac{\varepsilon \cdot q(X,x)}{2\Delta q})\}$, where Δq is the sensitivity of $q(X,x)$ and ε is the privacy budget.*

Sample and Aggregation Framework

The sample and aggregation framework of DP [114] is proposed to further decrease the value of local sensitivity to improve the utility of the computing output. The key idea of sample and aggregation is computing the function f over repeatedly sampled subsets of the original dataset X, and then aggregating the multiple sub-results using DP. In the sample and aggregation framework, the local sensitivity of the final aggregation step is usually less than the location sensitivity over the whole dataset X, as it is unlikely to sample all the items of X in the sampling step.

The most famous application of sample and aggregation framework is GUPT k-means clustering. Mohan et al. [104] proposed GUPT, a clustering algorithm that applies the sample and aggregation framework based on Lloyd's k-means clustering algorithm [87]. Briefly, GUPT uniformly samples items from an input dataset to different buckets, where the local clustering result of each bucket is generated by Lloyd's algorithm. The final clustering result is the mean of the local ones with Laplace noise applied.

2.3.3 Relaxation of Differential Privacy

Given that the standard DP is too strong, in terms of the adversary's background knowledge, for some applications (as in the real-life scenario), the worst-case assumption regarding the adversary's background knowledge is rarely achieved or it is rare to have some of the possible dataset from the required range; hence, it is natural to consider relaxing the standard DP. In this section, we briefly discuss two famous relaxations of standard DP: (ε, δ)-DP [36], which includes an additional probability of error and (α, ε)-Rényi DP (RDP) [102], which considers the differentially private bound for two arbitrary datasets.

In (ε, δ)-DP, this additional δ allows the output privacy loss to be larger than the bound $\exp(\varepsilon)$ in the standard DP. A Gaussian distribution is often used to achieve (ε, δ)-DP, as described in Definition 2.8.

Definition 2.8 ((ε, δ)-Differential Privacy [36]). *A randomised mechanism $\mathscr{T}$ is (ε, δ)-DP if and only if for all neighbouring datasets X and X' and for all output sets $S \subseteq Range(\mathscr{T})$, $\mathscr{T}$ satisfies:*

$$\Pr[\mathscr{T}(X) \in S] \leq \exp(\varepsilon) \times \Pr[\mathscr{T}(X') \in S] + \delta, \tag{2.6}$$

where ε is the privacy budget and δ is a small non-zero item.

(α, ε)-RDP aims to provide a differentially private bound for two arbitrary datasets, rather than two neighbouring datasets, to relax the standard DP. To measure the difference (to bound it by the privacy budget ε) between the two datasets, drawing from two different distributions, (α,ε)-RDP extends Rényi divergence [131] as defined in Definition 2.9.

Definition 2.9 (Rényi Divergence of Order α [131])**.** *Given two distributions P, Q, and a positive integer, α, the Rényi divergence of order α is*

$$D_\alpha(P||Q) = \frac{1}{\alpha-1}\log\left(\sum_{i=1}^{n}\frac{P(x)^\alpha}{Q(x)^{\alpha-1}}\right),$$

where $P(x)$ $(Q(x))$ is the density of P (Q) at x.

Based on Definition 2.9, we have the definition of (α,ε)-RDP as follows.

Definition 2.10 ((α,ε)-Rényi DP [102])**.** *A randomised algorithm f is (α,ε)-Rényi DP for any two neighbouring datasets X and X', if*

$$D_\alpha(f(X)||f(X')) \leq \varepsilon.$$

Note that, when $\alpha=\infty$, (∞,ε)-RDP is the standard ε-DP for the X and X', such that distributions in Definition 2.9 satisfy $P = f(X)$ and $Q = f(X')$.

Additionally, there are several other proposed relaxations of standard DP, such as, concentrated DP [35] and pufferfish privacy [74] (or, a similar concept, blowfish privacy [60]).

2.4 Summary

This chapter has introduced the related work and ideas on privacy. It has identified that indistinguishability is the core idea behind the three notions of privacy. In particular, it has identified that SMC defines secure computing by achieving indistinguishability between outputs from the ideal world and real world, anonymity measures privacy by preventing any individual from being distinguished against a group of individuals for an adversary with given background knowledge, while DP measures the trade-off between privacy and utility by implementing indistinguishable outputs from two neighbouring datasets to conceal an arbitrary item against an adversary who potentially has maximum auxiliary information. As a result of the adjustable bound in the worst-case assumption of the adversary, we use ε-DP as the notion of privacy for the remainder of this thesis, including discussing the state-of-the-art applying DP in data analytics in Chapter 3 and introducing the proposed DP approaches/algorithms addressing problems of existing work in Chapters 4, 5, 6 and 7.

Chapter 3

Literature Review

There are several major computing tasks in the process of data analytics, such as data aggregation (for collecting raw data), data publishing (for sharing data with the public for further research) and data mining/machine learning (for gaining insights from the data). However, because of privacy disclosure threats [66, 149, 159, 2, 154, 155], researchers have studied privacy preservation for such computing tasks. In this chapter, we briefly summarise the state-of-the-art approaches in differentially private data analytics. In particular, we review differentially private data aggregation, differentially private path publishing as a special application of data publishing and differentially private *k*-means clustering, as representatives of typical computing tasks.

We first review differentially private data aggregation with security guarantees [130, 140, 1, 38, 52, 39, 122] with a focus on the privacy of the final aggregation results against the strongest inference attacks and the security in the data collection process, where distributed data curators share their private data with other data curators or data aggregator(s). To guarantee both (output) privacy and (computing) security, according to our discussion in Chapter 2, DP and SMC are the two most commonly used tools in existing work. In addition, when implementing DP to preserve privacy for data aggregation in distributed environments, each data curator shares differentially private data (i.e., their data plus DP noise added) to address privacy concerns. Given that the DP noise is not added at a central server/data aggregator, it is necessary to consider the overall privacy budget guarantees of the whole distributed data aggregation, and hence consider the parallel composition of DP privacy budget.

We review differentially private data publishing by studying a special application, differentially private trajectory publishing, which aims to preserve privacy in trajectory data (e.g., the existence of a given location/road in the published trajectory) by considering some of the special features of trajectories, such as the movement direction or the spatiotemporal relationship. There are two main scenarios in this field: publishing a single trajectory [71, 171, 61, 62, 113, 158, 157, 173, 54, 20] and publishing a set of trajectories [21, 138, 65, 19, 58, 3, 25]. In this thesis, we focus on the former.

Differentially private k-means clustering aims to gain insights from unlabelled data, while preserving privacy in the worst case of adversaries with potentially extensive background knowledge (see Chapter 7 for details of the worst-case adversary background knowledge). Most existing work [12, 34, 104, 176, 112, 147, 148, 118] achieves DP in an interactive setting by injecting DP noise into each iteration of the k-means clustering to guarantee the privacy of every single step of the clustering.

3.1 Background on Differentially Private Data Aggregation with Security Guarantees

In this section, we review the current work on differentially private data aggregation with security guarantees [130, 140, 1, 38, 52, 39, 122]. Figure 3.1 depicts the general idea of this topic, where x_i is the raw data of a data curator i, δ_i is the DP noise, f is the aggregation function, and y is the differentially private aggregation output.

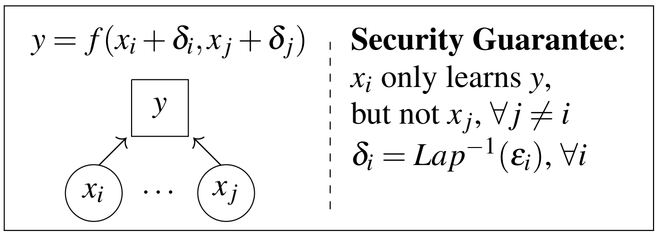

$y = f(x_i + \delta_i, x_j + \delta_j)$
y
x_i
$\cdots$
x_j
Security Guarantee:
x_i only learns y,
but not x_j, $\forall j \neq i$
$\delta_i = Lap^{-1}(\varepsilon_i)$, $\forall i$

Fig. 3.1 General Idea of Differentially Private Data Aggregation with Security Guarantees.

The computational task of data aggregation typically focuses on linear calculations of data that are allocated at distributed data curators. In this thesis, we study summation/counting function as a special application of linear calculations following existing studies in [99, 130, 140], since summation/counting function acts an important role in learning the statistics of aggregated data. There are two main branches of existing work. First, where there are multiple data curators/contributors that share data to one data aggregator [130, 140, 1, 38, 52]. Second, where there are multiple data curators/contributors that jointly compute the output of aggregation without a data aggregator [39, 122]. Given that we collect and aggregate data from distributed data contributors, to guarantee the security of computation and the privacy of each individual's data, a combination of SMC and DP is applied. In particular, the SMC paradigm is applied in the joint computation when sharing private data with other data curators to guarantee the security of computation, while DP is used for preventing privacy disclosure from the resultant output.

Rastogi et al. [130] proposed the first schemes to ensure private and secure data aggregation based on the combination of SMC and DP, called PASTE. PASTE is an approach that allows a semi-honest central server to compute the sum of values from distributed data curators (also semi-honest data curators) securely without privacy disclosure. PASTE entails the following steps. First, each data curator encrypts the sum of its own data and LapDP noise. It then sends the encrypted sum to the data aggregator. Note that the sum of the noise guarantees DP for the sum of the distributed data; however, the distributed noise cannot guarantee DP for the distributed data. Second, based on homomorphic encryption, the aggregator sums up the received data and sends them back to each data curator. Third, the data curators decrypt the message and send it back to the data aggregator. So far, the final output is secure and differentially private. However, once the central server has any data curator's decryption key, the privacy of that data curator is disclosed. Further, if some of the data curators collude with each other, because the LapDP noise with each data curator does not preserve individual privacy, the individual data are also not guaranteed to be differentially private.

Similar to [130], Shi et al. [140] proposed an addition-based encryption scheme that guarantees security (for the data aggregation) and DP (for each data curator) while achieving high data utility. The key point in [140] is applying the Diffie-Hellman key exchange protocol [27], by which two entities jointly generate the encryption keys together. Every data curator keeps r encryption keys, which are paired with the keys in other r entities. As a result, the data aggregator's decryption key is zero minus the sum of the data curators' keys, which guarantees that the aggregator learns nothing (not even the data curator's key) different than the final output. However, this scheme is vulnerable to collusion attack in the worst case, that is, if $n-1$ data curators work together (minus their sum from the overall output), they can determine the victim's private information with high probability.

Ács et al. [1] proposed a scheme based on requesting a set of computation entities to send encrypted and noise-added private data to a semi-honest data aggregator periodically. This method is similar to [130], yet assumes the data curators are *dishonest-but-non-intrusive*, that is, such adversaries may not follow the protocol and may collude with someone, yet do not modify the networks. The data aggregator in this scheme can only learn the answer to a query, but nothing else, because of the security guarantees of homomorphic encryption in each period. With such an assumption about adversaries, each data curator adds LapDP noise to preserve its own privacy in [1] (based on their own local sensitivity), rather than preserving the privacy of aggregated data, as in [130] (based on the local sensitivity of the given aggregated data or the global sensitivity of the aggregation function). However, if the central server is compromised, then the decryption key of any data curator is vulnerable; hence, the individual privacy would not be safe any longer.

Following [140] and [1], Goryczka et al. [52] provided an enhanced fault-tolerant (EFT) scheme, which is a fault-tolerant version of [1]. Given the similarity between [52] and [1], we only consider the extended part of [52], i.e., the fault-tolerant function of the EFT scheme. In particular, to achieve fault tolerance, in [52], the data aggregator is able to detect any faulty data curator, e.g., where they might be disconnected. Once there is a faulty data curator, all the neighbouring data curators remove the value received from the faulty one. In the worst case, if all neighbours of a party i are faulty, then sending the recovery key to the aggregator would reveal the contributed value x_i. In such a case, the party i subtracts x_i from the recovery key, which will remove it from the aggregated result as well. This fault-tolerant scheme recovers the status of data aggregation by removing the faulty data, rather than providing an estimated value of the overall aggregation utility. Therefore, this method will not guarantee the utility of the final aggregation output.

Eigner et al. [38] proposed a three-layer data aggregation protocol, PrivaDA, where data curators are placed in the first/bottom layer, computing servers are in the second layer and a data aggregator is in the third/top layer. They assumed that the data curators and data aggregator may collude and be corrupted, and the computing servers are semi-honest. Based on such assumptions of the adversary model, data are aggregated in the following three steps. First, the data curators split their data into multiple pieces according to the number of computing servers. Second, the computing servers receive and aggregate each encrypted piece of data individually with LapDP noise, and then send the local outputs to the data aggregator. Finally, the data aggregator reports the aggregated result. In addition, the security of PrivaDA is guaranteed by the Shamir secret-sharing scheme [137], which is applied when the data curators share their pieces of data, while the privacy is ensured by LapDP. However, as the data curators send encrypted original data to the computation servers, once these semi-honest computing servers are compromised or collude, the privacy guarantees at the second layer of PrivaDA are lost.

The above works aggregate data with the help of a data aggregator. Given that the data aggregator may be vulnerable to different types of attacks, such as collusion attack, other works aggregate the data in a fully distributed manner, that is, without a central data aggregator. Elahi et al. [39] proposed PrivEx to securely learn statistical information from the data of distributed data curators. PrivEx achieves DP by injecting Gaussian noise into the original data collected from the data curators. Secure information sharing is implemented with two schemes. The first scheme is based on secret sharing, which is similar to PrivaDA [38], whereby information is aggregated through a three-layer network comprising data curators, tally key servers and a data aggregator. The second scheme does not contain a data aggregator at the third layer. Instead, it uses a public bulletin board to save the results and keys from data curators. Note that, in the second scheme, the public bulletin board does not serve as a computational party in the data aggregation; therefore, this scheme only contains two layers, data curators and tally key servers, which check the correctness of data from data curators by zero-knowledge proof. PrivEx assumes that at least one data curator should be honest, which is weaker than PrivaDA.

In contrast to the above, Pettai et al. [122] studied the cost of combining DP and SMC using the sample and aggregation framework of DP because of its statistical functions, such as counting and arithmetic means. This method first applies a secret-sharing scheme, Sharemind [13], to split and share pieces of input with all involved parties. Then each party returns its local output with a fixed threshold to avoid data leakage. Finally, all parties aggregate the output for the final result. The major result of [122] shows that, for almost all arithmetic operations—including summation, division, square root, logarithm and bit shift—adding DP on top of SMC only introduces additional linear time complexity, which is an acceptably extra cost for SMC. However, without a constant number of data aggregators, this full connection network would result in $\mathcal{O}(n^2)$ time complexity for communication, where n is the number of data curators, which is significantly higher than other methods using constant number (c) of data aggregators, where the time complexity for communication is $\mathcal{O}(cn) = \mathcal{O}(n)$.

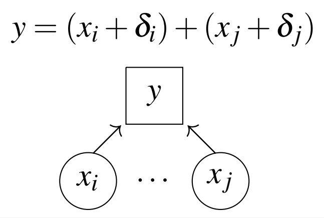
In summary, numerous researchers have applied the combination of SMC and DP schemes into privacy-preserving data aggregation, so that the aggregated result is both secure and differentially private. However, the existing work suffers from at least one of the following problems. First, potential privacy disclosure risk against malicious data curators (or aggregators) exists because of collusion attacks or a potential malicious central server. Second, a high communication cost may occur because of the fully connected network for information sharing without a central server. Therefore, in this thesis, we propose an approach that ensures both security and DP for distributed data aggregation, while retaining a linear communication cost against collusion attacks by malicious data curators and/or malicious server(s).

3.2 Background on Privacy Budget Guarantees of Distributed Systems

Following the privacy-oriented data aggregation approaches from the previous section, we further study the overall DP guarantee of a data aggregation system, where each individual data curator shares differentially private data based on their local privacy requirements. In DP, this problem is represented by the parallel composition of privacy budget.

We review the results [99, 140] in analysing the parallel composition of the privacy budget of a distributed differentially private summation system for the overall privacy budget guarantee. Figure 3.2 displays the general idea, where x_i is the raw data of data curator i,

δ_i is DP noise, ε_i is the privacy budget of data curator i, y is the aggregation result, ε is the overall privacy budget of the whole aggregation system, and g is the function mapping privacy budgets of individual data curators and the overall privacy budget of the aggregation result. Given that function g does not depend on any specific network topology, for simplicity, we use such an n to 1 star network in this thesis.



Parallel Composition of Privacy Budget:
Let $y = (x_i + \delta_i) + (x_j + \delta_j) = (x_i + x_j) + \delta$
i.e., $\delta = \delta_i + \delta_j$
$\because \delta = Lap^{-1}(\varepsilon)$, $\delta_i = Lap^{-1}(\varepsilon_i)$, $\forall i$
$\therefore g? \Rightarrow \varepsilon = g(\varepsilon_i, \varepsilon_j)$

Fig. 3.2 General Idea of Privacy Budget Guarantees of Distributed Systems.

There are two ways to preserve DP for such a system: implement DP bottom-up or top-down. For the former, we require each data curator to share differentially private data according to their own requirement for privacy preservation. We call such an implementation *local DP* in this thesis. For the latter, instead of injecting DP noise at each data curator, we ask the data aggregator to collect the raw data from the data curators, and then compute a differentially private aggregation output. We call such an implementation *global DP* in this thesis. Note that the local DP and global DP preserve DP against different adversaries' background knowledge. In particular, the local DP guarantees DP in the case of one missing item in the data curator's local data, while the global DP guarantees DP in the case of one missing data curator among all data curators. The global DP works against the strongest collusion attacks, where $n-1$ out of n data curators potentially collude with each other.

Based on the amount of injected noise, the final aggregated result could be both locally and globally differentially private. That is, there is a privacy budget for the global DP that has equivalent privacy performance to the composition of the local DP of each data curator. Therefore, our goal is to find the corresponding global DP privacy budget that matches the performance of the local DP composition. In particular, in this section, we review existing results on such summation functions [99, 140].

From the previous discussion, we know that the combination of SMC and DP can guarantee both the security of the computing process and the privacy of computing outcomes for distributed data aggregation systems. However, such a combination has two main problems: unknown guarantee of DP for the whole system (i.e., global DP) and the potentially high computational cost due to SMC. Specifically, the reason for the unknown guarantees of global DP is that these methods do not differentiate local DP and global DP in terms of the different assumptions related to adversaries' background knowledge, that is, one missing record in the dataset of each data curator (considered by local DP) or one missing data curator among all the data curators (considered by global DP). Thus, in existing work, when the authors prove that their scheme is differentially private globally, a Δf from each data curator's local DP is often used instead of the Δf of the whole distributed system.

Some researchers focus on analysing the privacy performance of global DP by ensuring the local DP of each data curator. McSherry [99] provided the first result on this topic by studying the property of DP in parallel composition (Theorem 2.2 in Chapter 2). The property of parallel composition ensures that, if each data curator i provides ε_i-DP locally, then the parallel composition of these data curators (like the data aggregation system in the last section) will have a $\max_i\{\varepsilon_i\}$-DP globally. This property is helpful for analysing the privacy performance of a distributed system, such as [172, 103, 88]. However, this parallel composition property introduces a condition in the proof. That is, $\Delta f_L(X) \geq \sum_i \Delta f_L(x_i)$, where $\Delta f_L(X)$ is the local function sensitivity for the data aggregator to have a $\max\{\varepsilon_i\}$-DP (global guarantee) and $\Delta f_L(x_i)$ is the local function sensitivity for each data curator to have ε_i-DP (local guarantee), where $X = (x_1, x_2, \ldots, x_n)$. When $\Delta f_L(X) < \sum_i \Delta f_L(x_i)$, the global DP guarantee is unknown in the parallel composition in [99]. Table 5.1 in Chapter 5 presents a counterexample of the conditions in [99].

Shi et al. [140] achieved $\sum_i \varepsilon_i$-DP globally when each data curator provides ε_i-DP locally without any such conditions. To do so, both the local and global DP use the same query function sensitivity $\Delta f = \Delta f_G$ where "G" is the global function sensitivity, defined in Equation (2.2). However, we observe two shortcomings of the result in [140]. First, using global sensitivity in this case can introduce more noise to conceal the potential difference between the outputs computed from two neighbouring datasets (one is ours having n records, the other is the attacker's having $n-1$ records). Second, the privacy budget for the global DP could be too large to have an acceptable level of data utility if the data curators set a large privacy budget for their local DP.

Based on the above analyses of existing work to guarantee distributed DP, we aim to provide a lower bound of privacy budget without conditions on any component of the system's local function sensitivity (in Equation (2.3) in Chapter 2) to guarantee enhanced privacy performance.

3.3 Background on Differentially Private Single-path Publishing

In this section, we briefly summarise the state-of-the-art work on differentially private single-trajectory publishing [71, 171, 61, 62, 158, 157, 173, 54, 20]. We consider both the advantages and disadvantages of these works. Figure 3.3 illustrates the general idea of existing works.

The existing differentially private single-trajectory publishing algorithms are robust to protect the privacy of vertices/locations of a given path/trajectory under potentially the same assumptions on an adversary's background knowledge, and often apply exponential mechanism of DP (ExpDP) [100] to support DP. In particular, they focus on location information as the privacy of the target trajectory, and assume the adversary does not know some locations

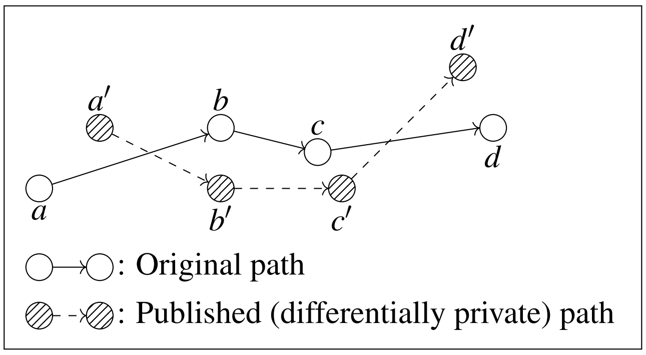


Fig. 3.3 General Idea of Differentially Private Single Path Publishing.

in a target trajectory. In the worst case, those existing differentially private single-trajectory publishing algorithms are resilient to the attacker that has $n-1$ out of n locations (and their connections) of the private trajectory (such background knowledge is considered the maximum background knowledge in DP). When achieving DP to preserve location privacy, the published trajectory may contain fake locations only, which are sampled via ExpDP from a pre-selected area to replace the real locations in the target trajectory. Researchers have developed different ways to decide on fake locations with ExpDP according to their objectives. These include the spatiotemporal relationships between locations [171, 173, 20], movement directions in a trajectory [71], and the similarity between real and fake locations [61, 62, 158, 79, 54, 93], e.g., [158, 79, 54] apply clustering-based strategies for determining fake locations.

The study in [71] presents one of the first differentially private algorithms for single-trajectory publishing. In [71], Jiang et al. proposed a sampling distance and direction (SDD) mechanism, which publishes a partially perturbed trajectory of a ship's movements on the sea. To retain the utility of the published trajectory, they assume that the starting/first location and the terminal/last location should be publically available. In the published trajectory, these two special locations are kept the same as that of the original trajectory. The remaining locations in the trajectory are determined by a noisy distance and a noisy direction based on the original adjacent locations. In particular, SDD samples the fake location $i+1$ with

ExpDP, according to the direction and distance from the fake location i to the real location $i+1$. Additionally, to ensure utility and privacy, the distance between any two adjacent positions is bounded by the speed of the ship; otherwise, it would be easy for an adversary to infer the fake locations.

Similar to [71], He et al. [61, 62] proposed a differentially private trajectories (DPT) system to select the most similar/possible locations in a given area to replace the real locations in a trajectory. The DPT system was achieved using three key components: hierarchical referencing, model selection and direction weighted sampling. Hierarchical referencing first splits the map into equal-sized grids. According to the resolution (the size of the grid cells), the DPT system creates multiple maps for the given trajectory. If there are real locations of the trajectory that appear in a grid, then the centre of the grid will be used as a fake location in the map. Doing so generates a group of possibly fake trajectory candidates. Afterwards, model selection selects a subset of the fake trajectories using the grid-based injected DP noise. Following this step, some of the most unlikely trajectories, which are easy to be filtered, are removed. Finally, direction weighted sampling adjusts the directions of each location in the fake trajectory to meet the directions in the original trajectory, and then a fake trajectory is selected randomly to publish the final DP trajectory.

Wang et al. [158, 157] proposed a private trajectory calibration system, which applies a clustering-based method to find representatively fake locations to replace the real ones for the published trajectory. In this system, as an initialisation step, frequently visited locations of a given user, whose trajectory will be published then, are clustered. Then an anchor location in each cluster is chosen as the representative of the cluster. In [158], the anchor location in a cluster could be the most visited location, such as the city council or the physical centroid of the cluster. After generating the anchor point based on the visiting frequency, LapDP noise is injected into the visit counts of the locations. For a given trajectory of a user, each location in this trajectory is replaced with its nearest anchor location, while considering the spatiotemporal relationship between adjacent locations to finally form the published trajectory.

Li et al. [79] proposed a differentially private trajectory analysis for points-of-interest (POI) recommendation based on a clustering approach. In general, this approach builds up differentially private connections between an individual user and POI for further recommendations. It first selects some of the POI on the trajectories of each user, and then clusters those POI based on their physical closeness to several super locations, called *hub*s in [79]. Finally, a matrix of user–POI with LapDP noise is published as the final recommendation data. In the user–POI matrix, each row indicates the perturbed visit records of a user; hence, each cell indicates whether a user visits a POI or not. Further, [79] assumes that one atomic cell in the user–POI matrix is unknown for an adversary, which is the maximum background knowledge under the DP framework because, in this case, the authors assumed that one arbitrary user's full trajectory is not known by the adversary. Note that the atomic cell in the matrix is actually a location in a trajectory, which is similar to assumptions in other differentially private trajectory publishing algorithms.

Gu et al. [54] proposed a clustering-based approach for single-trajectory publishing that is similar to [158]. It differs from [158] by providing a protected locations-oriented approach, rather than clustering any location to a map. Specifically, for each location in the trajectory, they first generate a cluster, whose centre is the protected location, with a given radius. The cluster only contains the locations that have enough visit counts, for example, over a given threshold. Then, it treats the cluster as a polygon, whose apexes are the most distanced locations to the cluster centre. Finally, the centroid of the polygon with LapDP noise is used as the replacement for the original location for subsequent publishing. Note that, in most cases, the centroid of the polygon is not the same as the centre of the cluster, that is, the original location because the given visit counts filter out some locations that are close to the real location.

Ma et al. [93] proposed an R-tree-based continuous location-publishing scheme, AGENT, based on dynamic programming. AGENT produces and saves fake locations that are geometrically close to the real locations, in an R-tree. It then searches the corresponding fake locations in the R-tree to publish the trajectory. In particular, several locations in a given area are mapped to a sanitised location, which is used as a replacement for those locations. Such a sanitised location is created by drawing a random location in polar coordinates, where the real location is the origin, and sampling a radius and angle with uniform distribution. When publishing the real location, AGENT searches its corresponding sanitised location in the R-tree within a bound given by LapDP, that is, the distance between the real location and the selected sanitised location is bounded by random LapDP noise.

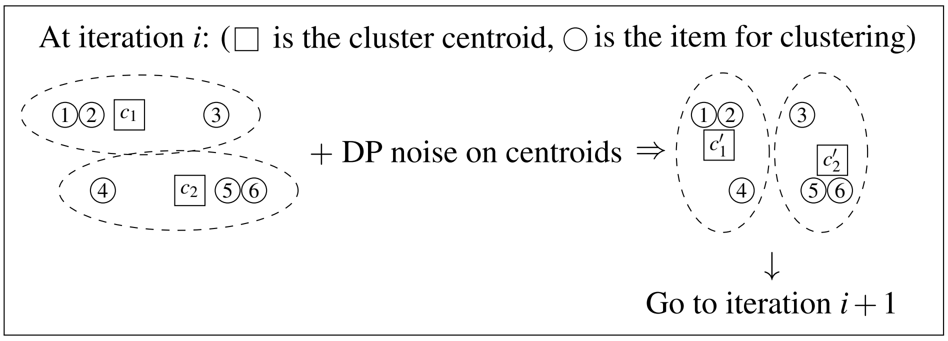
Xiao et al. [171, 173] observed that the spatiotemporal relationship may leak privacy, even if a perturbed trajectory is published. For instance, if a location A is sampled as the perturbation of the real location X, note that A should be close to X for utility; however, in the area of A, if X is the only possible place for a person to visit, then the adversary would immediately know that X is the real location. To address such privacy disclosure problems caused by spatiotemporal relationships, Xiao et al. [171, 173] prepared a δ-location$_t$ set for each location at timestamp t in the trajectory. All locations appearing in the δ-location$_t$ set can possibly be visited at timestamp t, which satisfies the requirement for the spatiotemporal relationship. In case the real location is not in the δ-location$_t$, a nearest location will be used instead. Then, the most possible location to replace the real one in the original trajectory will be sampled via ExpDP from the δ-location$_t$ set. Additionally, a planar isotropic mechanism (PIM) was proposed to achieve an optimal lower bound of DP for the δ-location$_t$ set. Following [171, 173], Cao et al. [20] proposed a private spatiotemporal event (PriSTE) framework to further study the spatiotemporal privacy of [171]. The researchers formally defined a spatiotemporal event as Boolean expressions of a set of (location, time) predicates, which is a generalisation of geometric-based trajectory.

In summary, by considering the features of a trajectory, such as the movement directions, spatiotemporal relationships and closeness of locations, the state-of-the-art DP single-trajectory publishing algorithms achieve satisfying trade-offs between the privacy guarantee and data utility of the published trajectories. However, we observe two weaknesses with existing work. First, through applying DP, even trusted users, who should have full access to the trajectory information, cannot recover the real trajectory. Second, because of assumptions about the adversary's background knowledge, these works are vulnerable when a given adversary's background knowledge does not fit their assumptions. Specifically, we assume that the adversary has full knowledge about the vertices in a path, but misses some edges (in the worst case, only one missing edge). With such assumptions about the adversary, the published trajectory from existing work would be vulnerable to releasing the missing edge to the adversary. Therefore, in this thesis, we aim to propose an approach that publishes a trajectory with full utility to trusted users, while preserving privacy against adversaries under our assumption.

3.4 Background to Differentially Private *k*-Means Clustering

In this section, we briefly summarise the work on differentially private *k*-means clustering in interactive settings [12, 34, 104, 176, 148, 118] and non-interactive settings [112]. Figure 3.4 displays the general idea of centroids updating at any iteration of the interactive differentially private *k*-means clustering for two clusters, where triangles and circles denote the centroid of a cluster and the items for clustering, correspondingly.

Lloyd's *k*-means algorithm [87] is the base algorithm for iteratively updating centroids. Differentially private *k*-means algorithms typically inject/embed DP into Lloyd's algorithm in some way according to their needs/demands and assumptions. Specifically, DP *k*-means



At iteration i: (□ is the cluster centroid, ○ is the item for clustering)
① ② c_1 ③
④ c_2 ⑤ ⑥
+ DP noise on centroids ⇒
① ② c'_1 ④
③ c'_2 ⑤ ⑥
Go to iteration $i+1$

Fig. 3.4 General Idea of Interactive Differentially Private k-Means Clustering.

clustering algorithms in both interactive and non-interactive settings assume that the adversary has maximum background knowledge about the dataset for clustering, that is, $n-1$ out of n items in the dataset. Existing work in interactive settings further assumes that the clustering result of any iteration may be disclosed to an adversary. Therefore, to preserve DP, the works in interactive settings inject DP noise to each iteration using three major DP mechanisms: the Laplace mechanism (LapDP) [37], the sample and aggregation framework [114] and the exponential mechanism (ExpDP) [100]. Meanwhile, the works in non-interactive settings mainly inject DP noise into the input and/or output using LapDP.

In interactive settings, there is a group of works [12, 34, 148] focusing on injecting Laplace noise to the centroids of each iteration of Lloyd's algorithm directly to ensure DP. The difference between these works is the way they allocate the privacy budget to each iteration. Blum et al. [12] proposed one of the first differentially private k-means algorithms by applying LapDP in a sublinear query (SuLQ) database. The key idea of [12] is adding LapDP noise with the same privacy budget to the k centroids at each iteration of Lloyd's algorithm. This algorithm preserves the privacy of each record in the input database using DP, even against adversaries with the maximum background knowledge. However, it is not easy to apply SuLQ in real-world databases because it would not terminate without a predefined number of iterations. To guarantee an acceptable utility of the clustered results, it is necessary to find a suitable value for the number of iterations by running experiments repeatedly, rather than analysing the algorithm theoretically to determine the number of iterations. When the size of the input database increases, the computational cost to find such a predefined iteration number also increases significantly.

To avoid running experiments for a predefined number of iterations, Dwork [34] proposed a privacy budget self-consuming approach to assign the privacy budget of each iteration. In particular, they set a total privacy budget for *k*-means clustering according to the privacy requirements, and then allocated a privacy budget to each iteration with decreasing exponential distributions, such as $\varepsilon/2$ for the first iteration, $\varepsilon/4$ for the second, and $\varepsilon/8$ for the third, etc. This way, the algorithm can terminate when the privacy budget is exhausted. However, the total noise injection cannot be ignored. A larger privacy budget implies less noise injection and hence better utility. The privacy budget allocation of [34] demonstrates that every later iteration will add more noise to the k centroids before we finally lose control of clustering utility because the final step will almost add ($\varepsilon = 0$)-DP noise, which means no further data utility. Additionally, if we want to control the amount of injected noise in [34], we must have a predefined number of iterations, which would be the same problem as faced by [12].

To improve the results of [12] and [34], Su et al. [148] proposed an "optimal" privacy budget allocation method, which was proven to assign the best privacy budget value to each iteration. The key idea of the optimised privacy budget allocation is two-fold. First, it calculates the mean square error (MSE) to measure the amount of injected noise to each iteration. Second, it chooses the number of iterations to ensure that the MSE of each iteration is bounded by a given threshold. Su et al. [148] guaranteed that the algorithm terminated within a finite number of iterations and had a bounded (or optimal) overall noise injection. However, in the proof of Equation (4) in [148], a key assumption was that the number of items in each cluster in Lloyd's algorithm was the same. This is not a realistic assumption in real-world databases. That is, the proposed "optimal" privacy budget allocation only works with perfectly clustered datasets.

To date, there is only one work that applies a sample and aggregation framework to ensure DP for an interactive *k*-means clustering algorithm. Mohan et al. [104] proposed GUPT (that means 'secret' in Sanskrit), which embeds the sample and aggregation framework of DP into Lloyd's algorithm. GUPT uniformly samples items from an input dataset to different buckets, where the local clustering results of each bucket are produced by Lloyd's algorithm. The final clustering result is the mean of local results with Laplace noise injected. GUPT guarantees both DP for the final result and algorithm convergence for *k*-means clustering. The clustering result of any iteration does not release the privacy of any arbitrary item, as uniform sampling ensures that the adversary does not know which bucket contains the missing item. However, the clustering quality is unsatisfactory because of the large amount of noise involved in the sample and aggregation steps, since the uniform sampling may result in skewness over the sampled buckets, that is, it can place items belonging to one cluster (the final cluster when the algorithm terminates) into the same bucket. Then, when we aggregate (calculate the average value of) the clustering result in each bucket, the aggregated centroids would have a significant distance from the exact Lloyd's result.

ExpDP was also used to ensure DP for *k*-means clustering in interactive settings. Zhang et al. [176] proposed a genetic algorithm (GA)–based differentially private *k*-means clustering algorithm, PrivGene. Unlike with traditional GAs, which select the most qualified candidates, PrivGene randomly samples the candidates with ExpDP for each iteration, and then applies the crossover and mutate operations of traditional GAs to iteratively update the centroids. PrivGene achieves fair clustering quality if the input dataset is relatively small because only then can it produce globally optimal clustering results with high probability. The reason for achieving the global optimum is that the mutate operation is computed on multiple selected centroids; hence, it is possible to find a global optimum with a small dataset. However, similar to [12], PrivGene also requires a predefined iteration number to terminate the algorithm, so has efficiency issues. Differing from the above algorithms, Park et al. [118] focused on

(ε, δ)-DP, rather than ε-DP, with given assumptions on the distribution of the input dataset, which narrows its applicability to real-world scenarios.

The above are some of the major methods that interactively modify Lloyd's algorithm to be differentially private. We also consider other methods to achieve differentially private *k*-means clustering based on non-interactive approaches by adding DP noise to the input/output data, rather than perturbing the algorithm itself. Nguyen et al. [112] proposed a framework (similar to ExpDP) to add noise to the final clustering results directly, based on the adversary's background knowledge (e.g., whether a single record belongs to a given cluster). In the final iteration, instead of calculating the mean of a cluster, Nguyen et al. calculated an estimated mean for each cluster. Their approach first creates a bag for each cluster, and then bootstrap samples (i.e., sampling with replacement) the items from one cluster to its corresponding bag, where the final number of items of a bag comprises the cluster. It then calculates the means for each bag as an estimation for the original cluster. In fact, any approach that can process raw data to be differentially private could be used for non-interactive *k*-means clustering. For example, Qardaji et al. [125] proposed differentially private grids for geospatial data to answer queries about two-dimensional data with high accuracy. Su et al. [147] improved the result in [125] by making the input data *k*-means differentially private. The improvement of [147] was to find a way to decide a key parameter in [125] for high-dimensional data. Adding noise to the data (either input or output) would be a straightforward way to achieve DP; however, once we assume that the clustering result of any iteration may be disclosed to an adversary, just injecting noise to the output would not be a safe choice. Further, when the size of the input database is very large, injecting noise to the input can result in an additional high computational cost on the data pre-processing prior to the clustering process.

Based on the above analysis of existing differentially private *k*-means clustering algorithms in both the interactive and non-interactive settings, we draw three observations for differentially private *k*-means algorithms. First, we should preserve privacy for all iterations in case of the accidental disclosure of clustering results of any iteration to preserve DP in interactive settings. Second, convergence is an important property of differentially private k-means algorithms for both efficiency (convergence avoids running algorithms repeatedly to determine a predefined number of iterations) and the resultant clustering quality (convergent algorithms terminate at one of the local optimums for k-means problems). Third, it is essential to have a good trade-off between the privacy of each single item in a dataset and the clustering quality. In this thesis, we explore how to guarantee convergence and improved clustering quality to meet the same DP requirement as existing work in interactive settings.

3.5 Summary

In this chapter, we have briefly reviewed the literature in state-of-the-art differentially private data analytics from the following aspects: differentially private data aggregation with security guarantees, privacy budget guarantees suitable for distributed systems, differentially private single-trajectory publishing and differentially private k-means clustering. We observe some problems in existing work, as follows:

- For differentially private data aggregation, the weakness of existing work is when data aggregator(s) maliciously collude to present a risk to data curator(s).

- For the privacy budget of distributed systems and the parallel composition of DP budget, existing lower bounds of privacy budget are insufficient for the summation/counting function.

- For differentially private single-path publishing, the state-of-the-art approaches fail to protect the existence of edge(s) when the adversary knows all information (vertices and edges) except one edge in a path.

- For differentially private k-means, poor clustering quality arises because of potential non-convergence.

We provide solutions and technical details to address these problems in Chapters 4 to Chapter 7, respectively.

Chapter 4

Differentially Private Data Aggregation with Security Guarantees

With the large deployment of smart devices, the collection and analysis of personal data can significantly benefit both industry and people's daily life. However, it gives rise to a serious risk to people's privacy. Recently, combining SMC and DP has offered a popular strategy to guarantee both (computing) security and (output) privacy when dealing with distributed data aggregation. To decrease the communication costs of the traditional SMC paradigm, the existing work introduces trusted/semi-honest data aggregator(s) that aggregate data collected from distributed semi-honest data curators. However, there is neither security nor privacy guarantees when trusted/semi-honest data aggregators collude. That is, collusion of data aggregators could result in exchange of received data that could be used to infer the privacy of contributing data curator(s). In the worst case, $n-1$ out of n data aggregators may collude. To address the privacy disclosure problem caused by the colluding and semi-honest data aggregators, while maintaining acceptable data utility, we propose an approach that randomly injects DP noise into both data curators and data aggregators. Theoretical and experimental analyses show that, to achieve the same security and privacy guarantees, our framework provides better data utility than existing approaches.

4.1 Introduction

The application and development of the IoT, radio frequency identification [72] and wireless sensor networks [97] have significantly affected the way we live [52, 14, 180]. For example, modern recommender systems provide us with accurate recommendations by collecting and analysing our online behaviour, such as shopping histories, visited locations and identifying users that have similar preferences to us [101].

In many real-life applications of the IoT, the source data are distributed across different parties or servers. To learn important insights from the distributed data, it is necessary to support computations over the distributed sources of data. However, such joint computation raises key challenges in (computing) security and resultant (output) privacy [122]. Thus, the security requirement aims to guarantee that every computational party only learns the knowledge that can be learnt from the computation outputs [57], while the privacy requirement aims to decrease the probability of inferring other parties' privacy based on the computation outputs.

To ensure security, techniques to achieve SMC (or MPC) (see details of SMC in Chapter 2) are the most established. As discussed, the SMC paradigm allows multiple parties to jointly compute a function over each party's private data, without a trusted third party taking the raw data from each party and computing the result. Consequently, each party is oblivious to the other parties' private inputs. After many years of development, there are a number of established techniques to implement SMC, including secret sharing, oblivious transfer, zero-knowledge proof and homomorphic encryption [24]. However, SMC remains vulnerable to inference attacks, as discussed in Chapter 2. Given that the final output of SMC makes no difference to the corresponding non-privacy-preserving computation, adversaries can still infer people's privacy based on the explicit output and auxiliary information.

To achieve privacy, researchers apply some famous data perturbation techniques, such as data randomisation [33, 160] and data anonymisation [150, 179] (that includes data suppression and data generalisation). These perturb the original data and/or algorithm to satisfy privacy requirements (a detailed introduction of these techniques can be found in Chapter 2). Among all the techniques for data randomisation and anonymisation, DP [33] is one of the most promising because of its strong assumptions about adversaries' background knowledge and the privacy budget, ε, that can trade-off data utility and privacy. However, DP does not provide any security guarantees in distributed environments.

Given that neither SMC nor DP can address the above privacy challenges independently, a common approach is combining SMC and DP to leverage the advantages of the two techniques. Researchers have applied a combined SMC and DP scheme in privacy-preserving data aggregation [130, 1, 140, 38, 39, 52, 122]. In [39, 122], the schemes worked on a full connected network; however, this resulted in a high communication cost. Meanwhile, in [130, 1, 140, 52, 39, 122], the schemes assumed that at least some of the computation parties (either data curators or data aggregators) were semi-honest, and could potentially disclose a specific user's (party's) privacy when there is collusion with $n-1$ parties (malicious adversaries). The research in [38] addressed the problems of colluding data curators and data aggregators and high communication cost by introducing a semi-honest server. Such a semi-honest server aggregated the final output. However, if the semi-honest server is compromised, user privacy will be at risk. To conclude, existing work suffers from at least one of the following problems: (1) potential privacy disclosure risk against malicious data curators (or aggregators) through collusion attacks or a malicious central server or (2) high communication costs due to the fully connected network for information sharing without a central server. To address these weaknesses, the contributions of this chapter are as follows:

- In contrast to existing work, which guarantees security and privacy against semi-honest computation parties or with a mix of semi-honest and colluding semi-honest (malicious) parties, we assume that all parties are potentially colluding semi-honest (malicious), while still achieving the same security and privacy guarantees.

- We propose a two-layer (data curator and data aggregator layers) data aggregation approach, where each party injects DP noise randomly, instead of injecting noise by all parties, as is common in existing work. Additionally, we demonstrate that our random noise-injection approach achieves the same DP requirements as existing work.
- We experimentally evaluate our approach against the state-of-the-art approaches using a synthetic dataset and summation function. To achieve the same security and privacy guarantees, our approach offers enhanced data aggregation utility because of the random noise injection.

Organisation. The remainder of this chapter is organised as follows. In Section 4.2, we provide a brief introduction to the preliminaries of this chapter, and then propose the novel privacy-preserving data aggregation scheme, which addresses the privacy disclosure problem of existing work. In Section 4.3, we provide an experimental evaluation of existing work and discuss our framework. Finally, in Section 4.4, we conclude this chapter.

4.2 Privacy-Preserving Data Aggregation with Colluding Semi-honest Aggregators

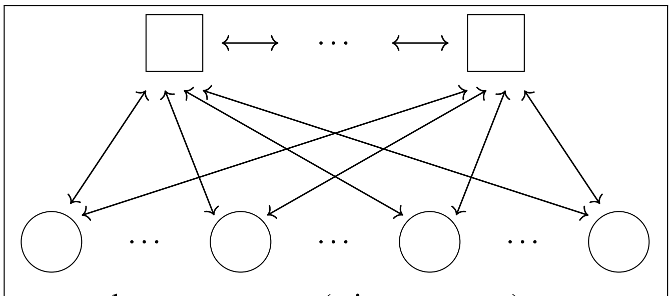
In this section, we introduce our privacy-preserving data aggregation framework and present a theoretical evaluation of its performance with regard to security, privacy and utility.

4.2.1 Adversary Model

Generally, we categorise adversaries into two models, semi-honest adversaries and malicious adversaries, according to their willingness to follow a privacy-preserving protocol. A semi-honest adversary is also known as a *honest-but-curious* (HBC) or *passive* adversary. That is, a semi-honest adversary always follows the protocol faithfully, yet tries to infer additional information (especially private information) from both the process and output of a protocol. A malicious adversary is also known as an *active* adversary who aims to cheat the protocol arbitrarily to violate the targeted victim's privacy. In this chapter, we define a malicious adversary as a colluding, semi-honest adversary.

4.2.2 Our Framework

This work is inspired by [38]. Given that the network model in [38] is not a fully connected graph, the communication cost will be sufficiently low: $\mathcal{O}(n)$, where n is the number of data curators. Moreover, the security is guaranteed through secret sharing between the data curators and aggregators from the architecture in [38]. Building on the two advantages of [38], our framework also comprises two layers, as shown in Figure 4.1. Note that, without a data aggregator, a ring network can potentially achieve low communication cost with both security and DP guarantee. That is, each data curator sends the aggregation of data received from a predecessor and his or her differentially private data to the successor. The final data curator would have the final aggregation result in $\mathcal{O}(n)$ time complexity. However, such a scheme would fail to deliver the final output if there is a faulty connection.



□ : c data aggregators (c is a constant)
○ : n data curators (at most $n-1$ are colluding)
$\leftrightarrow$: secure communication channels

Fig. 4.1 Framework Overview of Differentially Private Data Aggregation with Colluding Semi-honest Aggregators.

Specifically, in Figure 4.1, there are n data curators with private data as inputs to a linear data aggregation function, such as counting or summation. We assume that there are at most $n-1$ malicious (colluding semi-honest) data curators who work together to attempt to violate a victim's privacy. There are c malicious (colluding semi-honest) data aggregators who assist in this process. It is quite straightforward to make this assumption because curators' privacy/secret will be easily disclosed by cooperation of k out of c aggregators, where k is the security bound of the secret-sharing scheme. Note that, in the real world, compared with the number of customers (data curators, such as mobile telephone users), the number of servers (data aggregators) is much smaller; hence, we assume that the number of servers is constant in this chapter. In this model, we cannot send the original data to any aggregators directly. Instead, we must upload the differentially private data to the aggregators to ensure the privacy of the source curator data. However, a new problem will emerge if we only inject noise at the curator layer. Given that we care about the data utility, we must limit the extent that we perturb the original data; however, cooperation of adversaries can also potentially violate the victim's privacy from the final output. Thus, we must also inject differentially private noise at the aggregator layer. However, since we inject noise at both the curator layer and aggregator layer, the overall data utility will be significantly affected because of the extra noise. Therefore, in this chapter, we introduce two probabilities at the two layers for injecting the noise. The data utility will be controlled by tuning these two probabilities. Such noise injection probabilities can be treated as an additional privacy parameter/measurement to normal differential privacy, which are similar to the idea of privacy budget, therefore it is assumed known beforehand. Specifically, a higher noise injection probability indicates higher privacy guarantee. Since determining a suitable value of this noise injection probability is similar to that of privacy budget, it is out of the scope of this thesis. Algorithm 4.1 shows how the framework works.

Algorithm 4.1: Privacy Preserving Data Aggregation.

```
Input  : Original curator data set: $\mathscr{U} = \{u_1, u_2, \ldots, u_n\}$;
         Data aggregation request function (linear function): $f(\mathscr{U})$;
         Number of curators: $n$; Number of aggregators: $c$;
         Data splitting weights: $\omega_i^j$ for data curator $i$'s $j$th piece of data;
         DP budget: $\varepsilon_u^i$, for each curator $i$; $\varepsilon_s^j$, for each aggregator $j$;
         Noise adding probability: $\Pr_u^i$, for each curator $i$; $\Pr_s^j$, for each aggregator $j$.
Output : Privacy preserved data aggregation result: $\mathscr{R}$.
1 Curator $i$ splits $u_i$ into $c$ parts: $u_i = \sum_{j=1}^{c} \omega_i^j u_i$, where $\sum_{j=1}^{c} \omega_i^j = 1$;
2 Curator $i$ generates a DP noise $Nu_i$ with $\varepsilon_u^i$ and local sensitivity $\Delta sum(\omega_i^j u_i)$, splits
    $Nu_i = \sum_{j=1}^{c} \omega_i^j Nu_i$; $u_i^{j'} = u_i^j + \omega_i^j Nu_i$ with probability $\Pr_u^i$, $u_i^{j'} = u_i^j$ with
    probability $1 - \Pr_u^i$;
3 Curator $i$ sends $u_i^{j'}$ to Aggregator $j$;
4 Aggregator $j$ receives a set $s_j = \{u_1^{j'}, u_2^{j'}, \ldots, u_n^{j'}\}$ from each curator $i$;
5 Aggregator $j$ generates a DP noise $Ns_j$ with $\varepsilon_s^j$ and local sensitivity $\Delta f(s_j)$, then
    calculates $f(s_j)' = f(s_j) + Ns_j$ with probability $\Pr_s^j$, $f(s_j)' = f(s_j)$ with
    probability $1 - \Pr_s^j$;
6 All aggregator $j$ construct a Secured Multi-party Computation for $\mathscr{R} = f(f(s_j)')$;
```

From Line 1 to Line 3, the data curators apply DP and a secret sharing scheme to guarantee both the security and privacy of their private data. A noise injection probability, $\Pr_u^i$, is used for a curator i to decide whether a noise will be added. From Line 4 to Line 6, the data aggregators apply DP and the basic SMC paradigm, so that none of the data at each aggregator will be released to other aggregators. Similar to what happened at the curator layer, each aggregator j will also use a noise injection probability, $\Pr_s^j$, to inject DP noise randomly.

4.2.3 Theoretical Evaluation

Privacy and Security Guarantee

Proposition 4.1. *The proposed privacy preserving data aggregation framework is ε-differentially private.*

Proof. Given that our framework works with two layers, we start from the data curator layer:

Each data curator i injects differential privacy noise with probability $\Pr_u^i$ (injects 0 with probability $1-\Pr_u^i$); thus we have the following expectation of the differentially private data after noise injection.

$$\hat{u}_i = u_i + \Pr_u^i \times Lap^{-1}(\max_{i=1}^n\{\omega_i^j u_i\}, \varepsilon_u^i) + (1-\Pr_u^i) \times 0,$$

where $\max_{i=1}^n\{\omega_i^j u_i\}$ is the local sensitivity of the aggregation function (summation/counting in this chapter) for u_i. Let it be $\Delta f_L(u_i)$, and then $Lap^{-1}(\Delta f_L(u_i), \varepsilon_u^i)$ is the LapDP noise generated by Laplace distribution. Thus for u_i, we have

$$\Pr[\hat{u}_i = x_i] = \Pr[noise = x_i - u_i] = \Pr_u^i \times Lap(\max_{i=1}^n\{\omega_i^j u_i\}, \varepsilon_u^i, noise),$$

where $Lap(\Delta f, \varepsilon, y)$ returns the probability of $noise = y$, $\Pr[noise = y]$. We then bound DP by

$$\frac{\Pr[\hat{u}_i = x_i]}{\Pr[\hat{u}_i' = x_i]} = \frac{\Pr[noise = x_i - \hat{u}_i]}{\Pr[noise = x_i - \hat{u}_i']} = \frac{\Pr_u^i \times Lap((\max_{i=1}^n\{\omega_i^j u_i\}), \varepsilon_u^i, (x_i - \hat{u}_i))}{\Pr_u^i \times Lap((\max_{i=1}^n\{\omega_i^j u_i\}), \varepsilon_u^i, (x_i - \hat{u}_i'))}.$$

According to the definition of DP (Definition 2.5, Chapter 2), assuming the aggregation function is summation, the above is upper bounded by $\exp(\varepsilon_u^i)$. That is,

$$\Pr[\hat{u}_i = x_i] < \Pr[\hat{u}_i' = x_i] \times \exp(\varepsilon_u^i).$$

Given that, at the curator layer, we have a parallel composition, according to Theorem 2.2 (Chapter 2), we have guarantee $(\max_{i=1}^n\{\varepsilon_u^i\})$-DP at this layer. Similarly, we will have a $(\max_{j=1}^c\{\varepsilon_s^j\})$-DP guarantee at the aggregator layer, given that the function f in this chapter is a linear function, it is easy to maintain the same property with the curator layer.

Overall, because our framework works with two layers sequentially, based on Theorem 2.1 (Chapter 2), our framework is $(\max_{i=1}^{n}\{\varepsilon_u^i\} + \max_{j=1}^{c}\{\varepsilon_s^j\})$-differentially private. □

Clearly, Proposition 4.1 ensures that the noise injection probabilities at both the curator and aggregator layer do not affect the privacy performance of our framework. Therefore, we can safely tune the two probabilities to achieve enhanced data utility, whilst retaining the same privacy guarantees as the existing privacy-preserving data aggregation schemes.

Proposition 4.2 ([38]). *The proposed privacy preserving data aggregation framework is secure.*

Utility Analysis

In this section, we use the variance of Laplace distribution to measure the performance of computation utility between our framework with PrivaDA that uses a Laplacian mechanism [38]. It is clear that the worst case for noise injection is dependent on the variance of the Laplace distribution. We have Inequality 4.1 to represent a relationship to achieve better data utility (than PrivaDA, because PrivaDA [38] proves it has better data utility than PASTE [130]), while guaranteeing the same DP. Specifically, on the left side of Inequality 4.1, we calculate the overall Laplace parameter involved in Algorithm 4.1, where the first item represents the n data curators (users) and the second item represents the c data aggregators (servers). On the right side of Inequality 4.1, we calculate the overall Laplace parameter for PrivaDA. To keep the overall privacy budget the same as Algorithm 4.1, that is, the same privacy guarantee, each data aggregator j in PrivaDA uses $\varepsilon_s^j + \max_{i=1}^{n}\{\varepsilon_u^i\}$ as the privacy budget by parallel and sequential composition of privacy budget (Theorems 2.1 and 2.2 in Chapter 2).

$$\sum_{i=1}^{n}(\Pr_u^i \frac{\Delta f}{\varepsilon_u^i}) + \sum_{j=1}^{c}(\Pr_s^j \frac{\Delta f}{\varepsilon_s^j}) \leq \sum_{j=1}^{c} \frac{\Delta f}{\varepsilon_s^j + \max_{i=1}^{n}\{\varepsilon_u^i\}}. \tag{4.1}$$

That is,

$$\frac{n}{c}\min_{i=1}^{n}\{\Pr_u^i\}(1+\frac{\max_{j=1}^{c}\{\varepsilon_s^j\}}{\max_{i=1}^{n}\{\varepsilon_u^i\}})+\min_{j=1}^{c}\{\Pr_s^j\}(1+\frac{\max_{i=1}^{n}\{\varepsilon_u^i\}}{\max_{j=1}^{c}\{\varepsilon_s^j\}})\leq 1,$$

Given that we assume that the maximum values of the privacy budget for both data curators (users) and data aggregators (servers) should be non-zero, let $ratio = \frac{\max_{i=1}^{n}\{\varepsilon_u^i\}}{\max_{i=1}^{n}\{\varepsilon_u^i\}+\max_{j=1}^{c}\{\varepsilon_s^j\}} \in (0,1)$, and we have:

$$\frac{n}{c}\times\frac{\min_{i=1}^{n}\{\Pr_u^i\}}{ratio}+\frac{\min_{j=1}^{c}\{\Pr_s^j\}}{1-ratio}\leq 1. \tag{4.2}$$

In Section 4.3, we use Inequality 4.2 to determine the value of the privacy budget *ratio* to achieve enhanced data utility in the experiments.

Further, if we take the right-hand side of Inequality (4.2) as a normalised noise injection of existing work (normalised to 1), then the left side of Inequality (4.2) measures the normalised noise injection of proposed Algorithm 4.1. Let it be a function of *ratio*, that is, $f(ratio) = \frac{n}{c}\times\frac{\min_{i=1}^{n}\{\Pr_u^i\}}{ratio}+\frac{\min_{j=1}^{c}\{\Pr_s^j\}}{1-ratio}$, and then it is easy to find (by solving $f'(ratio) = 0$) that $\exists raio \in (0,1)$, $f(ratio)$ has its global minimum value, i.e., minimum noise injection.

4.3 Experimental Evaluation

4.3.1 Evaluation Metric

In this chapter, we use the mean absolute error (MAE) [166] and the mean relative error (MRE) [145], to measure the data utility in the experiments:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|R_i - R|,\ \ MRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|R_i - R|}{R} \tag{4.3}$$

where N is the overall number of experiments, R_i is the output of the ith experiment on a privacy-preserving data aggregation scheme, and R is the ground truth, which is the output of the original data aggregation function. Specifically, in our experiments, we set $N = 10,000$. Clearly, a lower MAE (MRE) indicates better performance for data utility.

4.3.2 Experimental Results

Experimental settings. According to the analyses in Section 3.1, for our experiments, we compare the performance of data utility with schemes that share a similar topology for data aggregation: PrivaDA [38], PASTE [130], a naive idea (Orig, which injects differentially private noise at both the curator layer and aggregator layer without randomness, that is, with probability equal to 1), and the proposed idea in this chapter (ThisWork, which injects noise with randomness, as in Algorithm 4.1). In brief, PrivaDA has a three-layer network, where the data curator layer splits and sends local data to data aggregators, and the data aggregator layer aggregates the received data pieces, injects DP noise and sends the aggregated data to a central server. PASTE implements an n to 1 network, where n data curators send differentially private encrypted data to a data aggregator. Details about PrivDA and PASTE can be found in Section 3.1. To simplify the experiments, we assign the same privacy budget (in all of the above schemes) and inject noise with probability for all participants in the same layer for ThisWork. To achieve the same DP guarantee, we set the privacy budget as $\varepsilon_u + \varepsilon_s = \varepsilon$, where ε is the privacy budget for PrivaDA and PASTE. In particular, we have $\varepsilon_u = ratio \times \varepsilon$. The number of data curators $n = 10^4$ and the number of aggregators $c = 5$.

In the experiments, the values of $ratio$ approximate the theoretical optimal one (Inequality (4.2)), yet do not exactly match it because of the randomised mechanism involved in the experiments. Further, we do not use the real-world dataset because our objective is to evaluate the performance of the above schemes based on the numeric properties of the input data. Therefore, we generate synthetic data for each data curator with the same normal distribution (mean = 500, variance = 1) where we have ten thousands data curators, each data curator has five numeric data records. We implement the normal distribution with a

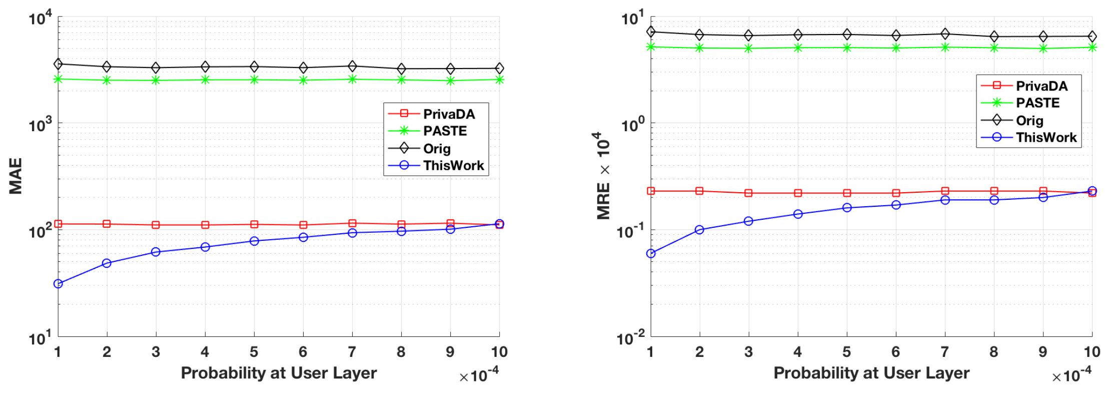


Fig. 4.2 Data Utility Affected by Probability at Curator Layer ($\varepsilon = 10$, $\Pr_s = 10^{-2}$).

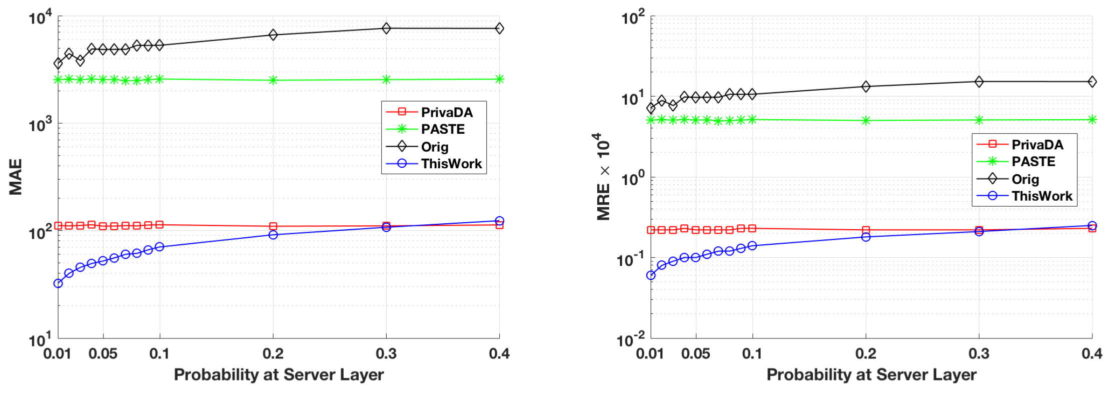


Fig. 4.3 Data Utility Affected by Probability at Aggregator Layer ($\varepsilon = 10$, $\Pr_u = 10^{-4}$).

default MATLAB function, `normrnd()`. For each single data record for data curators, we truncate the generated data to a positive float array, where each single array element has four decimal places. In addition, to comprehensively explore the data utility when applying DP on summation function, we set the values of privacy budget to a large range $[1, 100]$. (Source code: https://github.com/suluz/sec_dp_aggregation)

Experimental results analysis. Figures 4.2 and 4.3 display how the noise injection probability at both the curator layer and the aggregator layer affects the data utility. As expected, the noise injection probabilities has a negative effect on data utility because both the MAE and MRE values increase when we increase the noise injection probability. It is clear that, by tuning these two probabilities, we can control the overall output quality to achieve enhanced data utility over existing schemes. Figure 4.4 illustrates how the privacy budget ε

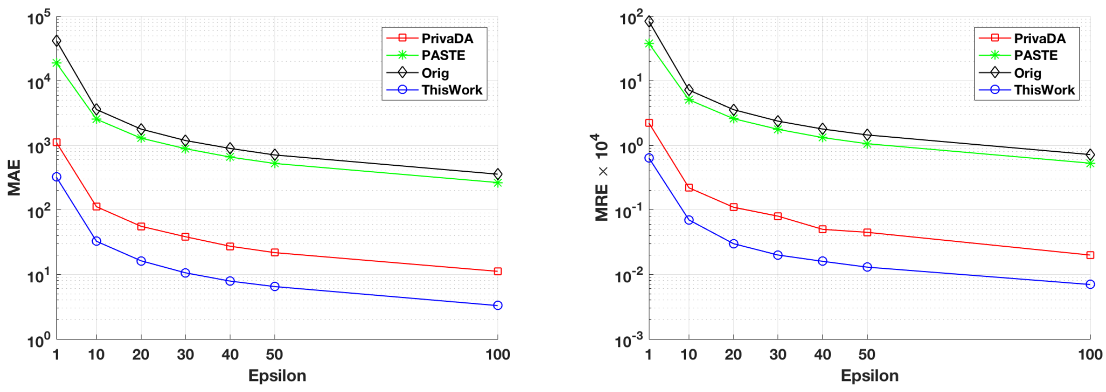


Fig. 4.4 Data Utility Affected by ε ($ratio = 0.75$, $\Pr_u = 10^{-4}$, $\Pr_s = 10^{-2}$).

in DP affects the data utility. As an important property of DP, a larger ε signifies that less noise is added to the original request output; hence, it achieves better data utility. Figure 4.5 demonstrates how the privacy budget ratio between the curator and aggregator layer affects the data utility. We see that, by tuning the privacy budget ratio, the schemes, which introduce privacy budget into both the curator and aggregator layer, can achieve optimal performance on data utility. In addition, we note that the MAE and MRE of ThisWork and Orig increase in the tail in Figure 4.5. Such behaviours match our theoretical analysis for the utility of Algorithm 4.1, that is, there exists a privacy budget $ratio \in (0,1)$ that provides the minimum noise injection (i.e., the minimum value of *MAE* and *MRE* in Figure 4.5). Further, because PrivaDA and PASTE do not inject DP noise for both data curators and data aggregators, their *MAE* and *MRE* values are stable when tuning the ratio.

4.4 Summary

The applications of the IoT make it possible for large-scale data collection and analysis through data aggregation. However, preserving personal privacy while ensuring acceptable data utility is a significant challenge for data aggregation applications. In this chapter, we have overcome the privacy disclosure problem of existing work caused by semi-honest data aggregators, and provided a novel privacy-preserving data aggregation scheme that

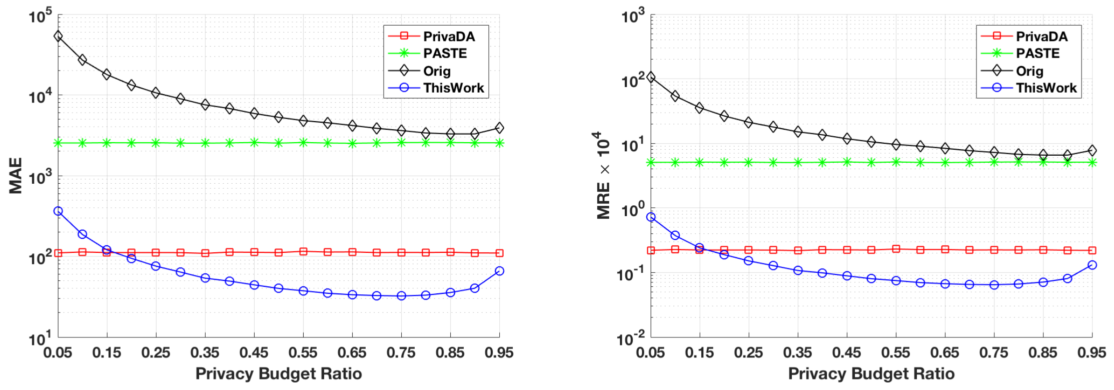


Fig. 4.5 Data Utility Affected by Privacy Budget Ratio ($\varepsilon = 10$, $\Pr_u = 10^{-4}$, $\Pr_s = 10^{-2}$).

operates with colluding semi-honest data aggregators. Both the theoretical and experimental evaluations demonstrated that, to achieve the same guarantee of security and privacy as in previous works, our scheme provides enhanced output data utility.

Chapter 5

Privacy Budget Guarantees of Distributed Systems

Following the previous chapter, when aggregating data from distributed data curators, to preserve the privacy of data from each data curator, we could apply DP on data curators' original data prior to sending them to a data aggregator. However, such a privacy-preserving process could be vulnerable to worst-case collusion attacks, where $n-1$ out of n colluding curators work together to infer a victim's privacy from the aggregation output. That is, the privacy performance of the whole aggregation system may not be differentially private with regard to the victim data curator. Therefore, it is crucial to study the privacy budget guarantees of distributed systems. Thus, the function to map the local DP budget of each curator (that reflects the privacy level of each curator's local data) to a global DP budget (that reflects the privacy level of the whole aggregation system) is important.

In this chapter, we study the parallel composition of privacy budgets of differentially private data aggregations through a special function, summation (counting), as this function is useful to learn statistics of the aggregated data; hence it is commonly used in real-life applications. Examples include analysing electricity consumption by aggregating data from smart electricity meters, and counting the overall number of HIV-positive patients by collecting data from each hospital. Currently, there are two major results of parallel composition of privacy budget: $\varepsilon = \max_i\{\varepsilon_i\}$ [99] and $\varepsilon = \sum_i \varepsilon_i$ [140]. For a summation function, the former [99] suffers from the problem of unknown privacy guarantees when the aggregation sensitivity of the whole system is less than the sum of the data curator's aggregation sensitivity. The latter [140] suffers from the problem of low data utility by using the global function sensitivity. To address these problems, we provide a new lower bound of privacy budget, $\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}$ (where $|\mathscr{U}|$ is the number of data curators and $\Delta f_L(x_i)$ is the local sensitivity of data curator i), which works with an unconditional sensitivity of the whole aggregation system. In particular, we take advantage of a property of the summation function when computing local sensitivity. That is, the local sensitivity of summation for a dataset equals the maximum summation of any sub-dataset. Additionally, we demonstrate that data updating does not affect our lower bound of parallel composition for summation. Both theoretical and experimental evaluations show that our privacy bound offers better global privacy performance than existing work, without any special conditions on the function sensitivity of the DP.

5.1 Introduction

As a result of the prevalence of smart devices, such as smart phones, fitness wristbands and wireless sensors, personal information is generated and collected at every moment of our lives [82, 124]. By analysing such information, we can improve the quality of our life, our productivity and the correctness of our decision making [126]. For instance, it is reported [105] that, based on the daily records of heart rate data from a smart wristband, a group of doctors took a series of appropriate actions to save a middle-aged male's life from a serious heart attack.

Among all the functions in data analysis, the summation and counting functions are two of the most classic and important ones. For example, energy providers can optimise electricity allocation in a state based on the periodic power usage (summation) of suburbs in the state, or a hospital can learn seasonal flu trends by counting the daily number of flu victims. In this chapter, we focus on the privacy issues associated with summation and counting functions.

Currently, to steal private information, instead of brute force attacks, which require powerful computing ability, inferring privacy through publicly available data is easier to adversaries [16, 47, 108, 143]. In this chapter, we examine the worst-case assumption that $n-1$ out of n data curators collude to infer a victim's privacy using published aggregation results. To preserve privacy for distributed summation and counting against worst-case adversary scenarios, we use DP as our privacy notion. The traditional DP usually works in centralised scenarios, based on a trusted and secure database that holds all the data. Given that we aim to preserve the data curator's privacy in a distributed setting against the worst-case collusion attacks (where $n-1$ out of n participants share their local data to infer a victim's private information by comparing their data with the final computing results), we need to extend the centralised DP to distributed DP (DDP).

In the research field of DDP, Dwork et al. [36] proposed the first scheme to extend DP in distributed scenarios by combining SMC and centralised DP. Inspired by [36], a body of work has applied the combination of SMC and DP in different applications, such as data aggregation [122], machine learning [18], data publishing [63] and theoretical computer science [115]. However, the techniques of SMC in these works incur a significant computational cost, which is not realistic for many smart devices. Differing from the above, McSherry [99] and Shi et al. [140] provided a more formal definition of DDP. Their DDP scheme automatically guarantees global ε-DP (the presence or absence of any data curator barely affects the aggregation outputs) by ensuring local ε_i-DP at each data curator i. That is, they study the DDP through studying the research problem of parallel composition of the privacy budget, where we aim to find a function g, such that $\varepsilon = g(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$. Given that such a function g does not depend on a specific network topology, for simplicity, we consider an n to 1 star network in this chapter Figure 5.1 displays the basic network topology settings and assumptions, where each data curator sends local aggregation x_i to a central server to have the final aggregation y, and f is the aggregation function, which is specified as summation in this chapter.

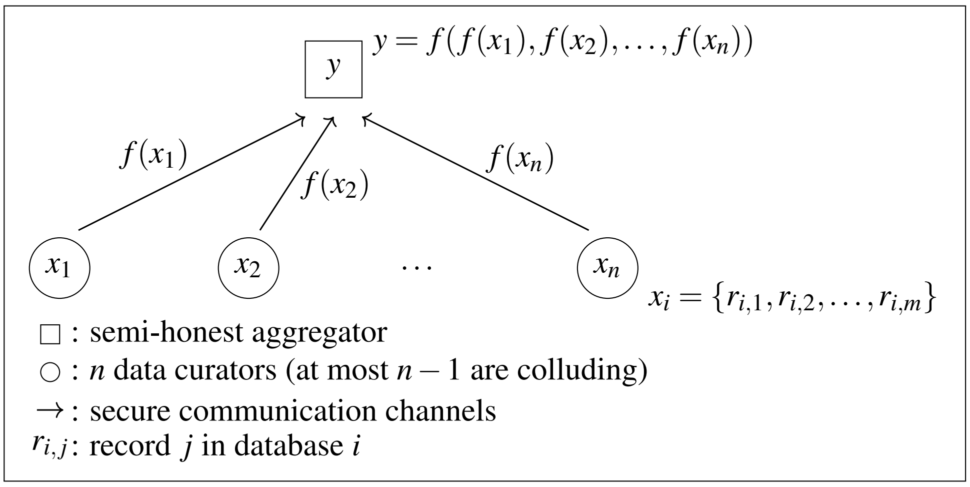

$y = f(f(x_1), f(x_2), \ldots, f(x_n))$
y
$f(x_1)$
$f(x_2)$
$f(x_n)$
x_1
x_2
$\cdots$
x_n
$x_i = \{r_{i,1}, r_{i,2}, \ldots, r_{i,m}\}$
□ : semi-honest aggregator
○ : n data curators (at most $n-1$ are colluding)
→ : secure communication channels
$r_{i,j}$: record j in database i

Fig. 5.1 Basic Network Topology Settings.

As noted, there are two major results: $\varepsilon = \max_i\{\varepsilon_i\}$ [99] and $\varepsilon = \sum_i \varepsilon_i$ [140][1]. However, we observe a weakness for each of the two solutions. The former [99] suffers from the problem of unknown privacy guarantees with conditional local function sensitivity, that is, the aggregator's local function sensitivity ($\Delta f_L(X) = \max_i\{f(x_i)\} = \max_i\{\sum_j x_{i,j}\}$, where $X = \{x_1, x_2, \cdots, x_n\}$ as defined in Equation (2.3), Chapter 2) should be greater than the sum of the data curator's local function sensitivity ($\sum_i \Delta f_L(x_i) = \sum_i\{\max_j x_{i,j}\}$). When we have a case where some of the data curators always return large values, for example, a smart meter at a factory always reports a greater electricity consumption than a smart meter at a residential apartment at any given time, the condition for [99] is always true. However, in some other cases (e.g., the MovieLens dataset used in our experiments), the above condition will not be achieved. The latter solution [140] suffers from the problem of low data utility

[1]In this chapter, we use the naive scheme of [140] because it has the same task as [99] and our work. That is, we all aim to guarantee the global DP only by injecting local DP noise at the data curators.

by using the aggregator's global function sensitivity (defined in Equation (2.2), Chapter 2) for each data curator. Table 5.1 presents a counterexample for parallel composition in [99] where t is the timestamp and x_i is data curator i. That is, for a summation function f, $\Delta f_L(X) < \sum_i \Delta f_L(x_i)$. In Table 5.1, each data curator x_i calculates the sum of their local data, for example, $f(x_3) = 1+1+1+27 = 30$. We also have their local sensitivity $\Delta f_L(x_i) = \max_{j=0}^{4}\{x_{i,j}\}$, for example, $\Delta f_L(x_1) = \max\{1,2,3,4\} = 4$, then the sum of each data curator's local sensitivity is $\sum_{i=1}^{3} \Delta f_L(x_i) = 4+8+27 = 39$. The local sensitivity of the overall aggregation function is $\Delta f_L(X) = \max_{i=1}^{3}\{f(x_i)\} = 30$. Thus we have $30 = \Delta f_L(X) < \sum_i \Delta f_L(x_i) = 39$, such a case is not considered in [99].

Table 5.1 Counterexample for Parallel Composition in [99].

t	x_1	x_2	x_3
0	1	5	1
1	2	6	1
2	3	7	1
3	4	8	27
$f(x_i)$	10	26	30
$\Delta f_L(x_i)$	4	8	27

Therefore, in this chapter, we aim to provide a new lower bound of the parallel composition of the privacy budget for differentially private data aggregation for a summation (or counting) function. This uses the local sensitivity of the summation function to inject less noise than [140], while such sensitivity has no condition as in [99]. We summarise our main contributions in this chapter as follows:

- We use the local sensitivity of the summation function to calculate the lower bound of parallel composition on the privacy budget, so that we have less noise injected compared to [140] to ensure better data utility with the same privacy requirements.

- We do not set any condition on the local sensitivities between the data aggregator and the data curator, such that, compared to [99], our result fits more general datasets to differentially private data aggregation using a summation function.
- We mathematically prove that our lower bound of parallel composition of the privacy budget also works with two common scenarios of time-serial data aggregation: passive data updates (malicious data cleansing) and active data updates (dynamically adding or removing data curators and/or serially refreshing data).
- We experimentally evaluate our result over two real-world datasets to illustrate the performance of our privacy budget bound across various experimental settings.

We organise the remainder of the chapter as follows. We show our lower bound of privacy budget and the proofs in Section 5.2. In Section 5.3, we study the performance of our result using two real-life data update scenarios. Then, in Section 5.4, we experimentally evaluate the performance of our privacy budget bounds with two real-world datasets. Finally, Section 5.5 concludes this chapter.

5.2 Unconditional Lower Bound of Privacy Budget

In this chapter, the major privacy risk comes from collusion attacks, where partial computing participants (data curators) share data (inputs) that they provide to data aggregators, and then attempt to infer victims' private information by comparing the published aggregation outputs against colluding shares. Given that we do not know the exact number of curators who may join in collusion attacks, we make the strongest assumption that $n-1$ out of n data curators are colluders.

Naturally, the traditional concept of DP is suitable to handle the above worst case because of its strongest assumption about an adversary's background knowledge. Differing from traditional DP, our objective is to guarantee ε-DDP to preserve a data curator's privacy against the strongest collusion attacks. According to the DDP achieved in [99] and [140], we also guarantee the global DP (with local sensitivity) by ensuring local DP for each data curator.

According to Definition 2.5 (in Chapter 2), for a mechanism $\mathscr{T}$ to be ε-global differentially private, we should have:

$$\mathscr{T}(X) = f(X) + DPnoise(\varepsilon, \Delta f(X)),$$

namely,

$$\frac{\Pr[\mathscr{T}(X) \in S]}{\Pr[\mathscr{T}(X') \in S]} \leq \exp(\varepsilon \times \Delta f(X)), \tag{5.1}$$

where $X = (x_1, x_2, \ldots, x_n)$ and X' is a neighbouring set of X and they have all the same elements except one: $|\ \|X\| - \|X'\|\ | = 1$.

Given that the global DP should preserve privacy against the strongest collusion attacks, we have the following Theorem 5.1 to find a lower bound of the privacy budget ε to achieve the global DP in Equation (5.1) by implementing the local DP for all data curators.

Theorem 5.1. *Let $\mathscr{T}_i$ provide ε_i-DP, $\forall i \in \mathscr{U}$, and then the parallel composition of $\mathscr{T}_i$ will be*

$$\max_{i\in\mathscr{U}} \left\{ \frac{|\mathscr{U}| \cdot \Delta f_L(x_i)}{f(x_i)} \varepsilon_i \right\}\text{-}\textit{differentially private,}$$

where f is a summation (or counting) function, $\mathscr{U}$ is the set of data curators.

Proof. For each data curator i, we have guaranteed the local DP,

$$\mathscr{T}_i(x_i) = f(x_i) + DPnoise(\varepsilon_i, \Delta f_L(x_i)),$$

that is,

$$\frac{\Pr[\mathscr{T}_i(x_i) \in s_i]}{\Pr[\mathscr{T}_i(x_i') \in s_i]} \leq \exp(\varepsilon_i \times \Delta f_L(x_i)).$$

Then according to [99], the parallel composition of $\mathscr{T}_i$s is,

$$\Pr[\mathscr{T}(X) \in S] = \prod_{i \in \mathscr{U}} \Pr[\mathscr{T}_i(x_i) \in s_i]$$

Therefore,

$$\begin{aligned}
&\frac{\Pr[\mathscr{T}(X) \in S]}{\Pr[\mathscr{T}(X') \in S]} = \prod_{i \in \mathscr{U}} \frac{\Pr[\mathscr{T}_i(x_i) \in s_i]}{\Pr[\mathscr{T}_i(x_i') \in s_i]} \\
&= \prod_{i \in \mathscr{U}} \exp\left(\varepsilon_i \times \Delta f_L(x_i)\right) \\
&= \exp\left(\sum_{i \in \mathscr{U}} \varepsilon_i \times \Delta f_L(x_i)\right) \\
&= \exp\left(\sum_{i \in \mathscr{U}} \frac{\varepsilon_i \times \Delta f_L(x_i)}{f(x_i)} \times f(x_i)\right) \\
&\leq \exp\left(\max_{i \in \mathscr{U}} \left\{\frac{\Delta f_L(x_i)}{f(x_i)} \varepsilon_i\right\} \times \sum_{i \in \mathscr{U}} f(x_i)\right) \\
&\leq \exp\left(\max_{i \in \mathscr{U}} \left\{\frac{\Delta f_L(x_i)}{f(x_i)} \varepsilon_i\right\} \times |\mathscr{U}| \times \max_{i \in \mathscr{U}} \{f(x_i)\}\right) \\
&= \exp\left(\max_{i \in \mathscr{U}} \left\{\frac{|\mathscr{U}| \cdot \Delta f_L(x_i)}{f(x_i)} \varepsilon_i\right\} \times \Delta f_L(X)\right)
\end{aligned}$$

Recalling the global DP guranteed by Equation (5.1) and the DP definition (Definition 2.5, Chapter 2), let ε in Equation (5.1) be $\max_{i \in \mathscr{U}} \left\{\frac{|\mathscr{U}| \cdot \Delta f_L(x_i)}{f(x_i)} \varepsilon_i\right\}$, and then we prove that the parallel composition of $\mathscr{T}_i$ is $\max_{i \in \mathscr{U}} \left\{\frac{|\mathscr{U}| \cdot \Delta f_L(x_i)}{f(x_i)} \varepsilon_i\right\}$-differentially private, if $\mathscr{T}_i$ is ε_i-differentially private. $\square$

In the proof of Theorem 5.1, because the query function f is a summation (or counting) function, then $\max_{i \in \mathscr{U}} \{f(x_i)\} = \Delta f_L(X)$. In contrast to [99], we do not have any condition on the value of $\Delta f_L(X)$. Actually, since we relax the condition $\Delta f_L(X) \geq \sum_{i \in \mathscr{U}} \Delta f_L(x_i)$ in [99], we treat our result as an approximation of the lower bound of the privacy budget for distributed DP. Note that once a dataset meets the condition of $\Delta f_L(X) \geq \sum_{i \in \mathscr{U}} \Delta f_L(x_i)$ in [99], our lower bound of privacy budget would be greater than the result of [99]. Namely, in that case, our result would inject more differential noise than [99]. Further, we use the local sensitivity $\Delta f_L(X)$, which is much smaller than the global sensitivity $\Delta f_G(X)$; therefore, we could provide better data utility than [140].

Additionally, to learn the worst case of the performance of distributed DP, we also study the upper bound of $\Delta f_L(X) = \max_{i\in\mathscr{U}}\{f_L(x_i)\}$ in Theorem 5.2, that is, the relationship between $\Delta f_L(X)$ and $\sum_{i\in\mathscr{U}} \Delta f_L(x_i)$, where f is the summation (or counting) function.

Theorem 5.2. *Let database x_i have m_i records and a summation (or counting) function f on x_i has local sensitivity $\Delta f_L(x_i)$, then the upper bound of local sensitivity of f on $X = (x_1, x_2, \ldots, x_n)$, $\Delta f_L(X) = \max_{i\in\mathscr{U}}\{f_L(x_i)\}$ will be* $\max_{i\in\mathscr{U}}\{m_i\} \times \sum_{i\in\mathscr{U}} \Delta f_L(x_i)$.

Proof. Let

$$S_1 = \sum_{i\in\mathscr{U}} \Delta f_L(x_i)$$

$$S_2 = \max_{i\in\mathscr{U}}\{\Delta f_L(x_i) \times m_i\}$$

As we know, $\Delta f_L(X) = \max_{i\in\mathscr{U}}\{f_L(x_i)\} \le S_2$; thus, if we can find a k so that $\Delta f_L(X) \le S_2 \le kS_1$, then we can use the kS_1 as an upper bound of $\Delta f_L(X)$. Then, the value of k can be S_2/S_1, that is,

$$\begin{aligned} k = \frac{S_2}{S_1} &= \frac{\max_{i\in\mathscr{U}}\{\Delta f_L(x_i) \times m_i\}}{\sum_{i\in\mathscr{U}} \Delta f_L(x_i)} \\ &\le \frac{\max_{i\in\mathscr{U}}\{\Delta f_L(x_i) \times m_i\}}{\max_{i\in\mathscr{U}} \Delta f_L(x_i)} \\ &\le \max_{i\in\mathscr{U}}\{m_i\} \end{aligned}$$

Therefore, $\Delta f_L(X) \le \max_{i\in\mathscr{U}}\{m_i\} \times \sum_{i\in\mathscr{U}} \Delta f_L(x_i)$. □

Algorithm 5.1 shows an approach to guarantee the distributed DP for a summation and counting function in a star network.

Algorithm 5.1: Differentially Private Data Aggregation.

```
Input  : Original data curator's database: $\mathscr{X} = \{x_1, x_2, \dots, x_n\}$;
         Data aggregation function: $f$;
         Local DP budget, $\varepsilon_i$, for each data curator $i$.
Output : Differentially private data aggregation output: $\mathscr{R}$.
1 Data curator $i$ generates DP noise $DP_i$ by $\varepsilon_i$ and its local function sensitivity $\Delta f_L(x_i)$,
   i.e., $DP_i = DP(\varepsilon_i, \Delta f_L(x_i))$, $\forall i \in \mathscr{U}$;
2 Data curator $i$ sends $f'(x_i) = f(x_i) + DP(\varepsilon_i, \Delta f_L(x_i))$ to the data aggregator, $\forall i \in \mathscr{U}$;
3 Data aggregator computes
4 $\mathscr{R} = f(f'(x_1), f'(x_2), \dots, f'(x_n))$;
```

5.3 Privacy Performance with Continuous Data Updates

In this section, we study the privacy performance (i.e., lower bound of privacy budget ε for global DP) of Algorithm 5.1 when we update data in real-world applications. Specifically, our discussion will mainly focuses on two types of data update: passive data updates (e.g., malicious data cleansing) and active data updates (e.g., dynamically adding/removing data curator and/or serially refreshing data).

Case 1: Passive Data Updates. In real-world applications, to mislead data aggregation, a malicious data curator i may arbitrarily pollute his or her original data $f(x_i)$ as $f'(x_i) = f(x_i) + \delta$, where δ is the arbitrary pollution. To ensure the correctness of the data aggregation, it is necessary to cleanse the polluted data. Currently, there are several ways to recover the original data from the polluted data, such as compressive sensing [70]. Instead of studying techniques for data recovery (such as determining what data to recover), in this chapter, we only focus on the lower bound of privacy budget for a distributed differentially private data summation (or counting) function (Figure 5.1) after cleansing the polluted data. There are two cases that should be considered: (1) when malicious data curators submit the polluted data with local DP guarantees and (2) when malicious data curators only submit the polluted data, but do not provide local DP with their submissions. We name the set of malicious data curators $\mathscr{M}$, the set of honest data curators as $\mathscr{H}$, and the set of all data curators as $\mathscr{U} = \mathscr{M} \cup \mathscr{H}$, where $\mathscr{M} \cap \mathscr{H} = \varnothing$.

Malicious data with DP guarantees. In this case, once polluted data are cleansed, the information submitted by a malicious adversary i would also be $\mathscr{T}_i(x_i) = f(x_i) + DPnoise(\varepsilon_i, \Delta f_L(x_i))$, $\forall i \in \mathscr{M}$, regardless of whether the adversary adds DP noise before or after the polluted data injection. Therefore, the lower bound of ε for global DP will be guaranteed by Theorem 5.1.

Malicious data without DP guarantees. After cleansing the polluted data (arbitrary value), the data submitted by a malicious data curator j, who did not provide any DP guarantees on the submitted data, will be $\mathscr{T}_j(x_j) = f(x_j)$. In this case, we use Proposition 5.1 to show the lower bound of the privacy budget ε for such a case.

Proposition 5.1. *Let $\mathscr{T}_i$ provide ε_i-DP, $i \in \mathscr{H}$, $\mathscr{T}_j$ does not provide DP, $j \in \mathscr{M}$, then the parallel composition of all $\mathscr{T}_i$ and $\mathscr{T}_j$ will not have DP guarantees.*

Proof. $\because \forall j \in \mathscr{M}$, $\mathscr{T}_j$ does not provide DP,
$\therefore \forall j \in \mathscr{M}$, $\mathscr{T}_j(x_j) = f(x_j) = f(x_j) + 0 = f(x_j) + DPnoise(\varepsilon_j, \Delta f_L(x_j))$. According to Definition 2.5, when a mechanism $\mathscr{T}_j$ does not provide DP guarantee $(DPnoise(\varepsilon_j, \Delta f_L(x_j)) = 0)$, $\varepsilon_j = +\infty > \varepsilon_i$, $\forall i \in \mathscr{H}$, then, based on the conclusion in Theorem 5.1, the parallel composition of all $\mathscr{T}_i$ and $\mathscr{T}_j$ will be $\{+\infty\}$-differentially private; hence we do not have any DP guarantees. □

Therefore, in Case 1, to guarantee distributed DP in data aggregation by applying a summation and counting function for the network in Figure 5.1, we must assume that the malicious adversaries only arbitrarily pollute the data, but faithfully follow the local DP protocol.

Case 2: Active Data Updates. In real-world applications, we may have the following cases: new data curators join, existing data curators leave, and the data curators regularly refresh their private data. In Proposition 5.2, we study the lower bound of the privacy budget for such cases.

Proposition 5.2. *If the current parallel composition of $\mathscr{T}_i$ provides* $\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}$*-DP, then adding or removing a $\mathscr{T}_j$, and refreshing data by a mechanism $\mathscr{T}_i$ will not affect the DP guarantee, unless there is a $\mathscr{T}_k$ that does not provide DP after an update.*

Proof. Assume the existing differentially private mechanism $\mathscr{T}$ satisfies:

$$\begin{aligned}&\frac{\Pr[\mathscr{T}(X)\in S]}{\Pr[\mathscr{T}(X')\in S]}\\ \leq&\exp\left(\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}\times\Delta f_L(X)\right),\end{aligned}$$

where $\mathscr{U}$ is the set of all existing data curators.

Then when we add a $\mathscr{T}_j$ that provides ε_j-DP, the parallel composition of the current $\mathscr{T}$ and $\mathscr{T}_j$ will be,

$$\begin{aligned}&\frac{\Pr[\mathscr{T}(X)\in S]}{\Pr[\mathscr{T}(X')\in S]}\times\frac{\Pr[\mathscr{T}_j(x_j)\in s_j]}{\Pr[\mathscr{T}_j(x'_j)\in s_j]}\\ \leq&\exp\left(\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}\times\Delta f_L(X)\right)\times\exp(\varepsilon_j\times\Delta f_L(x_j))\\ =&\exp\left(\max_{i\in\mathscr{U}}\left\{\frac{\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}\times|\mathscr{U}|\times\max_{i\in\mathscr{U}}f(x_i)\right)\times\exp\left(\frac{\Delta f_L(x_j)}{f(x_j)}\varepsilon_j\times f(x_j)\right)\\ \leq&\exp\left(\max_{i\in\{\mathscr{U}+j\}}\left\{\frac{\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}\times(|\mathscr{U}|+1)\times\max_{i\in\{\mathscr{U}+j\}}f(x_i)\right)\\ =&\exp\left(\max_{i\in\mathscr{U}'}\left\{\frac{|\mathscr{U}'|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}\times\Delta f_L(X)\right)\end{aligned}$$

Given that the new data curator set U' contains all current data curators, including the recently joined one, the conclusion from Theorem 5.1 holds. Similarly, we can prove the case of removing a $\mathscr{T}_j$ in the same way by simply replacing $\times$ with $\div$ and replacing $+$ with $-$ from the above equations. Moreover, we can treat data refresh as a sequential composition of addition and deletion, and then prove its lower bound of privacy budget by the above in two steps (or applying the sequential composition property of Theorem 3 in [99]). □

Based on Theorem 5.1, Proposition 5.1 and Proposition 5.2, we have Theorem 5.3.

Theorem 5.3. *Let a data curator $\mathscr{T}_i$, $i \in \mathscr{U}$, provide ε_i-DP, then a parallel composition of $\mathscr{T}_i$ will guarantee* $\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}$*-DP in the case of distributed summation and counting, cleansing polluted data, dynamically adding or removing data curators, and refreshing data, unless there is a malicious data curator $\mathscr{T}_j$ that refuses to guarantee DP on its original database.*

5.4 Experimental Evaluation

In this section, we use two real-world datasets and evaluate the performance of the lower bound of the privacy budget ε for distributed DP. We start by introducing the two real-world datasets and the evaluation metric we used, and then compare the results of the experiments with [99] and [140].

5.4.1 Datasets, Evaluation Metric, and Experiment Settings

The datasets we use are the MovieLens 100K dataset[2] and Electricity Load Diagrams 2011-2014 dataset[3]. Specifically, in the MovieLens 100K dataset, 943 users posted $100,000$ ratings on $1,682$ films, where each user rated at least 20 films, and each film received 20 to 250 ratings. In the Electricity Load Diagrams 2011-2014 dataset, 370 electricity meters reported energy consumption every 15 minutes for four years from 2011 to 2014.

Table 5.2a is an example of the MovieLens 100K dataset, where x_i is user i and y_j is film j, and the values are the ratings from user i on film j. The example in Table 5.2a shows that

[2]https://grouplens.org/datasets/movielens/100k/
[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

people's taste in films may be entirely different (e.g., Film 4 and Film 5). There is also no user who always provides a high rating on movies. This property means that this dataset does not fit the conditions of the existing parallel composition result of [99] because it is possible that the summation of the data curators' sensitivities is greater than the aggregator's sensitivity. For example, in Table 5.2a, $\sum \Delta f_L(x_i) = 14 (= 4+5+5) > \Delta f_L(X) = 13$ (maximum of individual summation: x_1).

Table 5.2 Examples of Datasets.

(a) Example of ML-100 Dataset.

	x_1	x_2	x_3
y_1	1	0	1
y_2	2	1	1
y_3	2	2	1
y_4	4	1	1
y_5	0	4	1
y_6	4 (max in x_1)	5 (max in x_2)	5 (max in x_3)
sum	13 (max of the three sums)	13	10

(b) Example of ELD-1114 Dataset.

	x_1	x_2	x_3
y_1	51	14	100
y_2	62	11	110
y_3	61	22 (max in x_2)	105
y_4	49	14	107
y_5	70 (max in x_1)	0	120 (max in x_3)
y_6	54	15	115
sum	347	76	657 (max of the three sums)

Table 5.2b is an example of the Electricity Load Diagrams 2011–2014 dataset, where x_i is electricity meter i and y_j is a timestamp j and the values are the electricity consumption reading by meter i between time $j-1$ and j. The example in Table 5.2b shows that the electricity consumptions read by each meter are always stable. The one that reads larger numbers will always do so. This property fits the condition of [99] because, when some

data curators always have greater data values and the summation of the data curators' sensitivities is always less than the aggregator's sensitivity. For example, in Table 5.2b, $\sum \Delta f_L(x_i) = 212(= 70+22+120) < \Delta f_L(X) = 657$ (maximum of individual summation: x_3).

The global lower bound of privacy budget in [99], [140] and this work provides an estimation of the global privacy performance for distributed differentially private data aggregation. In our experiments, we evaluate the privacy performance over different lower bounds of privacy budget based on the difference between noise injected from local DP and noise injected for global DP based on the estimated lower bound of the global privacy budget. A smaller difference indicates that the global lower bound of the privacy budget reflects the nature of distributed DP more accurately. To measure the difference, we use the mean absolute error (*MAE*):

$$MAE = \frac{1}{N}\left|\sum_{i=1}^{N}(|G_i| - |L_i|)\right|,$$

where N is the total rounds of experiments, L_i is the overall noise added by data curators at experiment round i (local DP) and G_i is the overall noise added by the global lower bound of the privacy budget (global DP). In our experiments, we set N = 100K. Clearly, a smaller value of *MAE* means a smaller difference between the estimated global DP guarantee and the local DP guarantee. Therefore, a smaller *MAE* shows better privacy performance results from the corresponding global lower bound of privacy budget.

For convenience, in our experiments, we assign the same epsilon value (X axis in the figures) for all data curators in one experiment for a given privacy level. Additionally, Shi et al. [140] used the global function sensitivity for the data aggregator, $\Delta f_G(X)$, for local DP noise injection, and then estimated a global lower bound of the privacy budget $\sum_{i\in\mathscr{U}} \varepsilon_i$. Meanwhile, McSherry [99] used the local function sensitivity for data curators, $\Delta f_L(x_i)$, for local DP injection. Given that the noise injection parameter in [140] is completely different from that in [99]'s and ours, in the experiments for evaluating the global lower bound of the privacy budget, we only compare the privacy performance between [99] and our work. Moreover, the global privacy budget is computed as $\max_{i\in\mathscr{U}}\{\varepsilon_i\}$ in MaxEpsilon [99] and $\max_{i\in\mathscr{U}}\left\{\frac{|\mathscr{U}|\cdot\Delta f_L(x_i)}{f(x_i)}\varepsilon_i\right\}$ from Theorem 5.3. We use greater values for privacy budget for data curators because we want to explore the utility performance of summation function in an extreme case where data utility is the first priority. (Source code: https://github.com/suluz/dist_DP)

5.4.2 Experimental Results Analysis

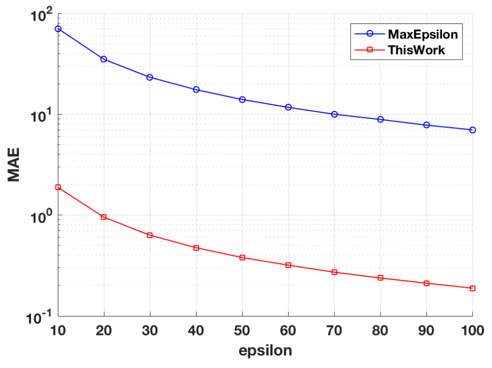
In the figures for our experimental results, we label the result of [99] as MaxEpsilon, the result of [140] as SumEpsilon, and this contribution as ThisWork. We rename the MovieLens 100K dataset as ML-100 and the Electricity Load Diagrams 2011-2014 dataset as ELD-1114.

Table 5.3 Comparison of Different Global Lower Bounds of Privacy Budget on Two Datasets.

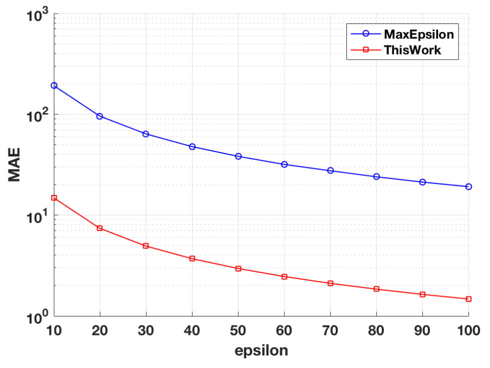
ε (used by data curators)	ML-100				ELD-1114	
	Summation		Counting		Summation ($\times 10^{-8}$)	
	MaxEpsilon	ThisWork	MaxEpsilon	ThisWork	MaxEpsilon	ThisWork
10	191.8323	14.7879	70.6503	1.8854	5.265	35.914
20	95.6142	7.3888	35.0246	0.9524	2.638	17.874
30	63.7804	4.9304	23.2179	0.6325	1.756	11.972
40	47.7364	3.6995	17.5044	0.4753	1.32	8.9504
50	38.228	2.957	13.9977	0.3798	1.0592	7.154
60	31.806	2.4607	11.6983	0.3159	0.8772	5.9543
70	27.5308	2.1067	9.9931	0.2716	0.7494	5.1044
80	23.9762	1.8447	8.8742	0.238	0.6534	4.4538
90	21.2536	1.6375	7.7922	0.2113	0.5846	3.9683
100	19.1398	1.4748	7.0025	0.1887	0.5267	3.5926

Figure 5.2 and Table 5.3 show the privacy performance of different global lower bounds of the privacy budget on the MovieLens 100K dataset (ML-100), where f is a counting function in Figure 5.2a and a summation function in Figure 5.2b. In the ML-100 dataset, $\Delta f_L(X) \leq \sum_{i\in\mathscr{U}} f_L(x_i)$ (e.g., when f is counting, we calculate the local sensitivity of the aggregation system, $\Delta f_L(X) = 730$, and the sum of the local sensitivities of the data curators, $\sum_{i\in\mathscr{U}} f_L(x_i) = 943$, based on the definition of the local sensitivity (Equation (2.3), Chapter 2)).

Based on the analyses in previous sections, the global lower bound of the privacy budget of MaxEpsilon will not be achieved in ML-100. That is, the noise added by the global lower bound of the privacy budget will be very different from the noise added by the local privacy budget. In fact, from both Figure 5.2a and Figure 5.2b, we can see that the noise injected by the global lower bound of privacy budget is closer to the real noise injected into the system compared to MaxEpsilon. Thus, our privacy budget provides better privacy performance than [99] in the case where $\Delta f_L(X) \leq \sum_{i\in\mathscr{U}} f_L(x_i)$.



(a) Counting



(b) Summation

Fig. 5.2 Privacy Performance on ML-100.

Figure 5.3 and Table 5.3 show the privacy performance (on the summation function) using the different global lower bounds of the privacy budget on the Electricity Load Diagrams dataset (ELD-1114). In the ELD-1114 dataset, $\Delta f_L(X) \geq \sum_{i\in\mathscr{U}} f_L(x_i)$; thus, according to Theorem 2.2 (in Chapter 2) and Theorem 5.3, the privacy performance of MaxEpsilon would be better than ours. In the case where $\Delta f_L(X) \geq \sum_{i\in\mathscr{U}} f_L(x_i)$, the lower bound of the privacy budget would be an approximation of the one in MaxEpsilon. In fact, Figure 5.3 clearly shows this. Further, if we compare the experimental results in Figure 5.2 and Figure 5.3, we can find that, in the cases where MaxEpsilon [99] does not work (Figure 5.2), the privacy performance of our global privacy budget is much better than [99]. In the cases where MaxEpsilon [99] provides $\max_{i\in\mathscr{U}} \varepsilon_i$-distributed DP correctly (Figure 5.3), the privacy performance of our privacy budget is close to MaxEpsilon [99]. Therefore, as an approximation of MaxEpsilon [99], the overall privacy performance of our privacy budget is satisfied. However, as our privacy budget does not rely on any condition, the lower bound of privacy budget in this chapter is more generally applicable.

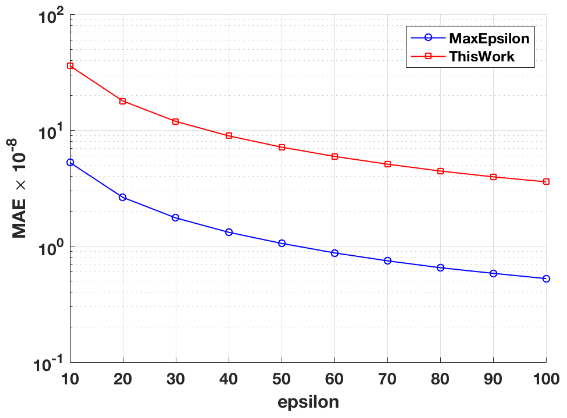
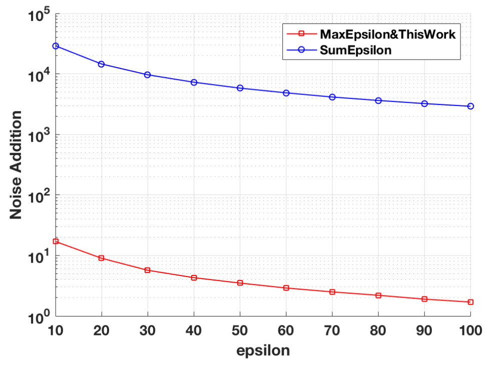


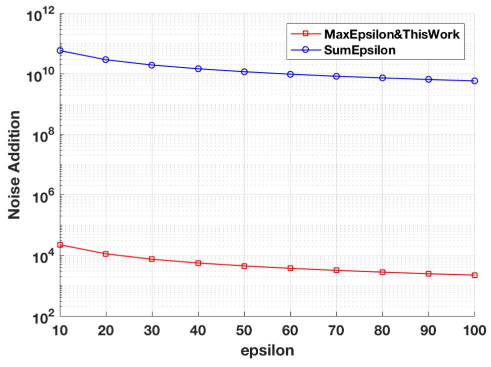
Fig. 5.3 Privacy Performance on ELD-1114 (summation).

In Figure 5.4 and Table 5.4, we compare the noise injected for local DP by SumEpsilon and ThisWork on the MovieLens 100K dataset (ML-100) and the Electricity Load Diagrams dataset (ELD-1114), where f is the summation function. As discussed in the analyses in Section 5.2, both Figure 5.4a and Figure 5.4b show that the noise injected by the schemes in [140] (using $\Delta f_G(X)$ to generate differentially private noise) is much greater than MaxEpsilon [99] where $\Delta f_L(x_i)$ is used to generate differentially private noise. The reason for these experimental results is that, according to the property of Laplace noise, a smaller function sensitivity would result in less noise injection. The summation of the individual privacy budgets scheme of [140] uses global sensitivity, which is much greater than the local sensitivity used by the maximum privacy budget scheme of [99] and ours; thus, MaxEpsilon and our approach provide much better data utility (less noise injection) than SumEpsilon in both Figure 5.4 and Table 5.4 with given privacy requirements (values of the privacy budgets).

In addition, we observe that all the curves in the figures of our experimental results show the same gradient, pattern or shape. This outcome occurs because, regardless of which final



(a) ML-100



(b) ELD-1114

Fig. 5.4 Noise Injection with Different Function Sensitivities for Local DP Guarantee (summation).

Table 5.4 Noise Injection with Different Function Sensitivities for Local DP Guarantee (summation).

ε (used by data curators)	ML-100		ELD-1114 ($\times 10^{-4}$)	
	SumEpsilon	ThisWork	SumEpsilon	ThisWork
10	29058	17	5870300	2.257
20	14562	9	2928800	1.1306
30	9655.5	5.7	1961600	0.7498
40	7280	4.3	1475000	0.564
50	5814.4	3.5	1177700	0.4516
60	4866.1	2.9	976270	0.3756
70	4147.6	2.5	833930	0.322
80	3641.2	2.2	734320	0.2815
90	3233	1.9	650930	0.2485
100	2915.4	1.7	584560	0.2246

privacy budgets and function sensitivities are used in the two schemes (sum or max), the privacy budget and function sensitivity are just two numeric parameters to noise injection in Figures 5.2, 5.3 and 5.4. Given that the two schemes follow the same noise injection distribution (Laplace distribution), they should have almost the same gradient or shape of curve, as illustrated in our figures.

5.5 Summary

To take advantage of sensor networks to collect and analyse data for better decision making, privacy is essential. Among all the data analysis functions, the summation and counting functions are two of the most important. To preserve individual privacy against the strongest collusion attacks, existing schemes either provide low data utility by using a global sensitivity of summation (or counting) function or use unknown privacy guarantees. To address such problems, in this chapter, we explored the privacy budget guarantee of distributed systems to offer a new lower bound of the parallel composition of privacy budgets with unconditional function sensitivity for both data aggregators and data curators in a distributed environment. Both the theoretical and experimental evaluations show that our result works in all conditions, in comparison with [99], and provides better data utility than [140], while achieving the same privacy requirements.

Chapter 6

Differentially Private Path Publishing

Paths in a given network (map) are a generalised form of time-serial chain that occur in many real-world applications, such as trajectories and Internet data flows. Differentially private trajectory publishing concerns publishing path information that is usable to the users, yet secure against adversaries, to reconstruct paths with maximum background knowledge. In existing studies, perturbed paths are published, where each vertex is sampled from a predefined set applying DP to replace the corresponding vertices in the original path. Such schemes are robust to protect the privacy of vertices of a given path/trajectory against adversaries that have knowledge of the whole path, but one missing vertex (and its associated edges) of the path. In this chapter, we relax such potential adversaries' background knowledge to be all except one edge of the path, and hence consider the scenario of more powerful adversaries with maximum background knowledge of the entire network topology and paths except for one (arbitrary) missing edge. Under such an assumption, the perturbed path produced by the existing work is vulnerable because the adversary can reconstruct the missing edge from the existence of edges in the perturbed path whose ends are close to the two ends of the missing edge. To address this vulnerability and effectively protect edge privacy, instead of publishing a perturbed path, we propose a novel scheme of graph-based path publishing to protect the original path by embedding the path in a graph that contains fake edges and replicated vertices applying DP techniques, such that only genuine users who have full knowledge of the network topology are able to recover the exact vertices and edges of the original path with high probability. We theoretically analyse the performance of our algorithm in terms of output quality: DP guarantees, utility and execution efficiency. We also conduct extensive experimental evaluations on a high-performance cluster system to validate our analytical results.

6.1 Introduction

Paths are pervasive in our daily life, e.g., geometric trajectories, internet traffic flows, supply chains and intelligence exchange networks [43, 56]. Publishing a path helps the owners of the path (or external researchers) discover more knowledge about the path and obtain information in the path. However, attackers may infer the privacy of a path from its published information by applying auxiliary information [22, 106]. Therefore, it is crucial to consider privacy disclosure risks when publishing a path.

In this chapter, we consider path publishing against adversaries with the maximum knowledge of the network topology (all vertices and edges except one arbitrary edge). That is, given a network $N = (V, E)$, which is not a complete graph, we publish a path $P = (V, E_P \subset E)$ protecting an arbitrary edge $e \in E_P$ against adversaries who have $N' = (V, E \setminus \{e\})$ containing $P' = (V, E_P \setminus \{e\})$. This problem can be found in some potential worst-case scenarios.

Disease Transmission Chain. In this scenario, N contains the social connections between patients, V is a set of patients from the same community, E is a set of relationships between patients and P is a disease transmission chain. Path publishing allows external researchers to study the way that the disease transmits, and "adversaries" are people that want to know the existence of relationship between the patients.

Intelligence Exchange. In this scenario, N is a spy network, V is a set of spies, E is a set of spy communication channels and P is the message exchange path. Path publishing is the way to let all spies know who should be contacted, and "adversaries" are counter-intelligence units that want to know how a targeted spy exchanges information with other spies.

Privacy-preserving path publishing against the maximum background knowledge is well studied in trajectory publishing with DP [33] because of its privacy-preserving guarantees against attackers potentially with arbitrary auxiliary information. There are two main tracks of research in differentially private trajectory publishing: set-of-trajectories publishing [21, 65, 113, 79, 19, 3, 58, 93] and (single-) trajectory publishing [71, 171, 61, 158, 157, 20]. In this chapter, we address the latter problem.

Existing studies in differentially private single-trajectory publishing share similar ideas. First, their solutions (in DP) are robust under potentially the same worst-case assumption of adversaries' background knowledge. That is, the adversaries do not have some locations of a given path. In the worst case, only one location (and its associated edge(s)) is missing at the attacker's side. Second, they use the exponential mechanism of DP (ExpDP) to implement DP. Specifically, the published trajectory contains fake locations only, which are sampled via ExpDP from a predefined area, to replace the real ones in the target trajectory. The differences over these methods mainly stem from the quality measurement for sampling fake locations based on different objectives, such as the spatiotemporal relationship between locations [171, 20], the movement directions in a trajectory [71] and the similarity between the real and fake locations [61, 158].

The state-of-the-art results in single-trajectory publishing achieve a satisfying trade-off between utility and privacy by DP against adversaries with missing vertices. However, we identify two weaknesses in existing work [71, 61, 171, 158, 20]. First, the assumption of missing vertices does not cover the worst scenario of adversaries with maximum background knowledge, that is, when an adversary has full knowledge of the map, including all vertices and edges, except one edge on the trajectory. In this case, the published (perturbed, privacy-preserved) path by existing solutions is vulnerable because an adversary could reconstruct the missing edge with knowledge of the existence of edges in the perturbed path whose two ends are close to the two ends of the missing edge. Note that the vertices' closeness could be measured by different metrics, such as spatiotemporal relationships, movement directions in a path and semantic similarities. Second, the existing work does not generally publish real vertices on the path, so it is difficult for genuine users (who have full information about the network) to reconstruct the original path from the published one to confirm the exact links between the real vertices. Figure 6.1 depicts an example attack against existing work, where fake/published vertices are spatially close to the actual vertices, considering the utility of the published trajectory. As a result of the spatial closeness between the published/perturbed vertices and the actual vertices (attackers' background knowledge), the attackers aim to infer the missing information by recognising the corresponding actual vertex/edge of each published vertex/edge, e.g., edge $(\hat{v}_1, \hat{v}_2)$ is the fake version of actual edge (v_1, v_2). Since we assume that the attackers have maximum background knowledge, i.e., all except one missing edge, and the fake edges/vertices maintain the connectivities of the actual edges/vertices, such a missing edge cannot be an edge that is missing in both fake and real paths, e.g., edge (v_1, v_3). Based on the published and axillary information, edge $(\hat{v}_2, \hat{v}_3)$ in the published path implies the existence of edge (v_2, v_3).

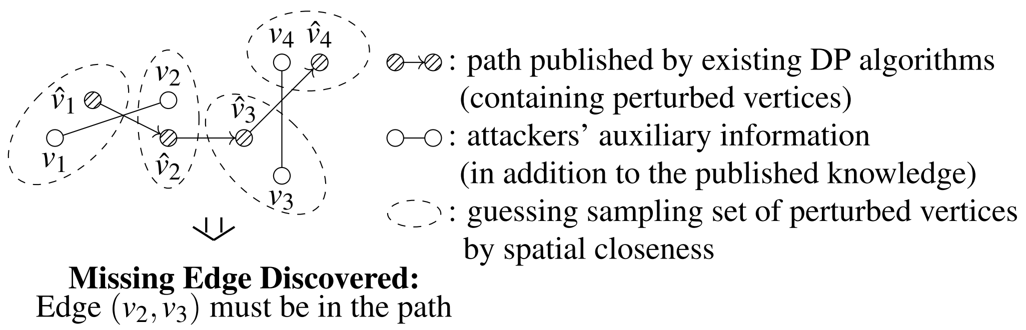

v_4 $\hat{v}_4$
v_2
$\hat{v}_1$
$\hat{v}_3$
v_1
$\hat{v}_2$
v_3
: path published by existing DP algorithms (containing perturbed vertices)
: attackers' auxiliary information (in addition to the published knowledge)
: guessing sampling set of perturbed vertices by spatial closeness
Missing Edge Discovered:
Edge (v_2, v_3) must be in the path

Fig. 6.1 Example Attack Inferring Privacy from Published Path of Existing DP Algorithms.

To fill the research gap, this chapter studies how to protect an arbitrary edge when publishing a path. In summary, our main contributions are as follows:

- We address the problem of differentially private path publishing in a generalised form against inference attacks from adversaries with more background knowledge than existing work. Differing from existing work, where the adversary knows the network topology and $n-1$ out of n vertices of the path, we preserve privacy against adversaries who have full knowledge of the network and all vertices on the path, except one missing edge of the path.

- Our proposed method guarantees high data utility and privacy (DP). That is, genuine users (who have full knowledge of the network) are able to reconstruct the path, but adversaries are unable to infer the missing edge and hence reconstruct the path. In particular, we publish a graph-based path where fake edges and duplicate real vertices are created to embed the real path into a graph for publishing, such that only genuine users are able to reconstruct the exact vertices and edges of the original path, and determine the visit order of the vertices with high probability.

- We evaluate key properties of the proposed method both theoretically and experimentally. Specifically, we mathematically analyse DP guarantees and the data utility of our scheme. We conduct experiments on a cluster system and extensively evaluate our scheme with synthetic datasets with various network sizes and densities.

To the best of our knowledge, this is the first work that considers edge privacy (i.e., the existence of an edge between two vertices), demanding through differentially private path publishing.

The remainder of this chapter is organised as follows. In Section 6.2, we provide a brief introduction of the preliminaries of this chapter, including the path and network models and the attackers' background knowledge. We then present our differentially private path publishing in Section 6.3. Mathematical analyses are also displayed in this section. Afterwards, in Section 6.4, we provide experimental evaluation results on the performance of our algorithms using synthetic datasets. Finally, we conclude the chapter in Section 6.5.

6.2 Preliminaries

In this section, we briefly introduce map and path models, as well as assumptions about adversaries' auxiliary information.

6.2.1 Network and Path Models

Without loss of generality, a network, such as, roadmap, internet node map or spy network, etc., is a bidirected/undirected yet incomplete graph, $N = (V, E)$, where V is the set of vertices and E is the set of edges. In this chapter, the network is a connected graph (every pair of vertices is connected by a path, rather than an edge) that contains the path P.

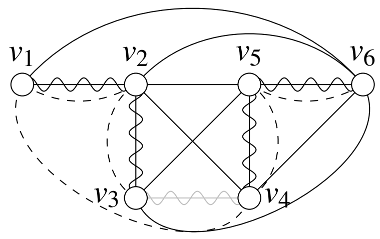
The path $P = (V, E_P \subset E)$ is a subset of the network N. We define a path as a sequence of vertices in a given order, that is, $P = \{v_1, v_2, \ldots, v_n\}$, where $v_i \in V$ is the ith vertex in P and $e_i = (v_i, v_{i+1}) \in E_P$ is the ith edge in P. Given that the path P may contain rings, we may have $\exists i \neq j$, $v_i(\in V) = v_j(\in V)$.

6.2.2 Adversaries' Auxiliary Information

In contrast to the assumption of adversaries' auxiliary information in existing work on differentially private trajectory publishing, we assume that adversaries have correct information about the vertices, but incorrect information about the edges of the path. We assume the worst case of the adversaries' auxiliary information, where the adversaries do not know the existence of one arbitrary edge in the original path P (and the network N), say $e_i = (v_i, v_{i+1})$. That is, the adversaries have $N' = (V, E')$, $P' = (V, E_{P'})$ that $e_i = E \setminus E' = E_P \setminus E_{P'}$.

6.2.3 An Example

Figure 6.2 illustrates an example of the models of network, path and adversaries' background knowledge, where edge (v_3, v_4) is the missing edge to the adversaries. Consider the example of publishing disease transmission chain here. The vertices (white dots) and edges (solid lines) in Figure 6.2 represent patients and their relationship, respectively. Path *P* (acyclic snake lines) is the disease transmission chain. In this given network/graph, there is only one unknown relationship (edge (v_3, v_4)) to adversaries. Clearly, relationships between patients in the transmission chain could include private information, for example, this could disclose information about extramarital sex.



$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$
E : ——
P : ∿∿∿
$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$
$E' : E \setminus \{(v_3, v_4)\}$
$P' : v_1 - v_2 - v_3, v_4 - v_5 - v_6$
The attacker guesses a possible path: - - - -
$v_6 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$

Fig. 6.2 Example of Network $N = (V, E)$, Path (The Real One *P*), and Adversaries' Partial Path as Background Knowledge (P').

Given that the adversaries only know that an unknown relationship (the missing edge) is a non-existent edge in network $N' = (V, E')$, in Figure 6.2, this unknown relationship could be edge (v_1, v_3), (v_1, v_4), (v_1, v_5) and (v_3, v_4). By considering these four possible edges and adversaries' partial path P', if we do not publish the real chain directly, the adversaries could have several possible transmission chains (acyclic paths). For example, it could be a path $v_6 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ (marked by dashed lines) or a path $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$ (that is the real one). Note that, since P' is a partial path, the adversaries cannot confirm the transmission direction of the original chain (path *P*); hence the adversaries may have other possible chains in a different transmission direction, such as, $v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$.

Therefore, our goal is to publish a privacy-preserved real path *P*. Such a published path allows the genuine users to recover the original path; however, the missing edge remains unknown for the adversaries that potentially have maximum background knowledge about the network.

6.3 Proposed Algorithms

In this section, we show a differentially private algorithm (and associated analysis) to preserve privacy for path publishing against adversaries with background knowledge, as described in Section 6.2.

6.3.1 Idea Overview

In short, we propose a graph-based path publishing that uses the topology of the original network and the connections between vertices of the path to conceal the edge existence against adversaries who do not have full knowledge about the edges in the given network. Our idea is inspired by the permutation idea in [57]. That is, genuine users' full knowledge about the network acts as a private key to recover the true information from the perturbed/published data.

In general, when building a graph-based path, we apply DP to introduce uncertainties to hide the exact information of edges in the path with the following privacy-preserving rules:

1. For the edge $e = (u, v) \in E_P$, vertex u and v are not placed in the same branch of the graph-based path.

2. For the edge $e = (u, v) \in E \setminus E_P$, vertex u and v are placed in the same branch of the graph-based path.

3. For the non-existent vertex connections $e = (u, v) \notin E$, vertex u and v are either in or not in the same branch of the graph-based path randomly (by DP).

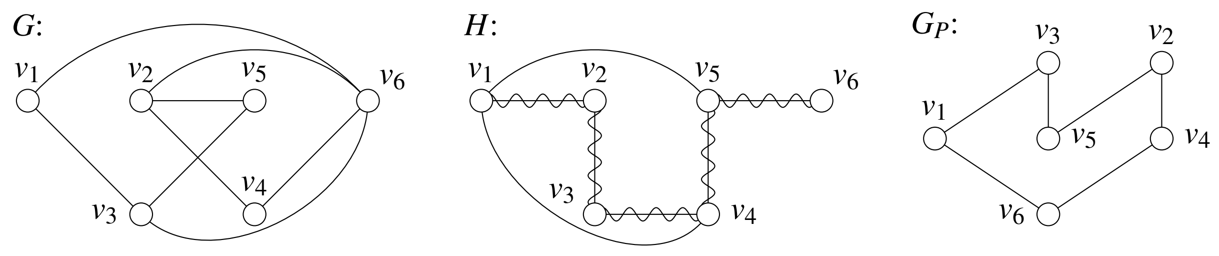
4. For vertices that appear multiple times in the graph-based path, this does not implicitly indicate whether those vertices are in a circle or not (because of DP).

5. For a group of nodes without a parent and a group of nodes without a child in the graph-based path, we place the group that contains the vertex with the closest index to the first node of the path as the root of the graph-based path.

In the above rules, Rule 1 and Rule 2 guarantee that the genuine users with full knowledge about the network can recover the exact vertices and edges of the original path. Rule 5 allows the genuine users to determine the visit order of the original path with high probability. In contrast, Rule 3 and Rule 4 introduce uncertainties to confuse adversaries who do not have full knowledge about P (i.e., missing edge(s)).

Figure 6.3 shows an example of producing a graph-based path by Rules 1, 2, 3 and 5 from a given network $N = (V, E)$, shown in Figure 6.2. We will show how Rule 4 works in the next section. We first construct a graph $G = (V_G = V, E_G)$ and G's complement $H = (V_H = V, E_H = U \setminus E_G)$, where U consists of all two-element subsets of V. For G and H, we have $(E \setminus E_P) \subset E_G$, $E_P \subset E_H$. Given that N is not a complete graph, it is clear that G and H contain fake edges not belonging to N, that is, $(E_G \cup E_H) \setminus E \neq \emptyset$. For example, in Figure 6.3, edge (v_1, v_3) in G, and edges (v_1, v_4) and (v_1, v_5) in H are non-existence edges of N (in Figure 6.2). Note that H is a Hamilton graph, e.g., P is one Hamilton path in H. Next, if $e = (u, v) \in E_G$, we place vertices u and v to the same branch of G_P (e.g., vertices v_3 and v_6); otherwise, we place vertices u and v to different branches of G_P (e.g., vertices v_2 and v_3). Finally, we choose v_2 and v_3, rather than v_6, as the root of G_P because v_2 has the closest index to the first node (v_1) of the original path among all vertices in root and leaf: v_2, v_3 and v_6. Note that, in this chapter, the following concepts, *root*, *leaf*, *branch* and *level*, follow the same meaning as that of the tree (or forest) data structure.

When the genuine users receive G_P, they recover the original path by two steps. First, they recover the path without knowing the correct direction by Rule 1 and 2 based on their full information about the network N. Second, after recognising v_1 and v_6 as the two ends of the path, they confirm v_1 as the first node of the path by Rule 5 because v_2, as one of the roots



Network N contains an acyclic path P (Figure 6.2) $\rightarrow (G,H) \rightarrow$ graph-based path G_P

Fig. 6.3 Example of Producing the Graph-based Path.

of G_P, has the closest index to v_1 rather than v_6. In contrast, according to the rules applied on G_P, the missing edge could be (v_1,v_4), (v_1,v_5) and (v_3,v_4) because those vertices pairs are in different branches of G_P. That is, the published G_P maintains the indistinguishability between the real missing edge (v_3,v_4) and non-existent edges (v_1,v_4) and (v_1,v_5) to the adversaries.

Further, G and G_P are equivalent for graph-based path publishing. G hides the path information, and potentially indicates the visit order with Rule 5. However, it can require a high computational cost to recover the real path. We demonstrate that it is more efficient to recover the original path from G_P than that from G in Corollary 6.1 in Section 6.3.3.

Table 6.1 lists the notations used in this chapter.

Table 6.1 Summary of Notations.

Notation	Description
$\cdot'$	The corresponding notation of adversaries' background knowledge
$\hat{\cdot}$	The processed $\cdot$
E; E_G; E_P	The set of edges in N; in G; in P
e, e_i	Edges
G	The input graph to produce G_P
G_P	The published graph-based path
H	The complement graph of G
N	The network/graph containing the path P
P	The original path for publishing
V; V_G	The set of vertices in N (and P); in G
v_i, $v_i^{(j)}$, v, u	The vertices

6.3.2 Map Pre-processing

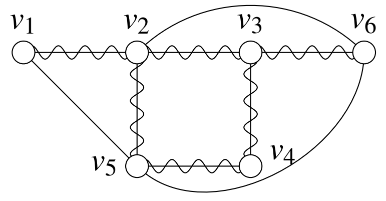
In this section, we pre-process the given network to meet the privacy-preserving Rule 3 and Rule 4 by injecting duplicate vertices and fake edges with the exponential mechanism of DP (ExpDP).

Differentially Private Vertices Pre-processing

Pre-processing vertices requires an acyclic path in H; otherwise, even genuine users cannot recover the exact path. For example, consider using the original network containing a cyclic path in Figure 6.4 (top) to produce a G_P. It is easy to conclude that we cannot recognise whether v_5 or v_3 is the successor of v_2 when we visit v_2 for the first time.

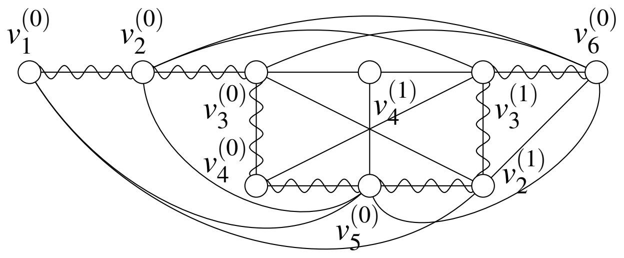
To convert a network to fit Rule 4 and the acyclic requirement, we apply the inheritance concept in object-oriented programming (OOP) to create base vertices and sub-vertex(s) in a network. This idea comes from a feature of cyclic paths, that is, when there is a cyclic path in the network, at least one vertex is repeatedly visited.

The given network:



Cyclic path: $v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_2 \to v_3 \to v_6$

Processed network with DP vertices:



Acyclic path: $v_1^{(0)} \to v_2^{(0)} \to v_3^{(0)} \to v_4^{(0)} \to v_5^{(0)} \to v_2^{(1)} \to v_3^{(1)} \to v_6^{(0)}$

Fig. 6.4 Example of DP Vertices Pre-processing.

All the unique vertices in a given network N are *base vertices*, and all repeated visited vertices in the path P and the duplicate vertices created by DP are sub-vertices. The relationship between base vertex and sub-vertex(s) should be publicly available; otherwise, the genuine users cannot recognise the sub-vertices to reconstruct the original path. Similar to inheritance in OOP, each sub-vertex inherits the connections from its corresponding base vertex. When inheriting a connection, the sub-vertices have to decide whether such a connection belongs to G or H. Figure 6.4 illustrates the idea of base vertex and sub-vertices. For example, in Figure 6.4, because vertex v_2 and v_3 are connected in the network N, their sub-vertices: $v_2^{(0)}$, $v_2^{(1)}$ and $v_3^{(0)}$, $v_3^{(1)}$ are connected in the processed network $\hat{N}$. Moreover, according to the cyclic path in the original network, $(v_2^{(0)}, v_3^{(0)}) \in E_P$, but $(v_2^{(0)}, v_3^{(1)}) \notin E_P$. Note that, since, in the proposed idea, the genuine users need the connections between real vertices to recover the actual path, we design the published graph-based path containing all base vertices and sub-vertices. Hence, it is natural that such information is publicly available.

Although such an operation converts a cyclic path to an acyclic one, there is a potential privacy disclosure risk against the adversaries knowing $m-1$ out of m edges as the background knowledge. Given that the relationship between the base vertex and sub-vertices is publicly available, it is easy for adversaries to learn whether or not there is a ring in the path based on those sub-vertices to subsequently infer the missing edge. Therefore, to preserve privacy (hide the real ring of the path) against such adversaries, we apply ExpDP to create more sub-vertices (duplicate/fake ones), for example, vertex $v_4^{(1)}$ in Figure 6.4 is a duplicate/fake sub-vertex created with ExpDP.

To create the duplicate/fake sub-vertices, the main task is related to the number of sub-vertices for a given base vertex. To minimise the overall number of vertices after pre-processing, we apply ExpDP to sample a number from a range $[num, \max\{maxNum, 2\}]$ for each base vertex, where num is the number of appearances of a given base vertex in the original path, $maxNum = \max\{num\}$ for all base vertices, e.g., $maxNum = 2$ in the cyclic path in Figure 6.4 because the maximum appearance of a vertex is 2 (based on vertex v_2 and v_3).

To confuse adversaries about whether or not there is a ring in the original path P, we want the numbers of sub-vertices of base vertices to be indistinguishable. That is, a base vertex with a smaller degree implies that there is less chance of being visited repeatedly; hence, it would have a higher probability of having more duplicate sub-vertices. For simplicity, we have the following quality function to measure the number (n) of duplicate sub-vertices for base vertex i, where d_i is the degree of base vertex i:

$$q_v(n,i) = \max\{n\} - n + \max\{d_i\} - d_i. \tag{6.1}$$

Depending on the adversaries' background knowledge, the sensitivity of $q_v(n,i)$ is determined by the maximum degree difference when there is an edge in the path missing. That is, $\Delta q_v = \max\{|q_v(n,i) - q_v(n,i')|\} = 1, \forall$ base vertex i.

Algorithm 6.1 shows how we convert a cyclic path to an acyclic one, while adding duplicate sub-vertices to the processed network with ExpDP (a.k.a differentially private ring removal). Lemma 6.1 proves the DP guarantees of Algorithm 6.1.

Lemma 6.1. *Algorithm 6.1 is ε_v-differentially private.*

Proof. For vertices pre-processing, we add duplicate vertices to the original network with quality function q_v (Equation (6.1)) and privacy budget ε_v. We use Equation (6.2) to calculate the overall privacy budget for this step, where V_P is the set of real vertices in the path P ($\exists u, v \in V_P, u = v$) and V_m is the set of real and duplicate vertices. Note that the difference between V_P and V_P' is that we have two pairs of (v, v') such that $|v.degree - v'.degree| = 1$, where $v \in V_P$, $v' \in V_P'$ and $v = v'$. We assume that the path is acyclic, so we split the set V_P into two subsets: V_s and V_r, where V_s contains the vertices appearing only once in the path

Algorithm 6.1: Pre-processing Differentially Private Vertices (a.k.a. Differentially Private Ring Removal).

```
Input  : A network $N = (V, E)$;
         A path $P = \{v_1, v_2, \dots, v_n\}$, $n \geq |V|$;
         A quality function: $q_v$;
         A privacy budgets: $\varepsilon_v$.
Output : A Hamilton network $\hat{N} = (\hat{V}, \hat{E})$;
         An acyclic path $\hat{P}$.
1  $(\hat{V}, \hat{E}) \leftarrow (V, E)$;
2  $\hat{P} \leftarrow P$;
3  $maxDegree \leftarrow \max\{v.degree \text{ in } P, \forall v \in V\}$;
4  for each $v \in V$ do
5      $v.subNum \leftarrow$ Sample a number from $[v.baseNum, maxDegree]$ with $q_v$
         (Eq. (6.1)) and $\varepsilon_v$;
6      for $i \leftarrow 2$ to $v.subNum$ do
7          $v.sub_{i-1} \leftarrow v$;
8          if $i \leq v.baseNum$ then
9              replace $i$th $v \in \hat{P}$ with $v.sub_{i-1}$;
10         else
11             $\hat{P} \leftarrow \hat{P}.push(v.sub_{i-1})$;
12 for each $v \in V$ do
13     for each $u \in V$ where $e = (v, u) \in E$ do
14         for each $u.sub_i \notin \hat{V}$ do
15             $\hat{V} \leftarrow \hat{V} + \{u.sub_i\}$;
16             $\hat{E} \leftarrow \hat{E} + \{(v, u.sub_i)\}$;
17 return: $\hat{N}$, $\hat{P}$;
```

and V_r contains the vertices that are repeatedly visited.

$$\begin{aligned}
&\frac{\Pr[V_P \rightarrow V_m]}{\Pr[V'_P \rightarrow V_m]} \\
=&\frac{\Pr[V_s \rightarrow V_{m0}] \times \Pr[V_r \rightarrow V_{m1}]}{\Pr[V'_s \rightarrow V_{m0}] \times \Pr[V'_r \rightarrow V_{m1}]} \\
=&\prod_{v.degree=v'.degree} \frac{\Pr[v.subNum = s]}{\Pr[v'.subNum = s]} \times \prod_{v.degree \neq v'.degree} \frac{\Pr[v.subNum = s]}{\Pr[v'.subNum = s]} \\
=&\prod_{v.degree \neq v'.degree} \frac{\Pr[v.subNum = s]}{\Pr[v'.subNum = s]} \\
\leq&\exp(\max\{\varepsilon_v\}) = \exp(\varepsilon_v).
\end{aligned} \tag{6.2}$$

The last step of Equation (6.2) comes from the parallel composition of DP [99]. Therefore, this lemma holds. □

Differentially Private Edge Pre-processing

When a network N contains an acyclic path P, irrespective of whether an edge belongs to the path or not, we split the processed network $N = (V, E)$ into two graphs: $G = (V_G = V, E_G = E \setminus E_P)$ and $H = P$. Then, to fit Rule 3, we turn all the non-existent vertex connections in this network to edges either in G or H randomly using ExpDP.

To do so, we first create a vertices relationship matrix $\boldsymbol{R}_{|V|\times|V|}$ for the path according to the real topology of the network, where

$$\begin{cases} r_{u,v} = 2, & \text{if } e = (u,v) \in E \setminus E_P; \\ r_{u,v} = 1, & \text{if } e = (u,v) \in E_P; \\ r_{u,v} = 0, & \text{if } e = (u,v) \notin E; \\ r_{u,v} = -1, & \text{if } u = v. \end{cases} \tag{6.3}$$

Now we replace the non-existent edges ($r = 0$) as fake edges either in the path ($r = 1$) or not ($r = 2$) with ExpDP. To conceal any possible missing edges for adversaries with background knowledge, we want to have more fake edges. Therefore, $r = 2$ should have a higher quality when implementing ExpDP to add fake edges. Given that adversaries may have a different number of edges than the original path, to measure the quality of $r = 1$ and $r = 2$, we have $q_e(2) = (|E|-1)/|E|$, $q_e(1) = 1/|E|$, namely

$$q_e(x) = \frac{|E|-2}{|E|}x - \frac{|E|-3}{|E|}, \tag{6.4}$$

where $\Delta q_e = |q_e(1) - q_e(2)| = (|E|-2)/|E| \approx 1$.

```
Algorithm 6.2: Pre-process Differentially Private Edges.
Input  : A network $N = (V, E)$;
         A path $P = \{v_1, v_2, \dots, v_n\}$;
         A quality function: $q_e$;
         A privacy budgets: $\varepsilon_e$.
Output : A vertices relational matrix $\boldsymbol{R}_{|V|\times|V|}$.
1 $\boldsymbol{R}_{|V|\times|V|} \leftarrow$ Traverse $N$ from $v$, values in $\boldsymbol{R}$ determined by Eq. (6.3);
2 for each $r_{u,v} = 0$ do
3     $r_{u,v} \leftarrow$ 1 or 2 with $q_e$ (Eq. (6.4)) and $\varepsilon_e$;
4 return: $\boldsymbol{R}_{|V|\times|V|}$;
```

Algorithm 6.2 shows the differentially private edge pre-processing. Lemma 6.2 proves the DP guarantee of Algorithm 6.2.

Lemma 6.2. *Algorithm 6.2 is $(\varepsilon_e + \ln 2)$-differentially private.*

Proof. For edges pre-processing, because we turn all non-existent connections in the network N to an edge in either graph G or graph H with a quality function q_e (Equation (6.4)) and privacy budget ε_e, we use Equation (6.5) to calculate the overall privacy budget.

$$\begin{aligned}
&\frac{\Pr[N \to (G,H)]}{\Pr[N' \to (G,H)]} \\
=&\prod_{edgesToG} \frac{\exp(\frac{\varepsilon_e q_e(2)}{2\Delta q_e})}{\exp(\frac{\varepsilon_e q_e(2)}{2\Delta q_e})} \times \prod_{edgesToH} \frac{\exp(\frac{\varepsilon_e q_e(1)}{2\Delta q_e})}{\exp(\frac{\varepsilon_e q_e(1)}{2\Delta q_e})} \times \frac{1}{\Pr[edge(= E \setminus E')ToH]} \\
=&\frac{\exp(\frac{\varepsilon_e q_e(2)}{2\Delta q_e}) + \exp(\frac{\varepsilon_e q_e(1)}{2\Delta q_e})}{\exp(\frac{\varepsilon_e q_e(1)}{2\Delta q_e})} \\
\le& 2\exp(\varepsilon_e) = \exp(\varepsilon_e + \ln 2).
\end{aligned} \tag{6.5}$$

Therefore, this lemma holds. □

6.3.3 Graph-based Path Production

In this section, we use Algorithm 6.3 to generate the graph-based path from the differentially private matrix $\boldsymbol{R}_{|V|\times|V|}$. In Algorithm 6.3, we first call Algorithm 6.1 and Algorithm 6.2

Algorithm 6.3: Publish Graph-based Differentially Private Path.

```
Input  : A network $N = (V,E)$;
         A path $P = \{v_1, v_2, \dots, v_n\}$;
         Two quality functions: $q_v$ and $q_e$;
         Two privacy budgets: $\varepsilon_v$ and $\varepsilon_e$.
Output : A graph-based path $G_P$.
1 $\hat{N} = (\hat{V}, \hat{E})$, $\hat{P} \leftarrow$ Call Alg. 6.1 with $(N, P, q_v, \varepsilon_v)$;
2 $\boldsymbol{R} \leftarrow$ Call Alg. 6.2 with $(\hat{N}, \hat{P}, q_e, \varepsilon_e)$;
3 $(\hat{V}, \hat{E}) \leftarrow$ Call Alg. 6.4 with $(\hat{P}, \boldsymbol{R})$;
4 if $u \in \hat{V}_{leaves}$ has closer index to $v_1$ then
5     $\hat{V} \leftarrow$ reverse the order of levels in $\hat{V}$;
6 return: $G_P \leftarrow (\hat{V}, \hat{E})$;
```

to pre-process a differentially private network, and then insert each vertex of the path into the graph G_P one by one by calling Algorithm 6.4 in Line 3 of Algorithm 6.3 to have an initial G_P for publishing. To recover the visit order of the original path with G_P, we check two groups of vertices: roots and leaves. The group, which contains the vertex with a closer index to the real first vertex in the path P, will serve as the root in the final output. That is, to fit Rule 5, we may turn G_P upside down to have the final output. This operation is implemented from Line 4 to Line 5 of Algorithm 6.3.

According to the sequential composition of DP [99], based on Lemma 6.1 and Lemma 6.2, we have Theorem 6.1 to study the overall privacy budget of the DP of Algorithm 6.3.

Theorem 6.1. *Algorithm 6.3 is $(\varepsilon_v + \varepsilon_e + \ln 2)$-differentially private.*

Lemma 6.3 and Proposition 6.1 study the probability of recovering the correct path P from the published graph-based path G_P.

Lemma 6.3. *There are at least two root nodes in the graph-based path G_P.*

Proof. Assume there is only one root v in a graph-based path, then according to Rule 1, u (neighbour of v in the path) and v should be placed into different branches (because edge $(u,v) \in E_P$). That is, u should also be a root of the graph-based path. Therefore, this lemma holds. □

Proposition 6.1. *Given a path $P = \{v_1, v_2, \dots, v_n\}$, the expected probablity of confirming the visit order of P is at least $1/2$ by our graph-based path G_P (without DP injection).*

Proof. According to Lemma 6.3, we assume there are $k \geq 4$ vertices that are roots and leaves in G_P. There is only one case where G_P cannot confirm the first node of P. That is, among the k root and leaf vertices, vertex v, whose index is the closest one to the first node of the path, and vertex u, whose index is the closest one to the last node of the path, are both roots. In such a case, for a genuine user that knows the two ends of the published path by Rule 1 and Rule 2, because both u and v are the closest nodes to one end of the path, based on Rule 5, the genuine user cannot distinguish which end is the first node of the path. The probability of such a case in all possible graphs G is $2/\binom{k}{2} = 4/k(k-1)$ because the root and leaves sets are exchangeable. Moreover, u should be closer to the last node than v to the first node. We assume it has $Pr \in [0,1]$ probability to have such case. Then G_P has a $1 - Pr \times 4/k(k-1)$ probability to confirm the real visit order of P, which is at least $2/3$. In addition, consider a special network with n vertices and $n-1$ edges. Give a path P from such network, because both the first node and last node will be the root in the graph-based path, we have $1/2$ probability to confirm the real first node. Therefore, we have probability $1/2 = \min\{1/2, 2/3\}$ of confirming the visit order of P such that this proposition holds. □

Algorithm 6.4 is used to produce the graph-based path G_P. Generally, in Algorithm 6.4, we maintain and track the status of each vertex during the insertion process to ensure the vertices are correctly placed in the graph-based path. Clearly, because we hide the exact information of the path in the topology of the graph-based path, the levels of G_P, where we place the vertices, play an important role. Thus, we prepare a possible levels set for each vertex against the present graph when placing the vertex into G_P (Line 5 to Line 8 in Algorithm 6.4). To ensure the first node of the path could be placed as the root in the graph-based path, we always use the maximum level from the set for inserting the vertex (Line 10 to Line 14 in Algorithm 6.4).

Algorithm 6.4: Insert Vertex.

```
Input  : A path $P$;
         A vertices relationship matrix $\boldsymbol{R}$.
Output : A vertex superset $\hat{V}$ and an edge set $\hat{E}$
Init: An empty stack $S$, $S_{bak} \leftarrow S$, $level.top \leftarrow 0$, $level.bottom \leftarrow 0$,
  $u \leftarrow P.dequeue()$, $u.level \leftarrow 0$;
$S.push(u)$;
$v \leftarrow P.pop()$;
while $P \neq \emptyset$ do
    if $v.visit = 0$ then
        for $i \leftarrow level.top - 1$ to $level.bottom + 1$ do
            if Call Alg. 6.5 with $(S, v, \boldsymbol{R}, i)$ then
                $v.levelRange \leftarrow v.levelRange + \{i\}$;
    if $v.levelRange \neq \emptyset$ then
        $v.level \leftarrow \max\{v.levelRange\}$;
        $level.top \leftarrow \min_{\forall x \in S}\{x.level, v.level\}$;
        $level.bottom \leftarrow \max_{\forall x \in S}\{x.level, v.level\}$;
        $v.visit \leftarrow 1$, $S.push(v)$;
        $v \leftarrow P.pop()$, $v.visit \leftarrow 0$;
        if $|S| > |S_{bak}|$ then
            $(S_{bak}, \boldsymbol{R}_{bak}, P_{bak}, v_{bak}) \leftarrow (S, \boldsymbol{R}, P, v)$;
    else
        $P.push(v)$;
        $v \leftarrow S.pop()$, $v.levelRange \leftarrow v.levelRange - \{v.level\}$, $v.level \leftarrow nil$;
        if $v = u$ then
            $(S, \boldsymbol{R}) \leftarrow$ Call Alg. 6.6 with $(S_{bak}, \boldsymbol{R}_{bak}, v_{bak})$;
            $P \leftarrow P_{bak}$, $v \leftarrow P.pop()$, $v.visit \leftarrow 0$;
for each $v \in S$ do
    if $r_{v,x} = 2, \forall x \in S$ then
        $\hat{E} \leftarrow e = (v, x)$;
    $V_{v.level} \leftarrow V_{v.level} + \{v\}$;
$\hat{E} \leftarrow$ Remove duplicate edges;
$\hat{V} \leftarrow \{V_i\}, \forall i \in [level.top, level.bottom]$;
return: $(\hat{V}, \hat{E})$;
```

To compute the possible levels set, we exclude all bad levels that introduce contradictions to the graph building rules of Algorithm 6.5. Claim 6.1 studies the way to navigate the bad levels when inserting a vertex v into G_P.

Claim 6.1. *The connectivity is transitive in a branch of G_P.*

Algorithm 6.5: Test Insertion Level.

```
Input  : A stack $S$;
         A vertex $u$;
         A vertices relationship matrix $\boldsymbol{R}$;
         A given level $n$.
Output : A boolean status.
if $r_{u's\ parent, u's\ child} = 1$ then
    return: False;
for each $v \in S$ do
    if $v.level = n$ AND $r_{u,v} = 2$ then
        return: False;
    if $v.level > n$ AND $r_{u,v} = 2$ AND $r_{u,v's\ child} = 1$ then
        return: False;
    if $v.level < n$ AND $r_{u,v} = 2$ AND $r_{u,v's\ parent} = 1$ then
        return: False;
return: True;
```

Claim 6.1 shows an important feature of our graph-based path: all the vertices in one branch should be connected to each other. That is, when inserting v into G_P, a bad level for v either breaks our Rule 2 for G_P or such a transitive relation in Claim 6.1 exists. The upper half of Figure 6.5 shows a counterexample to Claim 6.1. Given $u, x \in G_P$, $r_{u,x} = 2$, when inserting $v \in P$ to G_P where $r_{v,u} = r_{v,x} = 1$, there is no good level for v, if $\exists y \in G_P$ where $r_{y,v} = r_{y,u} = r_{y,x} = 2$ and y's level in G_P is between u's and x's. This case indicates that there may be bad placement(s) from the existing vertices that are already in the present graph. To overcome the bad placement(s), we record each inserted vertex in a stack. When we have a vertex with an empty possible levels set, the stack enables us to backtrack and re-insert the recently inserted vertices until we can successfully insert all the vertices into the graph. This backtracking operation is implemented from Line 18 to Line 19 of Algorithm 6.4. The lower half of Figure 6.5 illustrates an example addressing this issue.

Based on Claim 6.1, we have Theorem 6.2, which underpins the graph-based path.

Theorem 6.2. *A graph-based path exists unless $\exists u \in P$ such that u breaks the transitive relation of all possible topologies of G_P.*

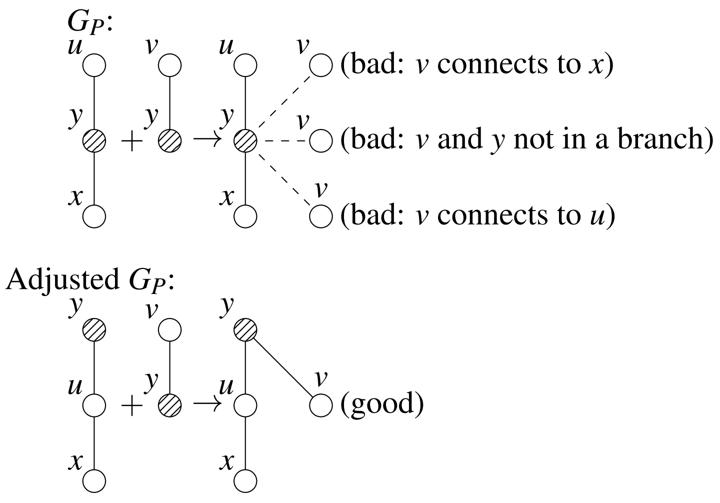


Fig. 6.5 Example of Claim 6.1.

According to Theorem 6.2, it is possible to have a network that cannot deliver a graph-based path G_P. Namely, to insert a vertex u to a G_P, after trying all the possible levels of each vertex in the present G_P and backtracking the stack in Algorithm 6.4, we cannot find a good level to insert u. The upper half of Figure 6.6 shows an example of a bad G_P by Theorem 6.2, where the present G_P is the only possible topology with the given vertices a, b, c and d. That is, a is only connected with c, b is connected with c and d. As we can see, there is no way to appropriately insert e, which is connected with a and d only, to the present G_P. Particularly, we cannot insert e to the level above a (or the level below d) because e will be connecting with c (or b) then. We cannot insert e to the same level as a and d because vertices in the same level are not connected, but we must keep e connected with a and d. We cannot insert e to the level between a and d because e will immediately connect a and d, which breaks the present connectivity in G_P where a and d are not connected.

The reason for the conflict when inserting a vertex is straightforward. When a vertex u has multiple connections in the network, according to Rule 1, Rule 2 and Claim 6.1, the topology of the graph-based path may not satisfy all the transitive relations relevant to u at the same time. To address this problem, we propose Algorithm 6.6 to eliminate such a conflict. The main idea of Algorithm 6.6 is splitting the vertices, which breaks the transitive

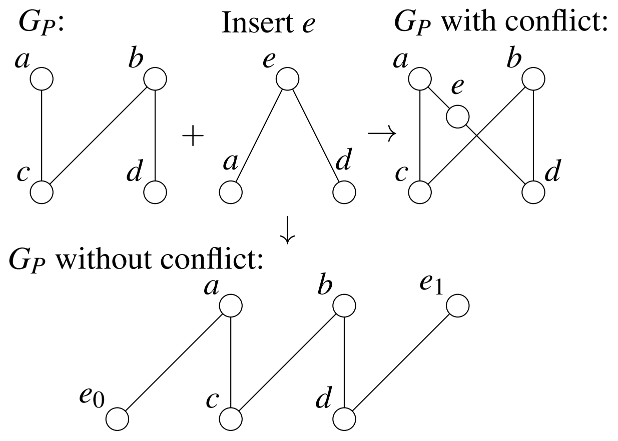


Fig. 6.6 Example of Resolving Conflicts in G_P.

relation in branches in the present G_P, depending on u and its degree of connectivity in the network. As a result, each piece has fewer connections from the original vertex. This way, it is easier to satisfy the connectivity of a conflicting vertex. In Algorithm 6.6, when splitting a vertex v, we have $v.connections = \bigcap_i v_i.connections$, where $i \le v.degree$. The lower half of Figure 6.6 shows that, after splitting the vertex e into e_0 and e_1, we resolve the conflict introduced by vertex e. The new G_P satisfies all the rules for a graph-based path.

Note that the splitting operation of Algorithm 6.6 is different to the differentially private vertices pre-processing of Algorithm 6.1, since, in Algorithm 6.1, the sub-vertices are duplicates of the base vertex, but have different connections according to the positions of the duplicates, that is, some of the sub-vertices are fake ones. However in Algorithm 6.6, after splitting the base vertex, each piece of the base vertex inherits partial connections from their base vertex, that is, those vertices are not fake ones. Theorem 6.3 proves that this splitting operation ensures we always have a graph-based path from any input network and path.

Theorem 6.3. *Algorithm 6.6 guarantees production of a graph that satisfies the rules for the graph-based path.*

Proof. When inserting a new vertex v_{new} to the graph-based path G_P by Algorithm 6.4, if we cannot find a level in the present graph-based path G_P to satisfy the connectivities of v_{new},

Algorithm 6.6: Resolve Inserting Conflicts.

```
Input  : A stack $S$;
         A vertices relationship matrix $\boldsymbol{R}$;
         A vertex $v_0$.
Output : A stack $S$;
         A vertices relationship matrix $\boldsymbol{R}$.
$S_{conflict} \leftarrow \emptyset$;
$level.top \leftarrow \min_{\forall u \in S}\{u.level\}$;
$level.bottom \leftarrow \max_{\forall u \in S}\{u.level\}$;
for each $u \in S$ do
    $S_{conflict}.push$(the conflicting $u$ for $v$);
    $S.remove(u)$;
if $S_{conflict} = \emptyset$ then
    $S_{conflict}.push(v_0)$;
for each $u \in S_{conflict}$ do
    $S_{temp} \leftarrow \emptyset$;
    for $l \in [level.top - 1, level.bottom + 1]$ do
        $S_l \leftarrow$ Insert the suitable $v \in S$ if $u.level = l$;
        $S_{temp}.push(S_l)$;
    $S_{temp} \leftarrow$ Solve a set cover for $u$ and $S_{temp}$;
    Split $u$ according to $S_{temp}$;
    $(S, \boldsymbol{R}) \leftarrow$ Insert the split $u$s to $S$ and $\boldsymbol{R}$;
if $v_0 \notin S$ then
    $S \leftarrow$ Insert $v$ to $S$ as Alg. 6.4;
return: $S$ and $\boldsymbol{R}$;
```

then we say there is a *conflict*. In the worst case, every time when inserting a vertex, we have a *conflict*. In such a case, we first apply Algorithm 6.6 to split each base vertex i in the network/graph into at most k_i pieces, where k_i is the degree of vertex i, then insert each piece of the base vertex i by Algorithm 6.4. Clearly, we will end up with a forest as the topology of the graph-based path, where the forest contains $|E_G|$ trees. Each of the trees contains only one edge of G. Therefore, this theorem holds. □

Finally, we study the depth of G_P in Proposition 6.2 that assists the proof of Corollary 6.1, where we show that it is efficient for genuine users to recover the original path from the graph-based path G_P.

Proposition 6.2. *The depth of the graph-based path is the length of the longest branch, and this is at most $|V_G|/2$.*

Proof. According to the way we build the graph-based path, the longest branch comes from the maximum-sized complete sub-graph in G. We construct G by removing the edges of the path from (and adding non-existent edges into) the original network; thus, in the worst case, there are at most $|V_G|/2$ vertices that could form a complete sub-graph in G. Therefore, this proposition holds. □

Corollary 6.1. *It takes $\mathscr{O}(|V_G|)$ time complexity for a genuine user to recover the original path from the graph-based path (G_P in Figure 6.3).*

Proof. For the genuine user, it takes $\mathscr{O}(|V_G|+|E_G|)$ time complexity to traverse all the edges in G to recover the path edges by Rule 1 and 2. In the worst case, $\mathscr{O}(|E_G|)$ is equivalent to $\mathscr{O}(|V_G|^2)$. However, according to Proposition 6.2, each vertex in G_P has at most $\mathscr{O}|V_G|$ edges. Thus, for the genuine user, it takes $\mathscr{O}(|V_G|+|V_G|)$ to search all edges in G_P. Therefore, it takes $\mathscr{O}(|V_G|)$ time complexity to confirm the path connections. □

6.4 Experimental Evaluation

In this section, we present the experimental evaluation of the proposed algorithm, including the dataset we used, experimental configurations and experimental results.

6.4.1 Dataset and Configurations

According to the inputs of our algorithm, we need a dataset that contains a network where all vertices form a path. To evaluate the performance of our algorithm over different sizes of graphs, similar to [42, 4], we generate and use a synthetic dataset. This is an bidirected/undirected network with different sizes and densities, which contains the path (either acyclic or cyclic) used for publishing. We produce the bidirected/undirected network with a given number of vertices $|V|$ and a given number of edges $|E|$ in three steps. First, we sample all the vertices from a two-dimensional plane uniformly. Second, we draw a path for publishing to connect all the vertices in the network. Third, all the edges not in the path yet in the network are randomly generated with a uniform distribution.

In the experiments, because the graph-based path can definitely recognise the vertices and edges of the published path, we measure the utility of the produced graph-based path by whether the visit order of the original path can be restored or not. We name the output confirming the correct visit order as "good output" in the figures for experimental results. In addition, because we prove that some network topology may not produce a graph-based path, in the experiments, we also show the percentage of "usable maps", which can produce a graph-based path.

Given that our differentially private algorithm is randomised, according to the law of large numbers [41], we report the expected algorithm performance of each aspect by calculating the average results of running the algorithm $1,000$ times. To simulate the real-world scenario, the size of a given network/path in our experiments is no more than 10 vertices. Accordingly, the number of edges of the synthetic network is in the range from $|V|-1$ (a network only has a Hamilton path for publishing) to $\frac{1}{2}|V|(|V|-1)$ (for the complete network). Thus, in our experiments, we have overall 122 different sizes of networks as inputs.

The experiments are implemented in Python 3.7, plotted with MATLAB R2019b, on a server, with 16 cores and 64 GB memory offered through the Phoenix HPC service at the University of Adelaide. (Source code: https://github.com/suluz/dp_path)

6.4.2 Experimental Results

In this section, we evaluate the performance of our algorithm from the following aspects:

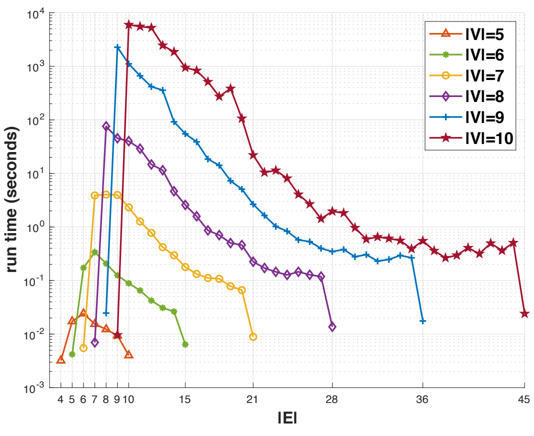
- the run time of the algorithm with DP on either vertices (Algorithms 6.1) or edges (Algorithms 6.2),
- the output quality of the random networks without DP (Algorithms 6.1, 6.2) and the splitting strategy (Algorithm 6.6) for the graph-based path,
- the quality of the differentially private output (Algorithms 6.1, 6.2).

Given that our graph-based path can definitely confirm whether an edge belongs to the original path, we evaluate the output (graph-based path) quality by testing whether the output can confirm the visit order of the original path, as shown in Section 6.3. In the figures in this section, each point on a curve represents a specific network with a given number of vertices and edges.

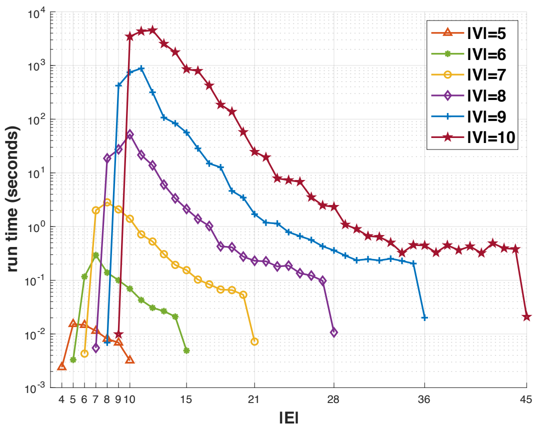
Run Time

Figure 6.7 depicts the run time of our algorithm with acyclic paths over different sizes of networks and different settings for DP on either vertices or edges. It is clear that, when increasing the number of vertices, the run time of the algorithm increases significantly. However, from Figure 6.7c and Figure 6.7d, the increasing number of edges results in a decreasing run time. The reason for this is that, with a given number of vertices and a given ε_e, when increasing the number of edges, fewer spaces are available to add fake edges; hence, the number of possible topologies of the network decreases, so it will take less time for the algorithm to find a suitable graph-based path.

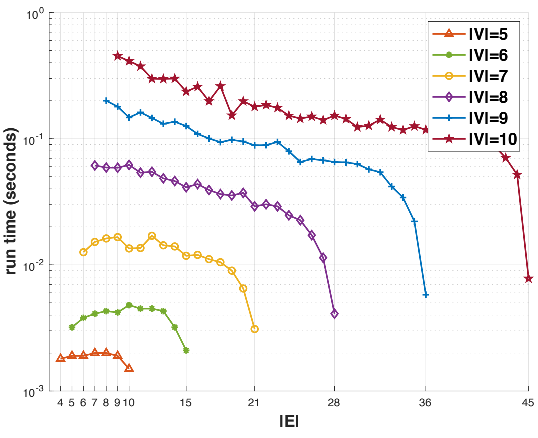
According to Equation (6.1), more vertices are injected with a smaller privacy budget on vertices ε_v, so that the run time in Figure 6.7a ($\varepsilon_v = 0.5$) is much higher than the run time in Figure 6.7b ($\varepsilon_v = 1$). Further, based on Equation (6.4), more edges are injected to the network (G in Figure 6.3) with a greater privacy budget on edges ε_e; hence, the run time in Figure 6.7c ($\varepsilon_e = 0.5$) is much higher than the run time in Figure 6.7d ($\varepsilon_e = 1$).



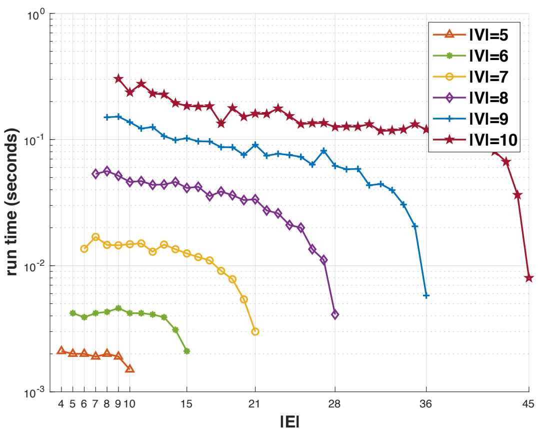
(a) DP on vertex ($\varepsilon_v = 0.5$)



(b) DP on vertex ($\varepsilon_v = 1$)



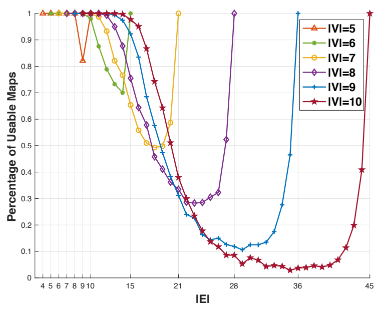
(c) DP on edge ($\varepsilon_e = 0.5$)



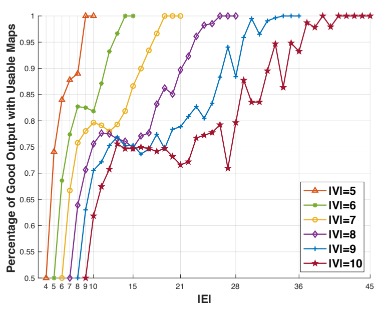
(d) DP on edge ($\varepsilon_e = 1$)

Fig. 6.7 Run Time (Acyclic Path).

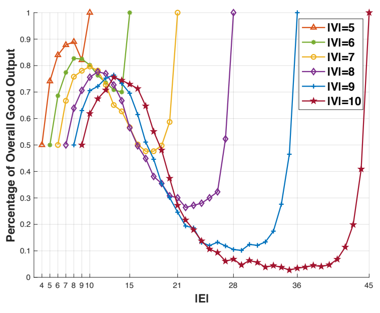
Outputs Quality over Random Networks without DP



(a) Percentage of Usable Networks



(b) Percentage of Good Output with Usable Networks



(c) Percentage of Overall Good Outputs

Fig. 6.8 Output Quality without DP and a Splitting Strategy.

In Figure 6.8, we study the performance of our graph-based path without both DP injection and a splitting strategy (as in Algorithm 6.6) for different sizes of graphs with an acyclic path.

Figure 6.8a shows the percentage of usable networks for successfully producing the graph-based path without a splitting strategy. In particular, about 43% (52/122) of the networks cannot produce the graph-based path with over 50% possibility, especially for networks (paths) with more than seven vertices. Moreover, only about 30% (30/122) of networks produce a graph-based path with 100% guarantee. Such networks are the complete graph and the sparse graphs $G = (V, E)$ where $|E| \in [|V| - 1, |V| + 3]$.

Figure 6.8b depicts the percentage of good outputs, which can confirm the correct visit order of the original path from the usable networks. According to how we recover the exact path from the graph-based path (Section 6.3), in our experiments, we count the good outputs as follows.

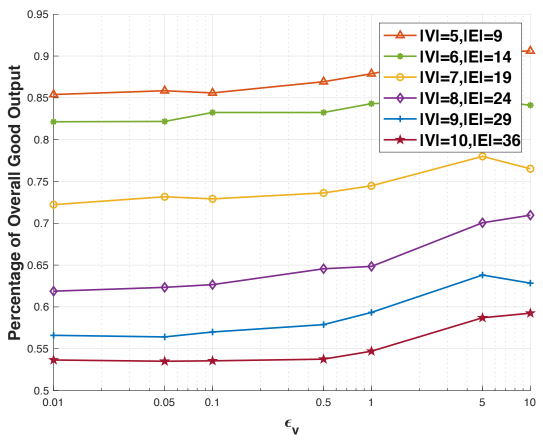
- If we confirm the first node of the original path directly, we count it as a good output.
- If the two ends of the original path have equal chance of being confirmed as the first node, we add 0.5 to the total number of good outputs.

Consequently, we have consistent results for both Figure 6.8b and Proposition 6.1. That is, with at least 50% probability, our graph-based path confirms the visit order of the original path. This is the exact original path, if we have full information about the vertices and the existence of the edges of a network. In addition, Figure 6.8b shows that, with a given number of vertices, the number of edges and the percentage of good outputs are roughly positively correlated. The reason is that, on average, more edges in the network mean more vertices are linked to the first node of the path (Line 3 and Line 10 of Algorithm 6.4); hence, we have a higher probability of keeping the first node as a root to have a good output. The result illustrated in Figure 6.8b provides a base result for analysing the experimental results in the next section.

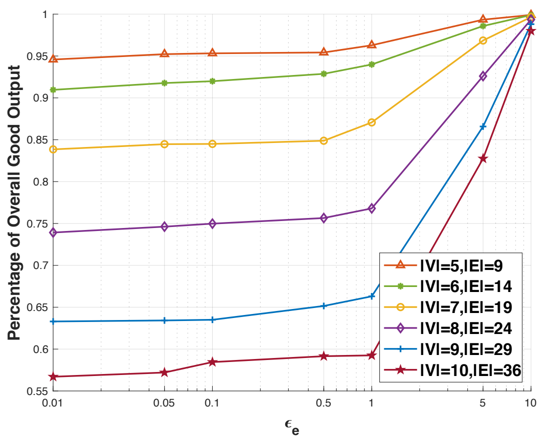
Finally, Figure 6.8c illustrates the Hadamard product [64] of Figure 6.8a and Figure 6.8b, namely, the overall good output percentage without a splitting strategy and DP injection.

Quality of Differentially Private Outputs

In this section, we report the output quality of our algorithm (with consideration of DP injection and the splitting strategy) on both acyclic paths (in Figure 6.9) and cyclic paths (in Figure 6.10). With the splitting strategy of Algorithm 6.6, we always produce a graph-based path from any network with 100% guarantee in the experiments, so we do not plot a figure to present it.



(a) DP on Vertices



(b) DP on Edges

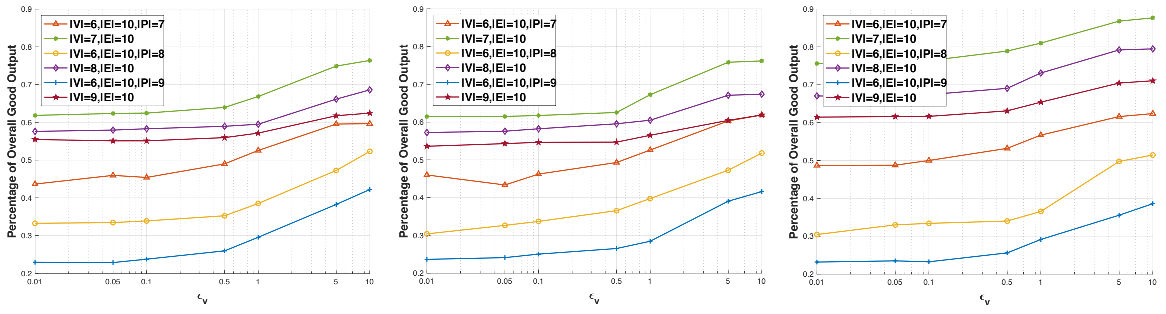
Fig. 6.9 Output Quality with DP on Acyclic Paths.

To evaluate the performance of our algorithm on acyclic paths in the worst case, we select the number of vertices and number of edges to form a graph, from which we observe a minimum number of usable networks in the experiments of Figure 6.8a. We use this worst-case graph configuration for the experiments of Figure 6.9.

In Figure 6.9a, we only inject DP on the vertex to learn the effect of the differentially private vertices pre-processing (Algorithm 6.1). Based on Equation (6.1) and Figure 6.8b, with an increasing ε_v, fewer fake vertices are added; hence, we have an increasing percentage of good outputs, which fits the results in Figure 6.9a.

Figure 6.9b shows the performance of differentially private edge pre-processing (Algorithm 6.2). Given a greater ε_e introduces more edges to the network (G in Figure 6.3), then, according to the result of Figure 6.8b, we expect an increasing percentage of good output, as in Figure 6.9b. In particular, when we have a very large ε_e, say 10 in Figure 6.9b, the network will be almost a complete graph, based on the differentially private edge pre-processing. According to the analysis of Figure 6.8b, the utility of the graph-based path will approach its maximum value when being a complete graph. Therefore, in Figure 6.9b, all the curves converge to the maximum utility.

Comparing the results of Figure 6.9a and Figure 6.9b, as injecting DP on vertices introduces fake vertices, it decreases the probability of placing the first node of a path as a root. This affects the performance of the graph-based path more significantly than DP injection on edges. Therefore, for acyclic path publishing, to protect private edges, only injecting DP on edges is required to meet both privacy and utility requirements.



(a) $\varepsilon_e = 0.5$ (b) $\varepsilon_e = 1$ (c) $\varepsilon_e = 5$

Fig. 6.10 Output Quality with DP on Cyclic Paths.

When we have a cyclic path in a given network, to preserve the privacy of the path (the existence of a ring and the existence of an arbitrary edge), DP on vertices and edges is always needed. Given that we have already studied the performance of DP on vertices in Figure 6.9a, we inject DP on both vertices and edges in Figure 6.10. Specifically, according to Figure 6.9b, we tune ε_v with three fixed ε_es: $\varepsilon_e = 0.5$ in Figure 6.10a, $\varepsilon_e = 1$ in Figure 6.10b, and $\varepsilon_e = 5$ in Figure 6.10c.

In Figure 6.10, we select the network with 6 vertices and 10 edges (because it achieves a local minimum for the percentage of good outputs) as a base network for cyclic path publishing. We compare the performance between a cyclic path and its corresponding acyclic path with the same number of vertices. With a given ε_v and a given ε_e, our algorithm achieves better performance on acyclic paths than cyclic paths, as the vertices in a cyclic path have more than one copy, according to Equation (6.1); hence, the differentially private vertex pre-processing would create more vertices for cyclic paths than acyclic paths. Then, based on the result of Figure 6.8b, more vertices damage the performance of the graph-based path; hence, the results in Figure 6.10 are as expected. Additionally, for the same reason behind Figure 6.9, when increasing ε_v and ε_e, the percentage of good output is increased, as shown in Figure 6.10.

6.5 Summary

In this chapter, we have proposed a differentially private graph-based path publishing algorithm. In this algorithm, we hide the real path in the topology of the original network by embedding it in a graph containing fake edges and vertices and applying DP techniques, so that only trusted users who have full knowledge of the existence of the edges in a network (which contains the target path) can reconstruct the exact vertices and edges of the original path, and hence determine the visit order of the vertices on the path with high probability. Our algorithm effectively protects edge existence, which is not supported with existing approaches for differentially private path publishing, while preserving the existence of vertices. Both theoretical analyses and experimental evaluation show that our algorithm guarantees both utility and privacy against adversaries who miss partial edge information, but have information on all vertices in the path.

Chapter 7

Differentially Private *k*-Means Clustering with Convergence Guarantees

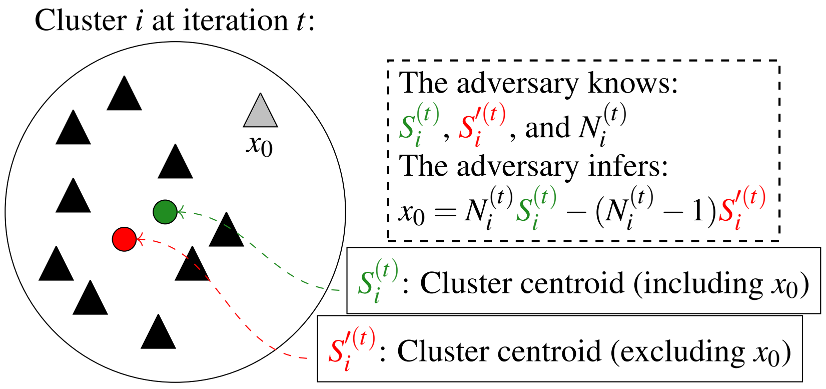
Iterative clustering around representative points is an effective technique for clustering that helps us gain insights about data to support various important applications. Unfortunately, it also provides security holes that may allow adversaries to infer the private information of individuals with some background knowledge. To protect individual privacy against such inference attacks, preserving DP for iterative clustering algorithms has been extensively studied. Existing differentially private clustering algorithms adopt the same framework to compute differentially private centroids iteratively by running Lloyd's *k*-means algorithm to obtain the real centroids, and then perturbing them with a DP mechanism. These algorithms suffer from non-convergence problems, that is, they provide no guarantees that they will terminate at a solution of Lloyd's algorithm within a bounded number of iterations. This problem severely affects their clustering quality and execution efficiency.

To address this problem, this chapter follows the same centroid updating pattern as existing work in interactive settings; however, we propose a novel framework for injecting DP into the real centroids. Specifically, to ensure convergence, we maintain the perturbed centroids of the previous clustering at the iteration $t-1$ to compute a *convergence zone* for each cluster in the current iteration t, where we inject DP noise. To have a satisfactory convergence rate, we further control the orientation of centroid movement in each cluster using two strategies: one takes the orientation of centroid movement from iteration $t-1$ to iteration t (past knowledge), and the other uses the additional information of the orientation from iteration t to iteration $t+1$ (future knowledge). We prove that, the expected number of iterations of our algorithm (in both strategies) converging to a solution of Lloyd's algorithm in at most twice as many iterations as Lloyd's algorithm. Further, when using both past and future knowledge, we prove that our algorithm converges to the same solution as Lloyd's algorithm (for the same initial centroids) with high probability, at the cost of a slower convergence speed compared with using only past knowledge, as a result of duplicated operations in each iteration required to compute the future knowledge. We perform experimental evaluations on six widely used real-world datasets. The experimental results show that our algorithm outperforms the state-of-the-art methods for interactive differentially private clustering with guaranteed convergence and enhanced clustering quality, while meeting the same DP requirements.

7.1 Introduction

In the era of big data analytics and rapid development of deep learning and its impressive achievements, such as the Google artificial intelligence Go player, AlphaGo, beating the best human Go player by being self-taught using deep neural networks [142, 162], traditional machine learning techniques, such as k-means clustering, are increasingly important for learning insights from small data without a ground truth. They also offer high running efficiency and prediction accuracy [125, 112]. In this chapter, we address the issue of effective privacy preservation when realising the popular Lloyd's k-means clustering algorithm [87].

Despite the benefits enjoyed from clustering, the privacy disclosure risk can thwart people's willingness to contribute data (especially data that may link to privacy) to clustering algorithms. Consider the following inference attack based on the difference between the outputs from a private dataset and an adversary's background knowledge. We consider a trusted data curator that manages a dataset X and an adversary that owns a dataset X'. In the worst case, X and X' differ by a single data element. At any arbitrary iteration t during clustering, assume a set of centroids in X is accidentally disclosed to the adversary. By comparing the difference between the set of centroids generated by X and X', the adversary can easily infer the value of the missing item x_0, and thus technically gain full knowledge of the dataset X. Figure 7.1 depicts how such an inference attack works, where $n_i^{(t)}$ is the overall number of items in cluster i at iteration t of X.



Cluster i at iteration t:
x_0
The adversary knows:
$S_i^{(t)}$, $S_i'^{(t)}$, and $N_i^{(t)}$
The adversary infers:
$x_0 = N_i^{(t)} S_i^{(t)} - (N_i^{(t)} - 1) S_i'^{(t)}$
$S_i^{(t)}$: Cluster centroid (including x_0)
$S_i'^{(t)}$: Cluster centroid (excluding x_0)

Fig. 7.1 Illustrated Example of An Inference Attack.

From the above inference attack example, it is clear that preserving the privacy of individual items in a dataset when running an iterative clustering algorithm must protect the true value of the centroids of the clusters at each iteration. Unfortunately, some of the well-known privacy-preserving paradigms, such as SMC or MPC [174] and anonymity [150, 95, 80], are vulnerable to such an inference attack because both the SMC paradigm and the family of anonymity are vulnerable to adversaries that have the maximum background knowledge (e.g., $n-1$ out of n items in the dataset).

To preserve privacy against inference attacks with maximum background knowledge (e.g., Figure 7.1), DP [33] has been applied in Lloyd's algorithm in interactive settings [148], whereby random DP noise is injected into each iteration when running Lloyd's algorithm. Numerous studies [12, 34, 104, 176, 148, 118] guarantee DP while achieving acceptable clustering quality in interactive settings through three DP mechanisms: the sample and aggregation framework of DP [114], the exponential mechanism of DP (ExpDP) [100] and the Laplace mechanism of DP (LapDP) [37]. We identify two weaknesses with existing work [12, 34, 104, 176, 148, 118]. The work in [104] is based on a sample and aggregation framework; however, it shows unsatisfactory clustering quality because its uniform sampling may result in skewness over the sampled buckets; hence, the aggregated centroids would have a significant distance to the Lloyd's result. Studies such as [12, 34, 176, 148, 118] apply ExpDP and LapDP; however, they suffer from a non-convergence problem, as unbounded noises are injected to an arbitrary direction. This problem severely affects the clustering quality (as the clustering result would have an unknown distance to the true centroids) and the efficiency of the algorithm (as it requires a large computational cost to determine predefined parameters). The importance of convergence guarantees (to a local optimum of Lloyd's *k*-means) for differentially private *k*-means clustering is two-fold. First, without convergence guarantees, a predefined iteration number is required to terminate a differentially private *k*-means algorithm. To find such an iteration number to satisfy the clustering quality with a given input dataset, we must run the algorithm over the dataset multiple times. Further, running the algorithm on different datasets requires recalculation of the iteration number through the above process repeatedly, which results in a large computational cost. Second, because a non-convergent result may have an unbounded large distance to the local optima of the *k*-means problem, existing work such as [12, 34, 176, 148, 118] on differentially private *k*-means clustering provides no clustering quality guarantees.

To overcome the above weaknesses, we propose a new differentially private *k*-means clustering algorithm in interactive settings that improves existing work by offering guaranteed convergence and enhanced clustering quality with the same DP requirements. In summary, our main contributions are as follows.

- We propose a novel approach for differentially private clustering that injects bounded DP noise into each iteration of the clustering process by applying ExpDP in a controlled orientation to preserve data privacy against inference attacks. In comparison with existing work that injects unbounded noise in arbitrary directions, our approach ensures convergence in at most double the number of iterations as with Lloyd's k-means clustering.
- We analyse key properties, including convergence, convergence rate and bound of DP, for the differentially private k-means clustering algorithm using two centroid updating strategies based on past knowledge of previous iteration centroid movement (using the same assumptions as existing work), and past and future knowledge, that is, centroid movements based on the previous and next iterations. The former requires fewer iterations for convergence, while the latter results in enhanced convergence quality.
- We experimentally evaluate the performance of the clustering quality with various experimental settings on six widely used real-world datasets. With the same DP guarantees, our algorithm for the centroid updating strategies achieves better clustering quality (utility) than other state-of-the-art differentially private k-means clustering algorithms.

To the best of our knowledge, our algorithm is the first that ensures convergence for differentially private k-means clustering in interactive settings.

The remainder of this chapter is organised as follows. In Section 7.2, we provide a brief introduction to the preliminaries of this chapter, including Lloyd's algorithm and definitions related to convergence. In Section 7.3, we introduce our approach to ensure convergence through noise injection in controlled centroid movement orientation, and introduce preliminary analysis on the convergence property. In Section 7.4, we propose two designs for noise sampling zones during each iteration of clustering. In Section 7.5, we describe our differentially private k-means clustering algorithm and its convergence and DP proof. In

Section 7.6, we provide the experimental evaluation and compare the clustering quality (data utility) of existing work and our algorithm. Finally, we conclude this chapter in Section 7.7.

7.2 Preliminaries - Lloyd's *k*-Means Algorithm

In this section, we briefly introduce Lloyd's *k*-means clustering algorithm [87]. Following the same pattern as existing differentially private *k*-means algorithms, we consider DP noise injected into real centroids computed by Lloyd's algorithm over several iterations. We also provide our definitions related to convergence.

The *k*-means clustering aims to split a dataset with N items into k clusters, where each item is allocated into a cluster with the nearest cluster centroid to itself. The formal cost function of *k*-means clustering is:

$$\underset{\mathbf{C}}{\operatorname{argmin}} J = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - S_i||^2, \tag{7.1}$$

where $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ is the set of k clusters, x is an item in the dataset $X = \{x_1, x_2, \ldots, x_N\}$ and S_i is the centroid of C_i. Equation 7.1 calculates the total cost of a set of centroids.

The most well-known *k*-means clustering algorithm is an iterative refinement algorithm called Lloyd's *k*-means clustering [87]. Lloyd's algorithm improves the quality of centroids by iteratively running a *re-assignment* step and a *re-centroid* step. In the *re-assignment* step, each item is assigned to its nearest centroid to build the k clusters. In the *re-centroid* step, the algorithm re-calculates the centroid (mean) for each cluster. For simplicity, in this chapter, the centroid of a cluster and the mean of a cluster are interchangeable concepts. The new/updated k centroids are used for the next *re-assignment* step. Lloyd's algorithm terminates when the k centroids remain the same after two neighbouring iterations. Lloyd's algorithm is guaranteed to converge to a local optimal solution of the *k*-means problem within a finite number of iterations.

To measure the quality of convergence, in this chapter, we define *convergence* and *convergent degree* for a differentially private k-means clustering algorithm.

Definition 7.1 (Convergence)**.** *Given a dataset $\mathscr{D}$, an integer k, Lloyd's algorithm $\mathscr{L}$, and the set of local optimal solutions of the k-means problem $\mathscr{C}$, we have $\mathscr{L}(\mathscr{D}) \to \mathscr{C}$. We say a differentially private k-means algorithm, $\mathscr{F}$, is convergent, if and only if, $\mathscr{F}(\mathscr{D}) \to \mathscr{C}$.*

Definition 7.2 (Convergence Degree)**.** *Given a set of initial centroids $d \in \mathscr{D}$, $\mathscr{L}(d) \to c \in \mathscr{C}$, the convergence degree of $\mathscr{F}$ is the probability* $\Pr[\mathscr{F}(d) \to c]$.

In addition, Table 7.1 lists the notations used in this chapter.

Table 7.1 Summary of Notations.

Notation	Description
$\hat{\cdot}$	Corresponding notation ($\cdot$ from Lloyd's algorithm) in privacy-preserving algorithms
$a_i^{(t)}$	Distance between $S_i^{(t)}$ and $\hat{S}_i^{(t)}$
$b_i^{(t)}$	Distance between $S_i^{(t)}$ and $S_i^{(t+1)}$
$C_i^{(t)}$	Cluster i at iteration t
Δ	Value difference of the cost function between two iterations
$\varepsilon_i^{(t)}$	DP budget for $C_i^{(t)}$
I	Overall iterations of Lloyd's algorithm
q	Quality function from DP
$J^{(S_i^{(t)})}$	Value of the cost function for $C_i^{(t)}$ with centroid $S_i^{(t)}$
$S_i^{(t)}$	Cluster centroid in $C_i^{(t)}$
$X_i^{(t)}$	Centroid updating orientation controller in $C_i^{(t)}$

7.3 Controlled Centroid Updating Orientation for Noise Injection

In this section, we first provide an overview of our approach, and then preliminarily analyse the convergence property for randomised centroid updating for k-means clustering.

7.3.1 Approach Overview

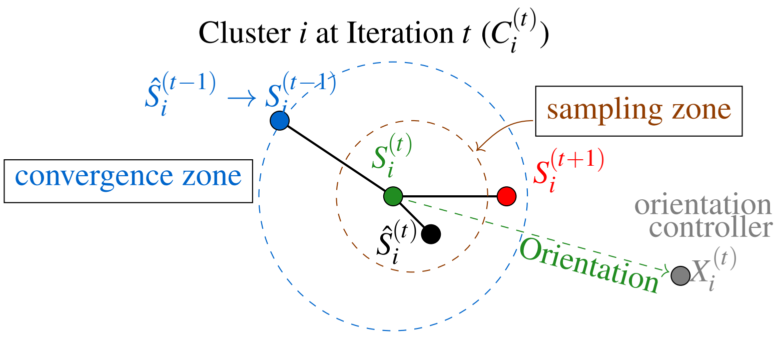


Fig. 7.2 Overview of Orientation Control.

Intuitively, if we can decrease the value of the cost function (Equation 7.1) iteratively, a k-means clustering algorithm would finally converge. In this chapter, we leverage this convergent property of Lloyd's algorithm to bound DP noise injection. The main idea of our approach is that we bound/control DP noise in a selected area for each iteration during the clustering process to guarantee the convergence. Given that there are many possible centroid movement directions in the selected area, to ensure the clustering quality, we apply ExpDP in a controlled orientation for centroid updating to approximately approach the direction of convergence of Lloyd's algorithm. Our idea differs from the existing work, where the updated cluster centroids are arbitrarily produced by a DP mechanism. Figure 7.2 illustrates the overview of our approach. In general, we have four steps to update a set of differentially private centroids at each iteration t.

1. Run Lloyd's algorithm using the past iteration $t-1$ centroid $S_i^{(t-1)}$ for a real centroid $S_i^{(t)}$ for each cluster i. Note that this $S_i^{(t-1)}$ is the differentially private centroids $\hat{S}_i^{(t-1)}$.
2. Compute a *convergence zone* by $S_i^{(t)}$ and $S_i^{(t-1)}$ for each cluster i.
3. Generate a *sampling zone* in the *convergence zone* by $S_i^{(t)}$ and *orientation controller* $X_i^{(t)}$ for each cluster i.
4. Sample a differentially private centroid $\hat{S}_i^{(t)}$ in this *sampling zone* with ExpDP.

We define a *convergent zone* (for convergence guarantee) and the corresponding *sampling zone* for centroid updating formally in Definition 7.3. The specific requirement for the *convergent zone* comes from Lemma 7.1 in the next section.

Definition 7.3 (Convergent & Sampling Zones). *In $C_i^{(t)}$, a convergent zone is a set of nodes, given by Converge Zone = $\{Node\ S:\ ||S - S_i^{(t)}|| < ||S_i^{(t-1)} - S_i^{(t)}||\}$, where $S_i^{(t)}$ is the mean/centroid of $C_i^{(t)}$. A sampling zone is a subset of the convergent zone.*

Definition 7.4 (Orientation Controller). *In $C_i^{(t)}$, an orientation controller is node $X_i^{(t)}$ such that the differentially private centroids $\hat{S}_i^{(t)}$ is randomly sampled by ExpDP according to the orientation $S_i^{(t)} \leftarrow X_i^{(t)}$.*

The challenge in our scheme is to fill the gap and design a suitable *sampling zone* and *orientation controller* in interactive settings that guarantees convergence and achieves enhanced clustering quality while meeting the same DP requirements as existing work. In the following sections, we propose two types of *sampling zone* (depending on whether we have knowledge of future centroid movement [90] or not) for our differentially privacy clustering algorithm.

7.3.2 Preliminary Analysis of Convergence Properties

In this section, we present a preliminary analysis that provides the foundations for our algorithms in the next section. In general, the following properties of the proposed algorithms are considered:

- the convergence of the proposed algorithms;
- the rate of convergence of the proposed algorithms compared with Lloyd's algorithm;
- the trade-off between utility and privacy of the proposed algorithms.

We first study the convergence of a randomised iterative clustering algorithm in Lemma 7.1.

Lemma 7.1. *A randomised iterative clustering algorithm is convergent if, in $C_i^{(t)}$, the sampled $\hat{S}_i^{(t)}$ satisfies $||\hat{S}_i^{(t)} - S_i^{(t)}|| < ||\hat{S}_i^{(t)} - S_i^{(t-1)}||$ in Euclidean distance, $\forall$ t, i.*

Proof. In Lloyd's algorithm, after the *re-assignment* step, prior to the *re-centroid* step, we build $C_i^{(t)}$ and have $J^{(S_i^{(t-1)})} = \sum_{x \in C_i^{(t)}} ||x - S_i^{(t-1)}||^2$, where $S_i^{(t-1)}$ is the mean/centroid of $C_i^{(t-1)}$ that is used in the *re-assignment* step to generate $C_i^{(t)}$. Similarly, after the *re-centroid* step, when the members of $C_i^{(t)}$ do not change, we have $J^{(S_i^{(t)})} = \sum_{x \in C_i^{(t)}} ||x - S_i^{(t)}||^2$.

Assuming Euclidean distance between $S_i^{(t-1)}$ and $S_i^{(t)}$ is $a_i^{(t)} = ||S_i^{(t-1)} - S_i^{(t)}||$, each record x_i in the dataset X has d dimensions, we have:

$$J^{(S_i^{(t-1)})} = \sum_{j=1}^{||C_i^{(t)}||} ||x_j - S_i^{(t-1)}||^2 = \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left(x_{jp} - S_{ip}^{(t-1)}\right)^2, \tag{7.2}$$

$$J^{(S_i^{(t)})} = \sum_{j=1}^{||C_i^{(t)}||} ||x_j - S_i^{(t)}||^2 = \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left(x_{jp} - S_{ip}^{(t)}\right)^2. \tag{7.3}$$

The distance $a_i^{(t)}$ between $S_i^{(t-1)}$ and $S_i^{(t)}$ can be further split to $(a_i^{(t)})^2 = \sum_{p=1}^{d} (a_{ip}^{(t)})^2$, where $a_{ip}^{(t)} = S_{ip}^{(t-1)} - S_{ip}^{(t)}$. Note that $a_i^{(t)} > 0$, $a_{ip}^{(t)}$ can be any real number. Then

$$\begin{aligned}
J^{(S_i^{(t-1)})} - J^{(S_i^{(t)})} &= \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left[(x_{jp} - S_{ip}^{(t-1)})^2 - (x_{jp} - S_{ip}^{(t)})^2\right] \\
&= \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left[(S_{ip}^{(t-1)} - S_{ip}^{(t)})(S_{ip}^{(t-1)} + S_{ip}^{(t)} - 2x_{jp}\right] \\
&= \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left\{a_{ip}^{(t)}[a_{ip}^{(t)} - 2(x_{jp} - S_{ip}^{(t)})]\right\} \\
&= \sum_{j=1}^{||C_i^{(t)}||} \sum_{p=1}^{d} \left(a_{ip}^{(t)}\right)^2 - 2\sum_{p=1}^{d} \left[a_{ip}^{(t)} \times \sum_{j=1}^{||C_i^{(t)}||} (x_{jp} - S_{ip}^{(t)})\right] \\
&= ||C_i^{(t)}|| \times (a_i^{(t)})^2 - 2\sum_{p=1}^{d} \left(a_{ip}^{(t)} \times 0\right) \\
&= ||C_i^{(t)}|| \times (a_i^{(t)})^2
\end{aligned} \tag{7.4}$$

where $||C_i^{(t)}||$ is the number of items in $C_i^{(t)}$. Note that, in Lloyd's algorithm, $J^{(S_i^{(t)})}$ is the minimum cost in $C_i^{(t)}$. If we pick a random node $\hat{S}_i^{(t)}$ from $C_i^{(t)}$ as the centroid for $C_i^{(t)}$ that satisfies $||\hat{S}_i^{(t)} - S_i^{(t)}|| = \delta_i^{(t)} a_i^{(t)} < ||S_i^{(t-1)} - S_i^{(t)}|| = a_i^{(t)}$ $(0 < \delta_i^{(t)} < 1)$, then we have $J^{(S_i^{(t-1)})} - J^{(\hat{S}_i^{(t)})} = ||C_i^{(t)}|| \times (1 - (\delta_i^{(t)})^2) \times (a_i^{(t)})^2 > 0$.

Thus, by updating the centroids to this set $\hat{S}^{(t)} = \{\hat{S}_1^{(t)}, \hat{S}_2^{(t)}, \ldots, \hat{S}_k^{(t)}\}$ (rather than the mean/centroid of clusters, $S^{(t)}$), the value of every item $\sum_{x \in C_i} ||x - S_i||^2$ can be further decreased, which results in a decrease of the cost function (Equation 7.1).

In addition, as we have a finite set of clustering solutions (at most k^N), and since we decrease the cost in each iteration with a randomised iterative algorithm, the algorithm satisfies the properties from the above proof and hence must converge (not approach) to a fixed value of the cost function. $\square$

Next, we consider the convergence and the convergence rate for a special case of $\hat{S}_i^{(t)}$ in Lemma 7.2 and Lemma 7.3. This special $\hat{S}_i^{(t)}$ (shown in Figure 7.3) is in the line segment of $\overline{S^{(t-1)}S^{(t)}}$, where $||\hat{S}_i^{(t)} - S_i^{(t)}|| = \delta_i^{(t)} \times ||S_i^{(t-1)} - S_i^{(t)}||$, $\delta_i^{(t)} < 1$. Lemma 7.2 and Lemma 7.3 assist us to prove the properties of our proposed algorithms in the following sections.

Lemma 7.2. *Given an algorithm ALG, if we randomly select an $\hat{S}_i^{(t)}$ in the line segment of $\overline{S^{(t-1)}S^{(t)}}$ in $C_i^{(t)}$, the degree of convergence of ALG is one.*

Proof. We know that the k-means clustering problem has a set of local optimal solutions, $\mathbb{S} = \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$, where $\mathbf{S}_i$ is one local optimum (the one to which Lloyd's algorithm converges to) that contains k centroids of the clusters, $\mathbf{S}_i = \{S_{i,1}, S_{i,2}, \cdots, S_{i,k}\}$. According to Lemma 7.1, assume ALG is convergent to $\hat{\mathbf{S}} = \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_k\} \notin \mathbb{S}$, then we must have room to further reduce the cost by either a *re-assignment* or a *re-centroid*. Therefore, $\hat{\mathbf{S}}$ is not the set of centroids that makes ALG convergent, unless $\hat{\mathbf{S}} \in \mathbb{S}$. Thus, ALG is convergent to, at least, one local optimum of the k-means clustering problem.

We say that a set of k nodes (each cluster contributes one node) belongs to a local optimum, $\mathbf{S}_i$, if Lloyd's algorithm converges to $\mathbf{S}_i$ by taking such a set of nodes as the initial set of centroids. Given that the two ends of the line segment $\overline{\mathbf{S}^{(t-1)}\mathbf{S}^{(t)}}$ belong to the same local optimum, then it is guaranteed that $\mathbf{S}^{(t-1)}$, $\mathbf{S}^{(t)}$, and $\hat{\mathbf{S}}^{(t)}$ always belong to the same local optimum for all iterations. Therefore, this lemma holds. $\square$

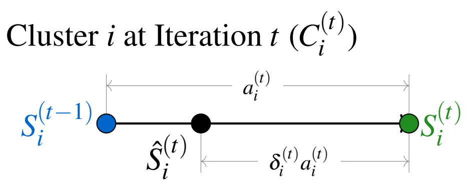

Cluster i at Iteration t ($C_i^{(t)}$)
$a_i^{(t)}$
$S_i^{(t-1)}$
$\hat{S}_i^{(t)}$
$\delta_i^{(t)}a_i^{(t)}$
$S_i^{(t)}$

Fig. 7.3 Help Figure for Lemma 7.2 and 7.3.

Lemma 7.3. *The expected number of iterations of algorithm ALG (described in Lemma 7.2) has at most $\frac{1}{1-\delta^2}$ the number of iterations of Lloyd's algorithm, where δ is the expected value of $\delta_i^{(t)}$, $\delta \in (0,1)$.*

Proof. Based on Lemma 7.2, the overall value difference of Equation 7.1 from the first iteration to the last iteration ($J = \sum_{i=1}^{k} J^{S_i^{(0)}} - \sum_{i=1}^{k} J^{S_i^{(I)}}$) is the same in both ALG and Lloyd's algorithm, where I is the total number of iterations. In each iteration, the cost is decreased by two steps: *re-assignment* and *re-centroid.* Without loss of generality, we have $J = \sum_{t=1}^{I}(\Delta_t^{(ra)} + \Delta_t^{(rc)})$ for Lloyd's algorithm, and $J = \sum_{t=1}^{\hat{I}}(\hat{\Delta}_t^{(ra)} + \hat{\Delta}_t^{(rc)})$ for ALG. Given the properties of Lloyd's algorithm, we know that $\Delta_t = \sum_{i=1}^{k} \Delta_i^{(t)}$ for all clusters at iteration t. According to Lemma 7.1, during *re-assignment* we have $\hat{\Delta}_i^{(t)} = (1-\delta^2) \times \Delta_i^{(t)}$, where $\delta = \mathbb{E}(\delta_i^{(t)})$. Thus, $\hat{\Delta}_t^{(ra)} = \sum_{i=1}^{k}[(1-\delta^2) \times \Delta_i^{(t)}] \in [\min_{i=1}^{k}\{1-(\delta_i^{(t)})^2\}, \max_{i=1}^{k}\{1-(\delta_i^{(t)})^2\}] \times \Delta_t^{(ra)}$. Considering the expected value of those Δs, we have $\hat{\Delta}_t^{(ra)} = (1-\delta^2) \times \Delta_t^{(ra)}$, $\delta =$

$\mathbb{E}(\delta_i^{(t)})$. In the worst case, $\hat{I} < \frac{1}{\min_{i,t}\{1-(\delta_i^{(t)})^2\}} \times I$. Given that $\hat{\Delta}_t^{(rc)} > \Delta_i^{(rc)}$, we have

$$\begin{aligned} J =& (\overline{\Delta^{(ra)}} + \overline{\Delta^{(rc)}}) \times I \\ =& (\overline{\hat{\Delta}^{(ra)}} + \overline{\hat{\Delta}^{(rc)}}) \times \hat{I} \\ >& [(1-\delta^2)\overline{\Delta^{(ra)}} + \overline{\Delta^{(rc)}}] \times \hat{I} \\ >& (1-\delta^2) \times (\overline{\Delta^{(ra)}} + \overline{\Delta^{(rc)}}) \times \hat{I}. \end{aligned}$$

Therefore, $\hat{I} < \frac{1}{1-\delta^2} \times I$ for the expected values of $\hat{I}$, I and δ. □

7.4 Sampling Zone Design

In this section, we first discuss the rules for building a *sampling zone*, and then propose the two designs for the *sampling zone*.

7.4.1 Design Rules

Ideally, in a *convergent zone*, when applying LapDP, the probability of a node S as the $\hat{S}_i^{(t)}$ needs to follow a monotonous decreasing function for the distance between S and $S_i^{(t)}$. A truncated LapDP [7] would be a straightforward way to achieve this. That is, once the random noise of LapDP is outside the *convergent zone*, we truncate it to the border of the *convergent zone*. However, this truncated LapDP will introduce issues, as the nodes in the border of the *convergent zone* may have a higher probability (sum of the probabilities of the nodes outside the *convergent zone*) than the ones closer to the $S_i^{(t)}$. Therefore, in this chapter, we apply the ExpDP in the *convergent zone* (in fact, in the *sampling zone*) to sample the $\hat{S}_i^{(t)}$.

When designing a *sampling zone*, we use the following rules. First, there should be a single *sampling zone* in $C_i^{(t)}$ for all parties, i.e., the trusted data curator and adversaries. Otherwise the differences among the *sampling zone*s in different parties will result in significant differences among their clustered results, which could be used for privacy inference. Second, the single *sampling zone* should not have an explicit relationship to $S_i^{(t)}$ (the real mean/centroid of $C_i^{(t)}$), as otherwise the adversary may easily learn the expected value of $S_i^{(t)}$ with high probability, and the expected value can be used as the real value. Third, to control the convergence orientation, the *orientation controller* should be used when building the *sampling zone*.

Based on the above discussions of the *sampling zone* and the research challenges presented in Section 7.3, we apply two strategies for the *orientation controller* to build two types of *sampling zone* in the following sections. The major difference between the two strategies is based on whether we use past knowledge only or both past and future knowledge [90] of the cluster centroids as the *orientation controller* for the *sampling zone*. Such a difference results in different clustering qualities and convergence rate.

7.4.2 Orientation Control with Past Knowledge

We observe that in $C_i^{(t)}$, past knowledge of the orientation of $S_i^{(t-1)} \to S_i^{(t)}$ indicates a trend of cluster centroid movement. Therefore, the *orientation controller* could be the point of intersection of the *convergent zone*'s borderline and the line $S_i^{(t-1)}S_i^{(t)}$. However, because such a point of intersection has an explicit relationship with $S_i^{(t-1)}$ and $S_i^{(t)}$, so we cannot use it as the *orientation controller* directly. To solve this problem, we simply shift this point of intersection with a random angle to have an *orientation controller*, $X_i^{(t)}$. Given that we still want the $X_i^{(t)}$ to be as close as the point of intersection, we use the following probability function for sampling an angle γ: $\Pr[\gamma_i^{(t)} = r] \propto \exp(1 - 2|r|/\pi)$, $r \in [0, \pi/2]$.

7.4.3 Orientation Control with Past and Future Knowledge

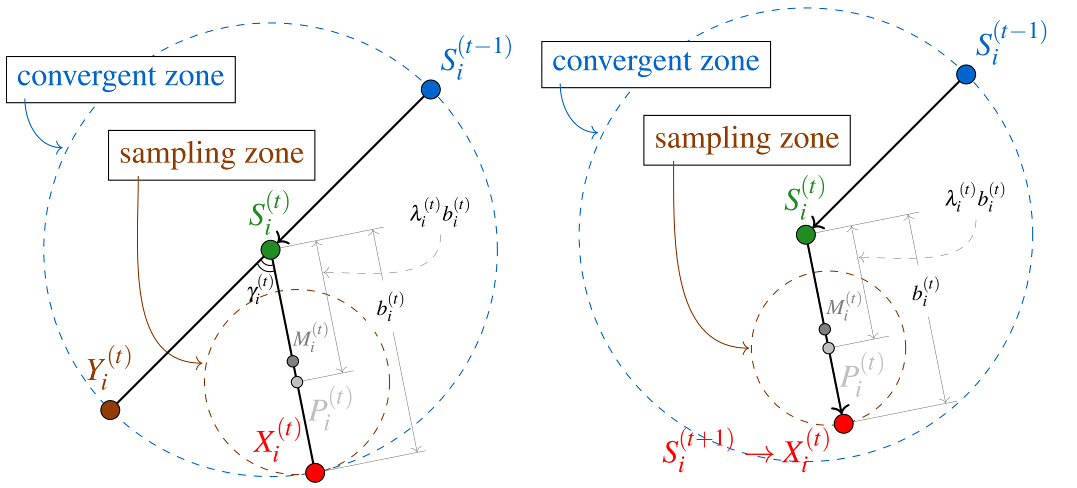
Clearly, the *sampling zone* with past knowledge of the cluster centroids cannot guarantee the convergence orientation towards the convergence of Lloyd's algorithm over the iterations. This results in a poor convergence degree. To improve the convergence quality, we use the centroid movement in future iterations as the orientation for centroid updating. As discussed, Lloyd's algorithm approaches a local optimum of the k-means clustering problem through iterations. If we use the final/convergent centroid, $S_i^{(t+r_t)}$, as the *orientation controller* in $C_i^{(t)}$, we can provide clustering quality using a random mechanism (i.e., the ExpDP) as much as possible. Note that, in $C_i^{(t)}$, $S_i^{(t+r_t)}$ is the future knowledge of the cluster centroids. However, taking such an $S_i^{(t+r_t)}$ means we have to further run Lloyd's algorithm for r_t iterations in $C_i^{(t)}$, which will result in a large rate of convergence when the differentially private algorithm converges. Therefore, considering the computational cost, we select the *orientation controller*, $X_i^{(t)}$, with $S_i^{(t+1)}$.

7.5 Proposed Algorithm and Associated Analysis

In this section, we present the differentially private k-means clustering algorithm with guaranteed convergence. We analyse its convergence, convergence rate and DP guarantees.

7.5.1 The Clustering Algorithm

The first step of our algorithm involves *sampling zone* generation. We generate our *sampling zone* by computing its centre and radius, respectively. The centre of the *sampling zone*, $P_i^{(t)}$, is determined by a random number $\lambda_i^{(t)} \in (1/2, 1)$, which is the offset in $\overline{S_i^{(t)} X_i^{(t)}}$. A larger *sampling zone* provides more choices for the $\hat{S}_i^{(t)}$; thus, we use the following probability function for sampling $\lambda_i^{(t)}$, $\Pr[P_i^{(t)}] = \Pr[\lambda_i^{(t)} = r] \propto \exp(2-2r) = p, r \in (1/2, 1)$. The radius of the *sampling zone* is $r_i^{(t)} = ||X_i^{(t)} - P_i^{(t)}||$. In this chapter, depending on whether we use past knowledge only or both past and future knowledge, we name the *sampling zone* as prior *sampling zone* (past knowledge) and posterior *sampling zone* (past and future knowledge). Algorithm 7.2 shows how we build the *sampling zone* with either past knowledge or past plus future knowledge regarding the cluster centroids. Figure 7.4 shows the key idea in building the *sampling zone*.



(a) Prior *Sampling Zone* (Past Knowledge). (b) Posterior *Sampling Zone* (Past and Future Knowledge).

Fig. 7.4 General Idea of the *Sampling Zone*.

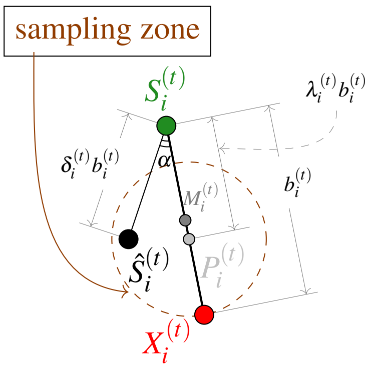


Fig. 7.5 Key Idea of Algorithm 7.1: Centroid Updating.

Once the *sampling zone* is established, each party samples their own $\hat{S}_i^{(t)}$ from this *sampling zone* with the ExpDP. In the implementation, we sample the $\hat{S}_i^{(t)}$ by sampling a pair $(\delta_i^{(t)} = \frac{||S_i^{(t)}-\hat{S}_i^{(t)}||}{||S_i^{(t)}-S_i^{(t+1)}||}, \alpha_i^{(t)} = \angle\hat{S}_i^{(t)}S_i^{(t)}S_i^{(t+1)})$, where $\delta_i^{(t)} \in (0,1)$, $\alpha_i^{(t)} \in (-\pi/2, \pi/2)$. An $\hat{S}_i^{(t)}$, that is close to the $S_i^{(t)}$, has better clustering quality for iterations in the interactive setting, as the scoring function should be monotonously decreasing for both $\delta_i^{(t)}$ and $\alpha_i^{(t)}$. In this chapter, we use the following scoring function for the pair $(\delta_i^{(t)}, \alpha_i^{(t)})$ because of its simplicity: $q(\delta_i^{(t)}, \alpha_i^{(t)}) = (1-\delta_i^{(t)}) + (1-2|\alpha_i^{(t)}|/\pi)$. It is easy to see that the local sensitivity of the scoring function is 2, i.e. $\Delta q = 2$.

Finally, when the clusters converge (to a real local optimum from Lloyd's algorithm), we apply the LapDP to inject noise to the final clustering result. Specifically, to have good clustering quality, we inject the Laplace noise to the counts when calculating the mean/centroid of each cluster (Line 12 in Algorithm 7.1). The local sensitivity of this counting function is 1. Algorithm 7.1 shows how the approach works.

Algorithm 7.1: Differentially Private k-Means Clustering.

```
Input  : $X = \{x_1, x_2, \ldots, x_N\}$: dataset in size $N$.
         $k$: number of clusters ($< N$).
         $\varepsilon_i^{(t)}$: privacy budget for Cluster $i$ at Iteration $t$, $C_i^{(t)}$.
         $\varepsilon_0$: privacy budget for the final output.
         $\Pr[P_i^{(t)}]$: probability to generate $SamplingZone_i^{(t)}$ for $C_i^{(t)}$.
         $q$: scoring function for the ExpDP when sampling the $\hat{S}_i^{(t)}$.
Output : $\mathbf{S}$: set of the final $k$ centroids.
1  Initialisation: Uniformly sample $k$ initial centroids $\mathbf{S}^{(0)} = (S_1^{(0)}, S_2^{(0)}, \ldots, S_k^{(0)})$ from
   $X$;
2  while clusters do not converge do
3      for each Cluster i at Iteration t do
4          $C_i^{(t)} \leftarrow$ assign each $x_j$ to its closest centroid $S_i^{(t-1)}$;
5          $S_i^{(t)} \leftarrow$ mean of $C_i^{(t)}$;
6          $SamplingZone_i^{(t)} \leftarrow$ run Algorithm 7.2;
7          $\hat{S}_i^{(t)} \leftarrow$ sample from $SamplingZone_i^{(t)}$ using ExpDP with $q$ and $\varepsilon_i^{(t)}$;
8          $S_i^{(t)} \leftarrow \hat{S}_i^{(t)}$;
9          Publish: $SamplingZone_i^{(t)}$, $q$, $\varepsilon_i^{(t)}$, $S_i^{(t)}$(optional);
10 $\mathbf{S} \leftarrow$ add noise to $\mathbf{S}^{(t)}$ by the LapDP with $\varepsilon_0$, publish $\varepsilon_0$;
```

7.5.2 Proof of Convergence and Differential Privacy

According to Lemma 7.1, 7.2 and 7.3, we have Theorem 7.1, 7.2, 7.3 and 7.4 to study the convergence and convergence rate of Algorithm 7.1, respectively. Theorem 7.5 studies the privacy bound of Algorithm 7.1.

Theorem 7.1. *Algorithm 7.1 (sampling zone with past knowledge) has convergence degree at least $1/m$, where m is the number of local optima of Lloyd's algorithm for a given dataset.*

Algorithm 7.2: The *Sampling Zone* Generator.

```
Input  : $S_i^{(t)}$: mean of $C_i^{(t)}$.
         $S_i^{(t-1)}$: mean of $C_i^{(t-1)}$.
         $\Pr[P_i^{(t)}]$: probability to generate $SamplingZone_i^{(t)}$ for $C_i^{(t)}$.
         $\Pr[\gamma_i^{(t)}]$: probability to generate angle $\gamma_i^{(t)}$.
         $useFuture$: future knowledge of cluster centroids.
Output : $SamplingZone_i^{(t)}$.
1  if useFuture is yes then
2      $C_i^{(t+1)} \leftarrow$ re-assign data points in $C_i^{(t)}$ by $S^{(t)} = \{S_i^{(t)}\}$;
3      $X_i^{(t)} \leftarrow$ mean of $C_i^{(t+1)}$;
4  else
5      $Y_i^{(t)} \leftarrow$ the point of intersection of the convergent zone's borderline and the line
       $S_i^{(t-1)} S_i^{(t)}$;
6      $\gamma_i^{(t)} \leftarrow$ sample by $\Pr[\gamma_i^{(t)}]$;
7      $X_i^{(t)} \leftarrow$ shift $Y_i^{(t)}$ on convergent zone's borderline with angle $\gamma_i^{(t)}$;
8  $M_i^{(t)} \leftarrow$ midpoint of the line segment $\overline{S_i^{(t)} X_i^{(t)}}$;
9  $P_i^{(t)} \leftarrow$ sample from the line segment $\overline{M_i^{(t)} X_i^{(t)}}$ by $\Pr[P_i^{(t)}]$;
10 $SamplingZone_i^{(t)} \leftarrow$ centre: $P_i^{(t)}$, radius: $r_i^{(t)} = ||X_i^{(t)} - P_i^{(t)}||$;
```

Proof. The convergent orientation is not determined when the *sampling zone* relies on past knowledge. With a uniform distribution for the orientation, if there are m local optimum of the k-means problem for a given dataset, the convergent degree will be at least $1/m$. □

Theorem 7.2. *The expected number of iterations of Algorithm 7.1 (sampling zone with past knowledge) to converge is at most double the iterations of Lloyd's algorithm.*

Proof. According to Lemma 7.1 and Theorem 7.4, the key points for analysing the convergence rate are the length of $||S_i^{(t)} - S_i^{(t+1)}||$ and $||\hat{S}_i^{(t)} - S_i^{(t+1)}||$. That is,

$$\hat{I} < \frac{||S_i^{(t)} - S_i^{(t+1)}||^2}{||\hat{S}_i^{(t)} - S_i^{(t+1)}||^2} \times I.$$

Given that, in this *sampling zone* (with past knowledge), we cannot determine the angle α in Figure 7.6 to determine the explicit expression for $||S_i^{(t)} - S_i^{(t+1)}||$ and $||\hat{S}_i^{(t)} - S_i^{(t+1)}||$, we simply use the triangle inequality to find the upper bound of $\hat{I}$. According to the triangle inequality, $(1-\delta_i^{(t)})^2 < \frac{||\hat{S}_i^{(t)}-S_i^{(t+1)}||^2}{||S_i^{(t)}-S_i^{(t+1)}||^2} < (1+\delta_i^{(t)})^2$. Note that, in this case, $\delta_i^{(t)}$ may be greater than 1. Thus, we have,

$$\hat{I} < \frac{1}{(1-\delta_i^{(t)})^2} \times I.$$

Given that our *sampling zone* is a subset of the *convergent zone*, we must have $\hat{I} \leq 2I$. Then we have $\hat{I} \leq \min\{2, \frac{1}{(1-\delta_i^{(t)})^2}\}$. □

Theorem 7.3. *Given a set of initial centroids, Algorithm 7.1 (sampling zone with past and future knowledge) has a convergence degree of 1, that is, it converges to the same (final) centroids as Lloyd's algorithm, with a probability of at least* $1-\frac{1}{2}(\frac{m}{n})^{\frac{d-1}{d}}$, *where n is the number of items in a dataset D, d is the dimension of an item and m is the number of local optima of Lloyd's algorithm on dataset D.*

Proof. In Algorithm 7.1, because each *sampling zone* is a subset of a *convergent zone*, according to Lemma 7.1, Algorithm 7.1 is convergent. According to Lemma 7.2, any arbitrary set of k nodes given as the initial set of centroids must converge to a local optimum in Lloyd's algorithm. However, for some sets of k nodes given as the initial centroids, they may belong to different local optimum. Such nodes appear at the border area between two local optima. Assume that a dataset D contains n items, where each item has d dimensions and the average distance between two items is l, then the overall size of the space of D is nl^d. The overall size of the border area space is $\frac{m}{2} \times (\frac{nl^d}{m})^{\frac{1}{d}} \times [2(b-\lambda b)]^{(d-1)}$, where m is the number of local optimum. Then we have at least $1-\frac{\frac{m}{2}\times(\frac{nl^d}{m})^{\frac{1}{d}}\times[2(b-\lambda b)]^{(d-1)}}{nl^d} \geq 1-\frac{1}{2}(\frac{m}{n})^{\frac{d-1}{d}}$ probability to not sample the initial nodes from the border area. Given that $||S_i^{(t+1)}-S_i^{(t)}|| < ||S_i^{(t-1)}-S_i^{(t)}||$, when $t>1$, all the sets of k nodes from $SamplingZone_i^{(1)}$ belong to same local optimum. Therefore, based on Lemma 7.2, this theorem holds. □

Theorem 7.4. *The expected number of iterations of Algorithm 7.1 (sampling zone with past and future knowledge) to converge is at most* $\frac{2}{-\delta^2+2\delta\cos\alpha+1} \in (1,2)$ *times the number of iterations of Lloyd's algorithm, where* δ *and* α *are the expected values of* $\delta_i^{(t)}$ *and* $\alpha_i^{(t)}$.

Proof. According to Lemma 7.3, the total number of iterations of Algorithm 7.1 depends on the distance $||\hat{S}_i^{(t)} - S_i^{(t+1)}||$. As seen in Figure 7.6, we have:

$$\begin{aligned}
&||\hat{S}_i^{(t)} - S_i^{(t+1)}||^2 \\
=&||\hat{S}_i^{(t)} - Temp||^2 + ||Temp - S_i^{(t+1)}||^2 \\
=&||\hat{S}_i^{(t)} - Temp||^2 + (||S_i^{(t)} - S_i^{(t+1)}|| - ||Temp - S_i^{(t)}||)^2 \\
=&[(\delta_i^{(t)})^2 - 2\delta_i^{(t)}\cos\alpha_i^{(t)} + 1](b_i^{(t)})^2.
\end{aligned}$$

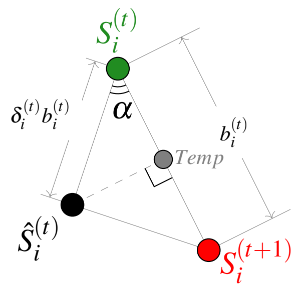


Fig. 7.6 Help Figure for Proof of Theorem 7.4.

Then we have the ratio

$$\frac{||\hat{S}_i^{(t)} - S_i^{(t+1)}||}{||S_i^{(t)} - S_i^{(t+1)}||} = \sqrt{(\delta_i^{(t)})^2 - 2\delta_i^{(t)}\cos\alpha_i^{(t)} + 1}.$$

In Algorithm 7.1, we calculate the centroid $S_i^{(t+1)}$ at iteration t, where it is expected to have a $\Delta_t + \Delta_{t+1}$ change for the cost value. However, by applying a similar idea from

Lemma 7.3, what we have is

$$
\begin{aligned}
&\hat{\Delta}_t + \hat{\Delta}_{t+1} \\
>&\Delta_t + (1 - (\sqrt{\delta^2 - 2\delta\cos\alpha + 1})^2) \times \Delta_{t+1} \\
=&\Delta_t + (-\delta^2 + 2\delta\cos\alpha) \times \Delta_{t+1},
\end{aligned}
$$

where $\hat{\Delta}_t = \Delta_t$.

Recall how Algorithm 7.1 converges, whereby half the iterations decrease the cost function by $\hat{\Delta}_t$ and half the iterations do so as Δ_{t+1}. Thus, assuming $\hat{I} = T \times I$, then the overall decrease of the cost function is given by:

$$
\begin{aligned}
&\frac{1}{2}\hat{I}\overline{\hat{\Delta}_t} + \frac{1}{2}\hat{I}\overline{\hat{\Delta}_{t+1}} \\
>&\frac{1}{2}\hat{I}\overline{\Delta_t} + \frac{1}{2}(-\delta^2 + 2\delta\cos\alpha) \times \hat{I}\overline{\Delta_{t+1}} \\
=&\frac{1}{2}T \times I\overline{\Delta_t} + \frac{1}{2}(-\delta^2 + 2\delta\cos\alpha)T \times I\overline{\Delta_{t+1}}.
\end{aligned}
$$

Then we have $\frac{1}{2}T + \frac{1}{2}(-\delta^2 + 2\delta\cos\alpha)T < 1$, so $T < \frac{2}{-\delta^2 + 2\delta\cos\alpha + 1}$. Note that, since $||\hat{S}_i^{(t)} - S_i^{(t+1)}|| < ||S_i^{(t)} - S_i^{(t+1)}||$, $\forall$ t and i, $\sqrt{(\delta_i^{(t)})^2 - 2\delta_i^{(t)}\cos\alpha_i^{(t)} + 1}$ is in $(0,1)$, then $\frac{2}{-(\delta_i^{(t)})^2 + 2\delta_i^{(t)}\cos\alpha_i^{(t)} + 1}$ is in $(1,2)$. Thus, based on Lemma 7.1 and Lemma 7.3, this theorem holds. □

Theorem 7.5. *Algorithm 7.1 is ε-differentially private, where $\varepsilon = \varepsilon_0 + \sum_{t=1}^{\hat{I}} \max_{i=1}^{k}\{\varepsilon_i^{(t)}\}$, $\hat{I}$ is the total number of iterations to converge.*

Proof. When applying the ExpDP to sample $\hat{S}_i^{(t)}$ (Line 11, Algorithm 7.1) in $C_i^{(t)}$, we have

$$
\begin{aligned}
&\frac{\Pr[\hat{S}_i^{(t)} = S]}{\Pr[\hat{S}_i'^{(t)} = S]} = \frac{\Pr[S, P_i^{(t)}, S_i^{(t)}, S_i'^{(t)} \textit{ in a plane}] \times \exp(\frac{\varepsilon_i^{(t)} h(\delta_i^{(t)}, \alpha_i^{(t)})}{2\Delta h})}{\Pr[S, P_i^{(t)}, S_i^{(t)}, S_i'^{(t)} \textit{ in a plane}] \times \exp(\frac{\varepsilon_i^{(t)} h(\delta_i'^{(t)}, \alpha_i'^{(t)})}{2\Delta h})} \\
\leq &\exp(\varepsilon_i^{(t)}).
\end{aligned}
$$

Thus, Algorithm 7.1 guarantees $\varepsilon_i^{(t)}$-DP in $C_i^{(t)}$. In each iteration, all items $x_i \in X$ are split into k mutually exclusive clusters, based on parallel and sequential composition [99] after $\hat{I}$ iterations. Algorithm 7.1 is ε-differentially private, where $\varepsilon = \varepsilon_0 + \sum_{t=1}^{\hat{I}} \max_{i=1}^{k}\{\varepsilon_i^{(t)}\}$. Note that, as the Lloyd's k-means algorithm usually converges after a small number of iterations, according to Theorem 7.4 and Theorem 7.2, the value of the overall ε would not be very large in average. $\square$

7.6 Experimental Evaluation

7.6.1 Datasets and Configuration

Table 7.2 illustrates the key features of the real-world datasets used to evaluate the clustering quality and convergence rate of Algorithm 7.1. As a matrix, each dataset contains #Records $\times$ #Dims cells. We use these datasets for two reasons. First, they are used for the clustering experiments in several research chapters for k-means clustering tasks, such as in [73, 133] for normal k-means clustering and [176, 148] for differentially private k-means. Second, their sizes are in different orders of magnitude, which help us to show the performance stability and scalability of our algorithm over different datasets. Iris [26] is one of the most famous datasets for clustering containing the sepal length/width and the petal length/width of three types of iris plants. House [45] and Image [45] are image datasets, consisting of the RGB-values of house pictures, where each colour is quantised by 5 bits and 8 bits, respectively. S1 [44] is a synthetic 2-D dataset with 15 non-overlapping Gaussian clusters. Birch2 [177] is also a synthetic 2-D dataset, where each cluster's centroid is placed in a sine curve. Lifesci [75] is a life sciences dataset, where 10 principal components are used for chemistry/biology experiments that fall into three categories.

We compare the clustering quality of Algorithm 7.1 (in both prior and posterior *sampling zones*) with state-of-the-art ε-differentially private k-means clustering algorithms and Lloyd's

Table 7.2 Descriptions of Datasets.

Dataset	#Records	#Dims	#Clusters
Iris [26]	150	4	3
House [45]	1,837	3	3
S1 [44]	5,000	2	15
Birch2 [177]	12,000	2	5
Image [45]	34,112	3	3
Lifesci [75]	26,733	10	3

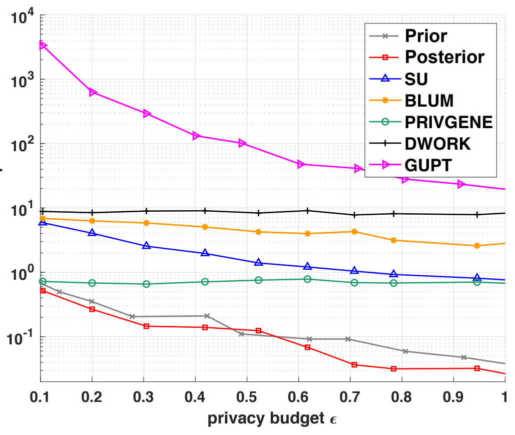
algorithm. The clustering quality is measured by the difference/gap of the final cost (Equation 7.1) between a differentially private *k*-means clustering algorithm and Lloyd's algorithm. A smaller gap indicates better clustering quality. In the experiments, we implement and name them as Prior (Algorithm 7.1 with past knowledge [91]), Posterior (Algorithm 7.1 with past and future knowledge [90]), SU [148], PrivGene [176], GUPT [104], DWORK [34], BLUM [12] and LLOYD [87]. In brief, SU [148] proposes an optimal privacy budget allocation scheme to allocate the overall privacy budget to each iteration. PrivGene [176] applies genetic algorithm that randomly samples the candidates with ExpDP for each iteration, and then applies the crossover and mutate operations of traditional GAs to iteratively update the centroids. GUPT [104] uniformly samples items from an input dataset to different buckets, where the local clustering results of each bucket are produced by Lloyd's algorithm. The final clustering result is the mean of local results with Laplace noise injected. DWORK [34] allocates the overall privacy budget to each iteration with decreasing exponential distributions. BLUM [12] predefines a number of iteration, and then allocates the overall privacy budget to each iteration uniformly. Details about SU [148], PrivGene [176], GUPT [104], DWORK [34] and BLUM [12] can be found in Section 3.4. Details about LLOYD [87] can be found in Section 7.2.

Given that the six algorithms that achieve ε-DP are randomised, we report their expected clustering quality. According to the law of large numbers, we run all seven algorithms 300 times and take the average results. The initial set of centroids is randomly selected for all methods in each run. For those relying on a predefined iteration number, we take the corresponding value (or function) from the original chapters. In addition, we normalise the data in all datasets to $[0,1]$. Further, we normalise the final cost of all involved algorithms, where the final cost of Lloyd's algorithm is always one.

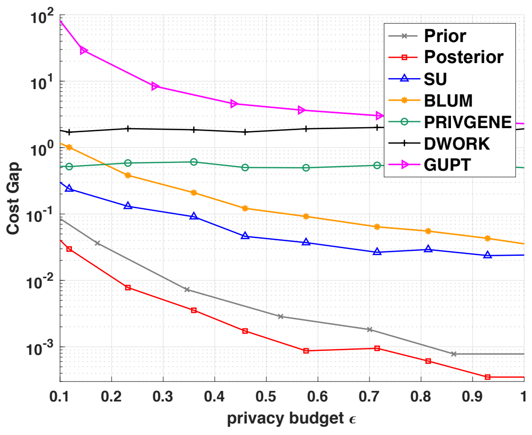
In each run, LLOYD, BLUM, DWORK, SU, and Algorithm 7.1 use the same initial centroids. Given that GUPT starts by splitting the original datasets into several buckets, it cannot use the same initial centroids as LLOYD. Note that calculating the overall privacy budget depends on whether a method converges. Algorithm 7.1 and GUPT calculate the overall privacy budget bottom-up. That is, once it terminates, we sum all the privacy budgets used in each iteration to attain the overall privacy budget. SU, PrivGene, DWORK, and BLUM calculate it top-down. Namely, the given overall privacy budget is split into each iteration at the initialisation step. Therefore, in the experiments, we first allocate the same privacy budget to each step for Algorithm 7.1 and GUPT, and then calculate their overall privacy budgets. Next we take the overall privacy budget of Algorithm 7.1 as the overall privacy budget for the methods that cannot converge. In the experiments, local sensitivity is applied for all DP algorithms. (Source code: https://github.com/suluz/diffpriv_clustering)

7.6.2 Experimental Results

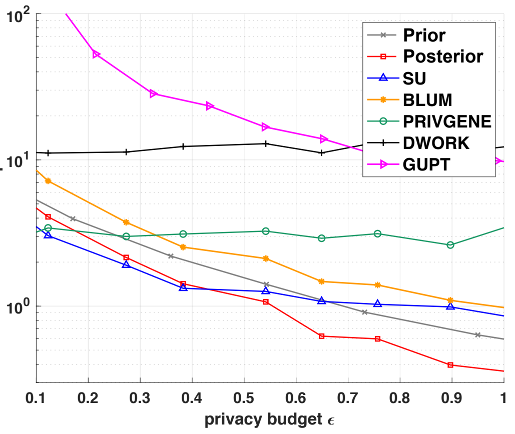
Figure 7.7 shows the expected clustering quality of each algorithm, where the cost gap is given in log scale and the privacy budget varies from $[0.1, 1.0]$. Generally, Algorithm 7.1 outperforms the state-of-the-art results with the same DP requirement in the six datasets in both prior and posterior cases. Additionally, the performance gap between Algorithm 7.1 and existing algorithms increases when increasing ε, which indicates a better trade-off between privacy and utility with our algorithm. Further, Algorithm 7.1 performs much better than



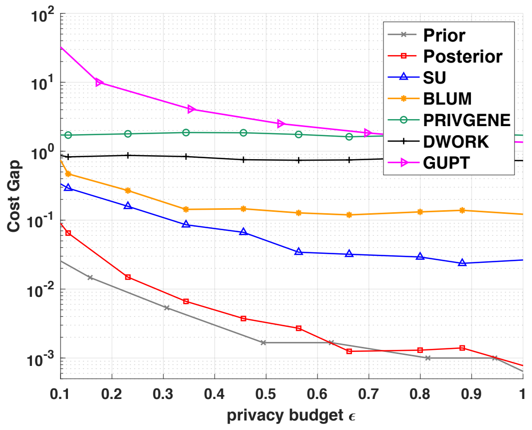
(a) Iris ($k = 3$)



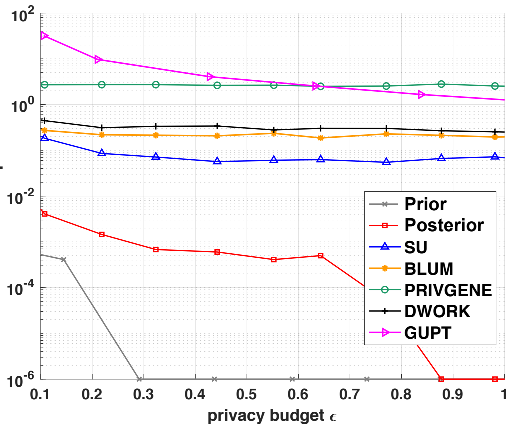
(b) House ($k = 3$)



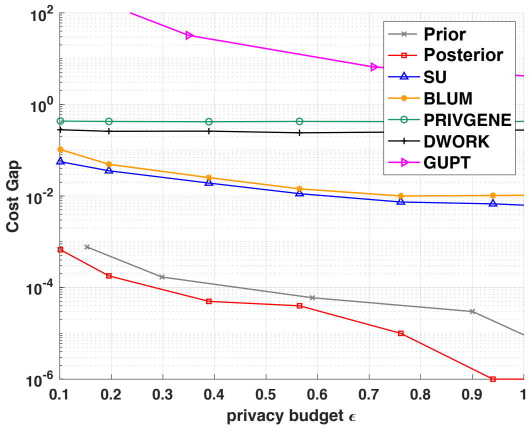
(c) S1 ($k = 15$)



(d) Birch2 ($k = 5$)



(e) Image ($k = 3$)

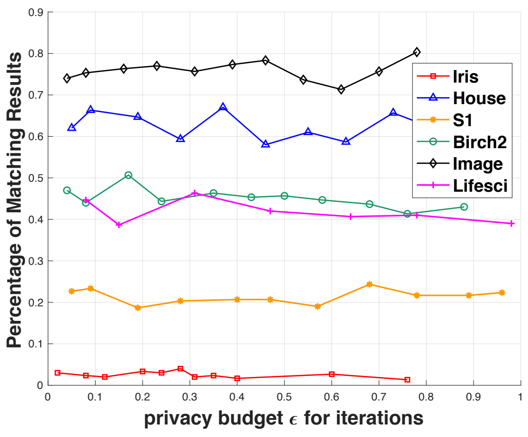


(f) Lifesci ($k = 3$)

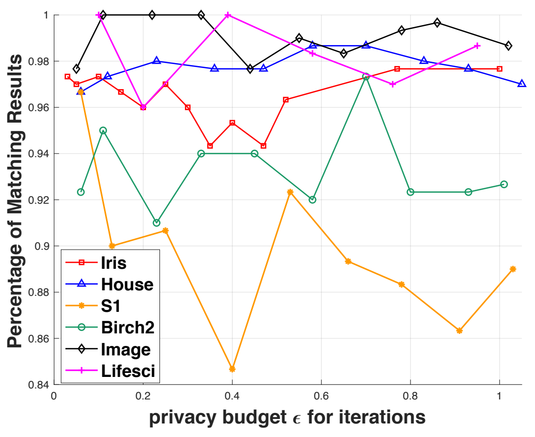
Fig. 7.7 Clustering Quality Comparisons.

other algorithms using larger datasets (e.g., Image and Lifesci), which reflects the scalability of our algorithm.

Figure 7.8 depicts the convergence degree of Algorithm 7.1 using two strategies. We study the convergence degree by comparing whether the output set of centroids of our approach (without the final DP noise of Line 10 in Algorithm 7.1) is the same as that of Lloyd's algorithm. Given that we round the values in the clustering process, once the output of our approach is in $[0.99, 1.01]$ of Lloyd's algorithm, we call it a *match*. We report the percentage of the matching results of the two strategies over all six datasets as the convergence degree. From Figure 7.8, the posterior strategy, which uses both past and future knowledge, outperforms the prior strategy, which only uses the past knowledge, in convergence performance because of the convergence guarantees of Theorem 7.3. The posterior strategy matches at least 84% (90% in most cases) output centroids of Lloyd's algorithm, while the prior matches no more than 80% (50 % in most cases).



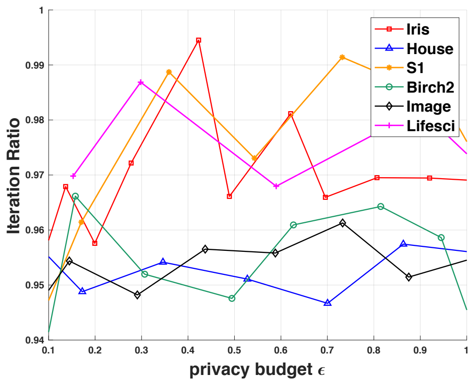
(a) Convergence Degree of the Prior Strategy (Past Knowledge)



(b) Convergence Degree of the Posterior Strategy (Past and Future Knowledge)

Fig. 7.8 Convergence Degree of Our Approach.

Figure 7.9 and Figure 7.10 show the iteration ratio between Algorithm 7.1 and Lloyd's algorithm and how they converge, which confirms the theoretical analyses of Theorem 7.2 and Theorem 7.4. That is, using the past knowledge (prior strategy) for centroid updating provides faster convergence rate than using past and future knowledge (posterior strategy). In particular, we compare the numbers of iterations that Algorithm 7.1 and Lloyd's algorithm execute until they terminate. Note that, in the experiments, the privacy budget does not affect the number of iterations significantly because the experimental performance of ExpDP is not as good as its theoretical guarantee with a relatively small *sampling zone*.

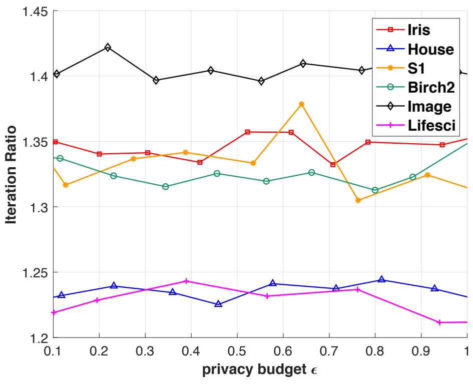


(a) Iteration Ratio over ε.

	Iris	House	S1	Birch2	Image	Lifesci
Prior	5.89	16.40	17.05	14.73	13.60	28.92
LLOYD	6.07	17.22	17.64	15.52	14.26	29.72
Ratio	0.97	0.95	0.97	0.95	0.95	0.97

(b) Average Iterations to Convergence.

Fig. 7.9 Iterations of the Prior Strategy (Past Knowledge) to Convergence.



(a) Iteration Ratio over ε.

	Iris	House	S1	Birch2	Image	Lifesci
Posterior	8.2	21.4	23.9	20.7	19.8	36.6
LLOYD	6.1	17.3	18.0	15.6	14.1	29.9
Ratio	1.34	1.24	1.33	1.33	1.40	1.22

(b) Average Iterations to Convergence.

Fig. 7.10 Iterations of the Posterior Strategy (Past and Future Knowledge) to Convergence.

7.7 Summary

To address the non-convergence problem of the existing algorithms for differentially private *k*-means clustering in interactive settings, in this chapter, we have proposed a novel centroid updating approach by applying the exponential mechanism of DP in a selected area. The novelty of our approach is the orientation control of centroid movement for noise injection in the iterations of the clustering process to achieve convergence. We have demonstrated the key properties of our approach and showed that it converges in at most twice as many iterations as Lloyd's *k*-means algorithm. The experimental evaluations validated that, with the same DP guarantees, our algorithm ensures convergence and achieves better clustering quality than the state-of-the-art differentially private algorithms in the interactive setting.

Chapter 8

Conclusions

In this thesis, we have studied differentially private data analytics from four aspects: differentially private data aggregation with security guarantees, privacy budget guarantees of distributed systems, differentially private single-path publishing and differentially private k-means clustering. From Chapter 4 to Chapter 7, for each research aspect, we proposed our novel approach and techniques that outperformed existing similar works in both mathematical and experimental evaluations. In this chapter, we summarise the main contributions of this thesis, and then provide some possible research directions for future work on differentially private data analytics.

8.1 Thesis Summary

With the rapid growth of data-driven applications, the data analytics behind such applications often use personal data to offer personalised services. Such processes occur at every single minute around the world; however, as discussed throughout this thesis, these processes compromise users' privacy. Specifically, the data analytics not only gain insights from these data to provide services, but also potentially violate users' privacy. For instance, to provide an accurate recommendation, a normal recommender system should learn our exact preferences by using historical records, and then make some recommendations about our potential interest based on such information. In this example, our preferences are part of our privacy. Therefore, to protect our privacy while having the benefits of these digital applications, it is essential to develop new methods and techniques for privacy preservation during data analytics to achieve acceptable trade-offs between individual privacy and data utility.

As discussed in Chapter 1, the main contributions of this thesis are as follows. First, we present a two-layer differentially private data aggregation with security guarantees against semi-honest yet colluding data aggregators, which ensures both high data utility and low communication cost. Second, we offer a new lower bound of privacy budget for distributed differentially private aggregation/summation. Third, we propose a graph-based single-path publishing approach that preserves the DP of each edge against adversaries with full information about other vertices and maximum background knowledge about the edges. Fourth, we provide a differentially private *k*-means clustering framework with convergence guarantees.

As discussed in Chapter 2, DP has emerged as an important approach for privacy preservation because of its mathematical bounds for privacy and utility trade-offs. In this thesis, we use it as the main notion of privacy. Specifically, we apply DP for some computing tasks of a data analytics process, as specificed in Section 1.1: data aggregation, data publishing and data mining/machine learning. We review the state-of-the-art works in Chapter 3. These underpin the research challenges of this thesis and motivate the work as a whole. From Chapter 4 to Chapter 7, we demonstrate the technical details of our solutions.

When aggregating data from distributed data curators, existing work achieves both output privacy and computing security; however, it faces at least one of the following problems: (1) collusion between data curators, (2) collusion between data aggregators (in the case of multiple aggregators), (3) the vulnerability of a single data aggregator and (4) high communication cost (in the case of no aggregator). To address those research challenges, in Chapter 4, we propose a two-layer data aggregation system based on semi-honest yet colluding aggregators. Our solution achieves DP for data output with high data utility by random DP noise injection to achieve security by secure information sharing, while achieving low communication costs by its non-complete topology. The experiments on real-world datasets show that it achieves the same privacy requirement and provides enhanced data utility.

To tackle the research challenges of unknown privacy guarantees for conditional local function sensitivities and poor data utility achievable using unconditional global function sensitivities for distributed differentially private summation, Chapter 5 provides a novel lower bound of privacy budget to measure the parallel composition of privacy budget during distributed differentially private summation. By taking advantage of a special property of the summation function when calculating function sensitivity, our solution guarantees an acceptable overall privacy performance without any conditions on the function sensitivity. We also confirm the theoretical bound by running experiments on the real-world datasets.

The existing differentially private single-path publishing algorithms preserve privacy against adversaries with maximum background knowledge of the vertices ($n-1$ out of n vertices in a graph/network), but no information about the edges. However, those algorithms are vulnerable once adversaries have different auxiliary information, such as full knowledge about the vertices plus $m-1$ out of m edges. In Chapter 6, we present a graph-based single-path publishing solution. In particular, we insert fake vertices and fake edges with DP to the original path, and then hide the perturbed paths in the topology of the publishing graph with a given rule. Through this, our approach ensures that only someone who has full knowledge about the original path could recover the exact path from the published graph. We use random synthetic datasets to experimentally evaluate the performance of the algorithm, including run time and rate of path recovery using different sizes of random maps.

The property of convergence is essential to randomised (differentially private) k-means clustering algorithms. Without convergence guarantees, the differentially private clustering output would be far from any local optimum of the k-means problem with given inputs. That is, the clustering quality is severely affected by convergence. To deal with the non-convergence problem with the existing differentially private k-means clustering algorithms, Chapter 7 researches the convergence and convergence rates of randomised k-means clustering. We show that, at any iteration, there is an area in each cluster where sampling any node as the updated centroid with DP can guarantee the convergence. To further ensure the convergence rate, we update the centroids with a given orientation based on the information of either past centroid movement or future centroid movement. The real-world experiments demonstrate that, because of the convergence guarantees, our clustering quality outperforms existing work.

To conclude, this thesis has developed new methods and techniques for differentially private data analytics to address the research challenges with differentially private data aggregation with security guarantees, with privacy budget guarantees of distributed systems, with differentially private single-path publishing, and with differentially private k-means clustering. We believe that the outcomes of this work will allow individuals to enjoy the benefits of data-driven products without concern about the disclosure of their private information.

8.2 Research Directions for Future Work

Although we provide novel solutions to address the research challenges with existing work on differentially private data analytics, there remain some possible extensions or research directions that could form the basis for future work:

- **Differentially private data cleansing**. In general, as a part of data aggregation, data cleansing is often used to remove noise from the original data, which is important for other data analytics tasks because the insights gained from the raw data heavily rely on the quality of data cleansing. In contrast, for privacy-preservation purposes, as discussed in Chapters 2 and 3, it is essential to have DP prior to using the data, that is, injecting DP noise into the source data. Based on the above, as we must have noise in the raw data anyway, it would be more efficient to achieve DP when cleansing data, that is, retain some original noise with the data for DP purposes. Thus far, to the best of our knowledge, only two works [76, 59] have focused on this topic; however, both of them simply combine DP with normal data cleansing (i.e., cleansing the raw data first, then injecting DP noise), which are not efficient enough.

- **A unified and unconditional lower bound of the parallel composition of privacy budget for data aggregation**. In Chapter 5, our lower bound for the parallel composition of the privacy budget only reflects the overall privacy performance of the distributed summation/counting function, without any conditions on local sensitivities of data curators and data aggregators. However, as our result benefits from a special property of the summation/counting function, it cannot be used on other distributed aggregation functions directly. Therefore, in the future, we could study the parallel composition of privacy budget for other distributed aggregation functions, such as minimum/maximum, mean and variance, so that we would have a unified lower bound of the parallel composition of the privacy budget that works with unconditional local sensitivities. In addition, a similar future work extension is also needed for the secure data aggregation studied in Chapter 4. We will consider extending the existing solution to more complex aggregation functions, such as classification and clustering.

- **Differentially private single path publishing against truly arbitrary auxiliary information**. Existing work in differentially private single-path publishing preserves the privacy of vertices against adversaries that are missing information about the vertices of the path, potentially $n-1$ out of n vertices of the path. In comparison, our work in Chapter 6 preserves the privacy about edges against adversaries with information about $m-1$ out of m edges of the path. Clearly, both our approach and the state-of-the-art approaches preserve privacy with DP by making specific assumptions about adversaries. Without such assumptions, the performance of privacy preservation would not be guaranteed. Therefore, it is crucial to discover a differentially private single-path publishing approach that works against truly arbitrary auxiliary information, including adversaries with maximum knowledge of vertices, edges, or both vertices and edges.

- **Optimal privacy budget allocation for a convergent differentially private k-means clustering**. The privacy budget allocation presented in Chapter 7 follows a bottom-up strategy. Namely, we allocate the same privacy budget to each iteration, and then calculate the overall privacy budget according to the total number of iterations until the algorithm converges. However, if it converges only after a large number of iterations, the overall privacy budget would be very high. Therefore, it would be preferable to have a top-down strategy as the allocation scheme, that is, we set a predefined privacy budget for the whole algorithm to bound the overall privacy performance, and then split the allocation privacy budget to each iteration. It is essential to study an optimum privacy budget allocation scheme that ensures every iteration has sufficient DP noise. The difficulty of such a scheme is that the total number of iterations is typically unknown when determining the amount of privacy budget to allocate to a given iteration.

- **Extension of using global sensitivity**. In this thesis, we use local sensitivity as the implementation of function sensitivity in both theoretical and experimental analyses, which helps achieve enhanced data utility. However, local sensitivity limits the significance of the results, as it does not consider all possibilities of the private dataset over a given distribution. To attain a comprehensive analysis, global sensitivity should be considered because it covers all possible neighbouring datasets in the context of DP [144]. Therefore, it would be meaningful to extend both the theoretical and experimental evaluations by using global sensitivity in future studies.

References

[1] Gergely Ács and Claude Castelluccia. I have a dream! (differentially private smart metering). In *International Workshop on Information Hiding*, pages 118–132. Springer, 2011.

[2] Erfan Aghasian, Saurabh Garg, Longxiang Gao, Shui Yu, and James Montgomery. Scoring users' privacy disclosure across multiple online social networks. *IEEE Access*, 5:13118–13130, 2017.

[3] Khalil Al-Hussaeni, Benjamin CM Fung, Farkhund Iqbal, Gaby G Dagher, and Eun G Park. Safepath: Differentially-private publishing of passenger trajectories in transportation systems. *Computer Networks*, 143:126–139, 2018.

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[5] Dima Alhadidi, Noman Mohammed, Benjamin CM Fung, and Mourad Debbabi. Secure distributed framework for achieving ε-differential privacy. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 120–139. Springer, 2012.

[6] Balamurugan Anandan and Chris Clifton. Laplace noise generation for two-party computational differential privacy. In *The 13th Annual Conference on Privacy, Security and Trust*, pages 54–61. IEEE, 2015.

[7] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 901–914. ACM, 2013.

[8] Credit Union National Association. Data breach costs will soar to $2t: Juniper, 2015. URL https://news.cuna.org/articles/105948-data-breach-costs-will-soar-to-2t-juniper. [Online; accessed 23-October-2019].

[9] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In *Theory of Cryptography Conference*, pages 336–354. Springer, 2004.

[10] Muzammil M Baig, Jiuyong Li, Jixue Liu, Xiaofeng Ding, and Hua Wang. Data privacy against composition attack. In *International Conference on Database Systems for Advanced Applications*, pages 320–334. Springer, 2012.

[11] Amos Beimel. Secret-sharing schemes: a survey. In *Coding and Cryptology*, pages 11–46. Springer, 2011.

[12] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138. ACM, 2005.

[13] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*, pages 192–206. Springer, 2008.

[14] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[15] Sabrina Boubiche, Djallel Eddine Boubiche, Azeddine Bilami, and Homero Toral-Cruz. Big data challenges and data aggregation strategies in wireless sensor networks. *IEEE Access*, 6:20558–20571, 2018.

[16] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. “you might also like:" privacy risks of collaborative filtering. In *The 2011 IEEE Symposium on Security and Privacy*, pages 231–246. IEEE, 2011.

[17] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

[18] Jianneng Cao, Fang-Yu Rao, Elisa Bertino, and Murat Kantarcioglu. A hybrid private record linkage scheme: separating differentially private synopses from matching records. In *The 31st IEEE International Conference on Data Engineering*, pages 1011–1022. IEEE, 2015.

[19] Yang Cao, Li Xiong, Masatoshi Yoshikawa, Yonghui Xiao, and Si Zhang. Contpl: controlling temporal privacy leakage in differentially private continuous data release. *Proceedings of the VLDB Endowment*, 11(12):2090–2093, 2018.

[20] Yang Cao, Yonghui Xiao, Li Xiong, and Liquan Bai. Priste: from location privacy to spatiotemporal event privacy. pages 1606–1609, 2019.

[21] Rui Chen, Benjamin Fung, Bipin C Desai, and Nériah M Sossou. Differentially private transit data publication: a case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–221. ACM, 2012.

[22] Shu Chen and Hong Shen. Semantic-aware dummy selection for location privacy preservation. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 752–759. IEEE, 2016.

[23] Richard Cleve, Daniel Gottesman, and Hoi-Kwong Lo. How to share a quantum secret. *Physical Review Letters*, 83(3):648, 1999.

[24] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation, an introduction. *Contemporary Cryptology*, pages 41–87, 2009.

[25] Fatemeh Deldar and Mahdi Abadi. Pldp-td: Personalized-location differentially private data analysis on trajectory databases. *Pervasive and Mobile Computing*, 49:1–22, 2018.

[26] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. archive.ics.uci.edu/ml.

[27] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[28] Bolin Ding, Harsha Nori, Paul Li, and Joshua Allen. Comparing population means under local differential privacy: with significance and power. In *The 32nd AAAI Conference on Artificial Intelligence*, pages 26–32, 2018.

[29] Domo. Data never sleeps 6.0. https://www.domo.com/learn/data-never-sleeps-6, 2018. [Online; accessed 21-October-2019].

[30] Domo. Data never sleeps 7.0. https://www.domo.com/learn/data-never-sleeps-7, 2019. [Online; accessed 21-October-2019].

[31] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[32] Yitao Duan. Differential privacy for sum queries without external noise. In *ACM Conference on Information and Knowledge Management*, 2009.

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

[34] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[35] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[36] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

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

[38] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 316–325. ACM, 2014.

[39] Tariq Elahi, George Danezis, and Ian Goldberg. Privex: Private collection of traffic statistics for anonymous communication networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1068–1079. ACM, 2014.

[40] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2 (2-3):70–246, 2018.

[41] Michael J Evans and Jeffrey S Rosenthal. *Probability and statistics: The science of uncertainty*. Macmillan, 2004.

[42] Nivan Ferreira, James T Klosowski, Carlos E Scheidegger, and Cláudio T Silva. Vector field k-means: Clustering trajectories by fitting multiple vector fields. In *Computer Graphics Forum*, volume 32, pages 201–210. Wiley Online Library, 2013.

[43] Leslie R Foulds. *Graph theory applications*. Springer Science & Business Media, 2012.

[44] Pasi Fränti and Olli Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–775, 2006. //cs.uef.fi/sipu/datasets/.

[45] Pasi Fränti and Olli Virmajoki. Clustering datasets, 2018. cs.uef.fi/sipu/datasets/.

[46] Benjamin CM Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.

[47] David Garcia. Leaking privacy and shadow profiles in online social networks. *Science Advances*, 3(8):e1701172, 2017.

[48] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Aymposium on Theory of Computing*, pages 169–178. ACM, 2009.

[49] Gerry and the Pacemakers. You'll never walk along, 1963. URL https://open.spotify.com/artist/3UmBeGyNwr4iDWi1vTxWi8. [Online; accessed 26-October-2019].

[50] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.

[51] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[52] Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 14(5):463–477, 2015.

[53] Vipul Goyal, Ilya Mironov, Omkant Pandey, and Amit Sahai. Accuracy-privacy tradeoffs for two-party differentially private protocols. In *Advances in Cryptology*, pages 298–315. Springer, 2013.

[54] Ke Gu, Lihao Yang, Yongzhi Liu, and Bo Yin. Efficient trajectory data privacy protection scheme based on laplace's differential privacy. *Informatica*, 42(3), 2018.

[55] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, 42(4): 767–799, 2014.

[56] Longkun Guo and Hong Shen. Privacy-preserving internet traffic publication. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 884–891. IEEE, 2016.

[57] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the 21st annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1106–1125. SIAM, 2010.

[58] Mehmet Emre Gursoy, Ling Liu, Stacey Truex, and Lei Yu. Differentially private and utility preserving publication of trajectory data. *IEEE Transactions on Mobile Computing*, 2018.

[59] Qilong Han, Qianqian Chen, Liguo Zhang, and Kejia Zhang. Hrr: a data cleaning approach preserving local differential privacy. *International Journal of Distributed Sensor Networks*, 14(12):1550147718819938, 2018.

[60] Xi He, Ashwin Machanavajjhala, and Bolin Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 1447–1458. ACM, 2014.

[61] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia M Procopiuc, and Divesh Srivastava. Dpt: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165, 2015.

[62] Xi He, Nisarg Raval, and Ashwin Machanavajjhala. A demonstration of visdpt: visual exploration of differentially private trajectories. *Proceedings of the VLDB Endowment*, 9(13):1489–1492, 2016.

[63] Yuan Hong, Jaideep Vaidya, Haibing Lu, Panagiotis Karras, and Sanjay Goel. Collaborative search log sanitization: Toward differential privacy and boosted utility. *IEEE Transactions on Dependable and Secure Computing*, 12(5):504–518, 2015.

[64] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[65] Jingyu Hua, Yue Gao, and Sheng Zhong. Differentially private publication of general time-serial trajectory data. In *2015 IEEE Conference on Computer Communications*, pages 549–557. IEEE, 2015.

[66] Kit Huckvale, José Tomás Prieto, Myra Tilney, Pierre-Jean Benghozi, and Josip Car. Unaddressed privacy risks in accredited health and wellness apps: a cross-sectional systematic assessment. *BMC medicine*, 13(1):214, 2015.

[67] IDC. Data growth, business opportunities, and the it imperatives, 2014. URL https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm. [Online; accessed 21-October-2019].

[68] Ioannis Ioannidis and Ananth Grama. An efficient protocol for yao's millionaires' problem. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 6–pp. IEEE, 2003.

[69] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Annual International Cryptology Conference*, pages 145–161. Springer, 2003.

[70] Sai Ji, Yajie Sun, and Jian Shen. A method of data recovery based on compressive sensing in wireless structural health monitoring. *Mathematical Problems in Engineering*, 2014, 2014.

[71] Kaifeng Jiang, Dongxu Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, page 12. ACM, 2013.

[72] Ari Juels. Rfid security and privacy: A research survey. *IEEE journal on selected areas in communications*, 24(2):381–394, 2006.

[73] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):881–892, 2002.

[74] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems*, 39(1):3, 2014.

[75] Paul Komarek. Komarix datasets, 2018. komarix.org/ac/ds/.

[76] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, and Tim Kraska. Privateclean: Data cleaning and differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 937–951. ACM, 2016.

[77] John Krumm. Inference attacks on location tracks. In *International Conference on Pervasive Computing*, pages 127–143. Springer, 2007.

[78] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[79] Chao Li, Balaji Palanisamy, and James Joshi. Differentially private trajectory analysis for points-of-interest recommendation. In *2017 IEEE International Congress on Big Data*, pages 49–56. IEEE, 2017.

[80] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *The 23rd IEEE International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[81] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):943–956, 2009.

[82] Yibin Li, Wenyun Dai, Zhong Ming, and Meikang Qiu. Privacy protection for preventing data over-collection in smart city. *IEEE Transactions on Computers*, 65(5): 1339–1350, 2015.

[83] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 456–466. Springer, 2005.

[84] Yehida Lindell. Secure multiparty computation for privacy preserving data mining. In *Encyclopedia of Data Warehousing and Mining*, pages 1005–1009. IGI Global, 2005.

[85] Wen Liu, Yong-Bin Wang, Ai-Na Sui, and Min-Yao Ma. Quantum protocol for millionaire problem. *International Journal of Theoretical Physics*, 58(7):2106–2114, 2019.

[86] Xin Liu, Shundong Li, XiuBo Chen, Gang Xu, Xiaolin Zhang, and Yong Zhou. Efficient solutions to two-party and multiparty millionaires' problem. *Security and Communication Networks*, 2017, 2017.

[87] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[88] Zhigang Lu and Hong Shen. Secured privacy preserving data aggregation with semi-honest servers. In *The 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 300–312. Springer, 2017.

[89] Zhigang Lu and Hong Shen. A new lower bound of privacy budget for distributed differential privacy. In *The 18th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 25–32. IEEE, 2017.

[90] Zhigang Lu and Hong Shen. A convergent differentially private k-means clustering algorithm. In *The 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 612–624. Springer, 2019.

[91] Zhigang Lu and Hong Shen. Differentially private k-means clustering with convergence guarantee. *IEEE Transactions on Dependable and Secure Computing*, Early Access, 2020.

[92] Zhigang Lu and Hong Shen. Protect edge privacy in path publishing with differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, submitted:under review, 2020.

[93] Xindi Ma, Jianfeng Ma, Hui Li, Qi Jiang, and Sheng Gao. Agent: an adaptive geo-indistinguishable mechanism for continuous location-based service. *Peer-to-Peer Networking and Applications*, 11(3):473–485, 2018.

[94] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *The 22nd International Conference on Data Engineering*, pages 24–24. IEEE, 2006.

[95] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3, 2007.

[96] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 277–286. IEEE Computer Society, 2008.

[97] Muhammad Adeel Mahmood, Winston KG Seah, and Ian Welch. Reliability in wireless sensor networks: A survey and challenges ahead. *Computer Networks*, 79: 166–187, 2015.

[98] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. Side-channel inference attacks on mobile keypads using smartwatches. *IEEE Transactions on Mobile Computing*, 17(9):2180–2194, 2018.

[99] Frank McSherry. Privacy integrated queries. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 2009.

[100] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *The 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2007.

[101] Prem Melville and Vikas Sindhwani. Recommender systems. *Encyclopedia of Machine Learning and Data Mining*, pages 1056–1066, 2017.

[102] Ilya Mironov. Rényi differential privacy. In *The 30th IEEE Computer Security Foundations Symposium*, pages 263–275. IEEE, 2017.

[103] Noman Mohammed, Rui Chen, Benjamin Fung, and Philip S Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–501. ACM, 2011.

[104] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. Gupt: privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2012.

[105] Elizabeth Armstrong Moore. Man's fitbit helps doctors save his life, April 2016. URL www.usatoday.com/story/tech/nation-now/2016/04/09/fitbit-mans-doctors-save-his-life/82830114/. [Online; accessed 9-April-2016].

[106] Xuejiao Mu, Hong Shen, and Zhigang Lu. Temporal caching-aware dummy location selection for privacy protection in location-based services. In *The 20th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 501–504. IEEE, 2019.

[107] Moni Naor, Benny Pinkas, and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[108] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy*, pages 739–753. IEEE, 2019.

[109] United Nations. The universal declaration of human rights, Dec 1948.

[110] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 644–655. ACM, 2015.

[111] NBN. You have a choice of speeds, 2019. URL https://www.nbnco.com.au/learn/speed. [Online; accessed 21-October-2019].

[112] Thanh Dai Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Privacy aware k-means clustering with high utility. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 388–400. Springer, 2016.

[113] Yiwen Nie, Liusheng Huang, Zongfeng Li, Shaowei Wang, Zhenhua Zhao, Wei Yang, and Xiaorong Lu. Geospatial streams publish with differential privacy. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 152–164. Springer, 2016.

[114] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 75–84. ACM, 2007.

[115] Erfan Nozari, Pavankumar Tallapragada, and Jorge Cortés. Differentially private distributed convex optimization via objective perturbation. In *2016 American Control Conference*, pages 2061–2066. IEEE, 2016.

[116] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[117] Trapti Pandey and Pratha Khare. Bluetooth hacking and its prevention. *L & T Technology Services.*, 2017.

[118] Mijung Park, James Foulds, Kamalika Choudhary, and Max Welling. Dp-em: Differentially private expectation maximization. In *Artificial Intelligence and Statistics*, pages 896–904, 2017.

[119] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.

[120] Jian Pei, Jian Xu, Zhibin Wang, Wei Wang, and Ke Wang. Maintaining k-anonymity against incremental updates. In *The 19th International Conference on Scientific and Statistical Database Management*, pages 5–5. IEEE, 2007.

[121] Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. An efficient and verifiable solution to the millionaire problem. In *International Conference on Information Security and Cryptology*, pages 51–66. Springer, 2004.

[122] Martin Pettai and Peeter Laud. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 421–430. ACM, 2015.

[123] Gina Pingitore, Vikram Rao, Kristen Cavallaro, and Kruttika Dwivedi. To share or not to share – what consumers really think about sharing their personal information, 2017. URL https://www2.deloitte.com/us/en/insights/industry/retail-distribution/sharing-personal-information-consumer-privacy-concerns.html. [Online; accessed 27-October-2019].

[124] Andreas P Plageras, Kostas E Psannis, Christos Stergiou, Haoxiang Wang, and Brij B Gupta. Efficient iot-based sensor big data collection–processing and analysis in smart buildings. *Future Generation Computer Systems*, 82:349–357, 2018.

[125] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *The 29th IEEE International Conference on Data Engineering*, pages 757–768. IEEE, 2013.

[126] Yongrui Qin, Quan Z Sheng, Nickolas JG Falkner, Schahram Dustdar, Hua Wang, and Athanasios V Vasilakos. When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, 64:137–153, 2016.

[127] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In *Conference on the Theory and Application of Cryptology*, pages 628–631. Springer, 1989.

[128] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005(187), 2005.

[129] Charles Rackoff and Daniel R Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Annual International Cryptology Conference*, pages 433–444. Springer, 1991.

[130] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 735–746. ACM, 2010.

[131] Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[132] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[133] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[134] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[135] Muhammad Sajjad, Salman Khan, Tanveer Hussain, Khan Muhammad, Arun Kumar Sangaiah, Aniello Castiglione, Christian Esposito, and Sung Wook Baik. Cnn-based anti-spoofing two-tier multi-factor authentication system. *Pattern Recognition Letters*, 126:123–131, 2019.

[136] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[137] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[138] Dongxu Shao, Kaifeng Jiang, Thomas Kister, Stephane Bressan, and Kian-Lee Tan. Publishing trajectory with differential privacy: A priori vs. a posteriori sampling mechanisms. In *International Conference on Database and Expert Systems Applications*, pages 357–365. Springer, 2013.

[139] Jian Shen, Shaohua Chang, Jun Shen, Qi Liu, and Xingming Sun. A lightweight multi-layer authentication protocol for wireless body area networks. *Future Generation Computer Systems*, 78:956–963, 2018.

[140] Elaine Shi, HTH Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium*. Internet Society., 2011.

[141] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE, 2017.

[142] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[143] Liwei Song, Reza Shokri, and Prateek Mittal. Membership inference attacks against adversarially robust deep learning models. In *2019 IEEE Security and Privacy Workshops*, pages 50–56. IEEE, 2019.

[144] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[145] Cliff R Stanley and D Lawie. Average relative error in geochemical determinations: Clarification, calculation, and a plea for consistency. *Exploration and Mining Geology*, 16(3-4):267–275, 2007.

[146] Jan Stanley and Barry Steinhardt. Bigger monster, weaker chains: The growth of an american surveillance society. In *Ethics and Emerging Technologies*, pages 269–284. Springer, 2014.

[147] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the 6th ACM Conference on Data and Application Security and Privacy*, pages 26–37. ACM, 2016.

[148] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, Min Lyu, and Hongxia Jin. Differentially private k-means clustering and a hybrid approach to private optimization. *ACM Transactions on Privacy and Security*, 20(4):16, 2017.

[149] Yongqiang Sun, Nan Wang, Xiao-Liang Shen, and Jacky Xi Zhang. Location information disclosure in location-based social network services: Privacy calculus, benefit structure, and gender differences. *Computers in Human Behavior*, 52:278–292, 2015.

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

[151] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[152] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.

[153] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Privacy-preserving anonymization of set-valued data. *Proceedings of the VLDB Endowment*, 1(1):115–125, 2008.

[154] Mina Tsay-Vogel, James Shanahan, and Nancy Signorielli. Social media cultivating perceptions of privacy: A 5-year analysis of privacy attitudes and self-disclosure behaviors among facebook users. *New Media & Society*, 20(1):141–161, 2018.

[155] Yao-Tung Tsou, Hung-Li Chen, and Yu-Hsiang Chang. Rod: Evaluating the risk of data disclosure using noise estimation for differential privacy. *IEEE Transactions on Big Data*, 2019.

[156] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[157] Shuo Wang and Richard O Sinnott. Protecting personal trajectories of social media users through differential privacy. *Computers & Security*, 67:142–163, 2017.

[158] Shuo Wang, Richard Sinnott, and Surya Nepal. Privacy-protected social media user trajectories calibration. In *The 12th IEEE International Conference on e-Science*, pages 293–302. IEEE, 2016.

[159] Tien Wang, Trong Danh Duong, and Charlie C Chen. Intention to disclose personal information via mobile applications: A privacy calculus perspective. *International Journal of Information Management*, 36(4):531–542, 2016.

[160] Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, volume 1558, 2016.

[161] John H Wensley, Leslie Lamport, Jack Goldberg, Milton W Green, Karl N Levitt, Po Mo Melliar-Smith, Robert E Shostak, and Charles B Weinstock. Sift: Design and analysis of a fault-tolerant computer for aircraft control. *Proceedings of the IEEE*, 66 (10):1240–1255, 1978.

[162] Wikipedia contributors. Alphago versus lee sedol — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=AlphaGo_versus_Lee_Sedol&oldid=863856305. [Online; accessed 6-November-2018].

[163] Wikipedia contributors. Laplace distribution — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Laplace_distribution&oldid=905647583, 2019. [Online; accessed 24-September-2016].

[164] Wikipedia contributors. List of data breaches — Wikipedia, the free encyclopedia, 2019. URL https://en.wikipedia.org/w/index.php?title=List_of_data_breaches&oldid=921743675. [Online; accessed 23-October-2019].

[165] Wikipedia contributors. Facebook–cambridge analytica data scandal — Wikipedia, the free encyclopedia, 2019. URL https://en.wikipedia.org/w/index.php?title=Facebook%E2%80%93Cambridge_Analytica_data_scandal&oldid=921288198. [Online; accessed 23-October-2019].

[166] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005.

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

[168] World Health Organisation. Number of people (all ages) living with hiv, 2019. URL https://www.who.int/gho/hiv/epidemic_status/cases_all/en/. [Online; accessed 17-Sep-2019].

[169] Belle Selene Xia and Peng Gong. Review of business intelligence through data analysis. *Benchmarking: An International Journal*, 21(2):300–311, 2014.

[170] Xiaokui Xiao and Yufei Tao. m-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 689–700. ACM, 2007.

[171] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

[172] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Workshop on Secure Data Management*, pages 150–168. Springer, 2010.

[173] Yonghui Xiao, Li Xiong, Si Zhang, and Yang Cao. Loclok: Location cloaking with differential privacy via hidden markov model. *Proceedings of the VLDB Endowment*, 10(12):1901–1904, 2017.

[174] Andrew Chi-Chih Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[175] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *The 27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

[176] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: differentially private model fitting using genetic algorithms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 665–676. ACM, 2013.

[177] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997. cs.uef.fi/sipu/datasets/.

[178] Wei-Wei Zhang, Dan Li, Ke-Jia Zhang, and Hui-Juan Zuo. A quantum protocol for millionaire problem with bell states. *Quantum information processing*, 12(6): 2241–2249, 2013.

[179] Xuyun Zhang, Lianyong Qi, Wanchun Dou, Qiang He, Christopher Leckie, Kotagiri Ramamohanarao, and Zoran Salcic. Mrmondrian: Scalable multidimensional anonymisation for big data privacy preservation. *IEEE Transactions on Big Data*, 2017.

[180] Ray Y Zhong, Shulin Lan, Chen Xu, Qingyun Dai, and George Q Huang. Visualization of rfid-enabled shopfloor logistics big data in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 84(1-4):5–16, 2016.