Toward Privacy in High-Dimensional Data Publishing

Rui Chen

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

September 2012

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Rui Chen

Entitled: Toward Privacy in High-Dimensional Data Publishing

and submitted in partial fulfillment of the requirements for the degree of

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. A. Kishk — Chair

Dr. D. Lee — External Examiner

Dr. P. Valizadeh — External to Program

Dr. C. Constantinides — Examiner

Dr. R. Witte — Examiner

Drs. B. C. Desai and B. C. M. Fung — Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

September, 2012

Dean of Faculty

ABSTRACT

Toward Privacy in High-Dimensional Data Publishing

Rui Chen, Ph.D.
Concordia University, 2012

Nowadays data sharing among multiple parties has become inevitable in various application domains for diverse reasons, such as decision support, policy development and data mining. Yet, data in its raw format often contains person-specific sensitive information, and publishing such data without proper protection may jeopardize individual privacy. This fact has spawned extensive research on *privacy-preserving data publishing* (PPDP), which balances the fundamental trade-off between individual privacy and the utility of published data. Early research of PPDP focuses on protecting private and sensitive information in relational and statistical data. However, the recent prevalence of several emerging types of *high-dimensional* data has rendered unique challenges that prevent traditional PPDP techniques from being directly used. In this thesis, we address the privacy concerns in publishing four types of high-dimensional data, namely *set-valued data*, *trajectory data*, *sequential data* and *network data*. We develop effective and efficient non-interactive data publishing solutions for various utility requirements. Most of our solutions satisfy a rigorous privacy guarantee known as *differential privacy*, which has been the de facto standard for privacy protection. This thesis demonstrates that our solutions have exhibited great promise for releasing useful high-dimensional data without endangering individual privacy.

Dedication

To my family with love

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisors, Dr. Bipin C. Desai and Dr. Benjamin C. M. Fung, for their constant guidance, support, care, understanding and patience. The Ph.D. program is a long journey. Without their continuous encouragement, I might have quit halfway. I could not have asked for better advisors than them. I am greatly indebted to Dr. Claude Castelluccia for giving me the invaluable research internship opportunity at INRIA and helping me shape my future career. His guidance has substantially contributed to this thesis.

I would like to express my sincere gratitude to my research collaborators, Dr. Gergely Arcs, Dr. Li Xiong, Dr. Xiaoqian Jiang and Dr. Philip S. Yu. I owe my sincere gratitude to my colleagues, Dr. Noman Mohammed, Mr. Wenming Liu, Mr. Ming Lu and Mr. Kunsheng Zhao. In the past four years, I greatly benefited from their helpful suggestions, great research ideas, and effective discussions. Thanks also to my entire thesis committee: Dr. Constantinos Constantinides, Dr. René Witte, Dr. Pouya Valizadeh and Dr. Dongwon Lee.

I am thankful for all faculty members and staff at Concordia University. Special thanks to Mr. Steven Shi and Ms. Pauline Dubois for providing me valuable work opportunities, which allow me to support my family during the program. I am also grateful to Ms. Halina Monkiewicz for her administrative support.

Thanks to my beloved parents. Their insightful, selfless decision of sending their only child to Canada is the very first step of this thesis. Most of all, thanks to my wife, Yuya, for her love and faith. Her courage to go abroad to be with me is the greatest encouragement to accomplish my degree.

I acknowledge financial support from Concordia University and the NSERC Canada Graduate Scholarship, which made this thesis possible.

TABLE OF CONTENTS

FIGURES

TABLES

Chapter 1

Introduction

With the current trend of digitalization, individual privacy is more subject to intrusions than ever before. Various personal information is being collected in different application domains, for example, retailing business, healthcare departments, public transit agencies, and online social networks. Such personal information in its raw format often contains person-specific sensitive information. Therefore, improper information sharing among different parties may jeopardize individual privacy. This is exemplified by several real-life privacy incidents given below.

Massachusetts voter list. Sweeney [107] successfully re-identified the former governor of Massachusetts by linking a public voter list with a medical database published by the Massachusetts Group Insurance Commission through the combination of zip code, date of birth and gender (called the *quasi-identifier* [33]). Sweeney [107] further indicated that for 87% of the U.S. population such characteristics had been recorded and available, which would likely make them identifiable based on only such quasi-identifiers.

AOL search queries. On August 4, 2006, AOL published approximately 20 million search queries collected from 650,000 users over a 3-month period. User identity information (e.g., such as name or SSN) had been replaced by some pseudonymous user IDs before release, but three days later, it had to remove the release due to the

re-identification of a user [56].

Netflix prize data. Netflix, the largest on-demand Internet streaming media service provider, released the anonymized movie ratings of 500,000 subscribers for a contest with the purpose of improving the accuracy of its recommendation system. However, Netflix had to cancel the contest because Narayanan and Shmatikov [94] revealed that when combining with the information in the public Internet Movie Database (IMDb), users in the released Netflix dataset could be re-identified with high probability.

These real-world privacy concerns have stimulated strong demands for privacy protection in data sharing. The current practice of information sharing primarily relies on policies and guidelines on the types of data that can be released and agreements on the proper use of published data. For example, the *Health Insurance Portability and Accountability Act* (HIPAA) [96] has become the standard privacy rule for medical data sharing. However, this approach alone may lead to either excessive data distortion or insufficient privacy protection. Comprehensive technological solutions are indispensable for providing formal, provable privacy guarantees. Consequently, extensive research has been conducted on *privacy-preserving data publishing* (PPDP) [46] with the goal of publishing useful data while protecting individual privacy even in a hostile environment. The essential trade-off between data utility and individual privacy forms the foundation of PPDP.

Early works on PPDP focus on protecting private and sensitive information in relational data, which is of a fixed schema with a small number of dimensions. In the context of relational data, various traditional privacy models (referred to as *partition-based* privacy models [49] in the sequel), such as k-anonymity [104], [108], ℓ-diversity [85] and confidence bounding [114], and many anonymization approaches have been proposed. These efforts have successfully shown their strength of privacy protection in publishing relational data, and have also raised expectations of effective privacy-preserving techniques for more complex data types.

In recent years, several emerging types of *high-dimensional* data, including *set-valued data*, *trajectory data*, *sequential data* and *network data*, have become prevalent. While they have become important sources of data analysis, they have simultaneously posed novel technical challenges that prevent traditional PPDP approaches from being directly used. The solutions developed in the context of relational data are useful for thwarting identity linkage and attribute linkage privacy attacks [46] in the case of small numbers of dimensions, however, they cannot be used effectively in the high dimensional case. Aggarwal [3] pointed out that with the increase of dimensionality one has to face with a choice of either undesirable data utility or insufficient privacy protection, known as the *curse of high dimensionality*. From the perspective of data utility, increasing dimensionality requires more information to be suppressed, rendering the released data useless; from the perspective of privacy, increasing dimensionality makes each record more distinctive from others, leaving the target victim easier to identify and therefore harder to protect. Furthermore, high dimensionality naturally poses challenges on computational complexity. Therefore, more effective and efficient solutions must be developed so that real-life high-dimensional data could be successfully handled.

In this thesis, we concentrate on developing practical technical solutions for these types of high-dimensional data under rigorous privacy models while providing meaningful data utility for various data analysis tasks. In addressing privacy concerns in high-dimensional data publishing, the first effort is to identify an appropriate privacy model. Recently, new types of privacy attacks, such as *composition attack* [49], *deFinetti attack* [67] and *foreground knowledge attack* [118], have been identified on the approaches derived using partition-based privacy models, demonstrating their vulnerability to an adversary's *background knowledge*. Due to the deterministic nature of partition-based privacy models, it is foreseeable that more types of privacy attacks could be discovered on these privacy models in the future.

Consequently, over the last few years *differential privacy* [37] has become the

de facto successor to partition-based privacy models. Differential privacy, stemming from the field of *statistical disclosure control*, provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. It, in general, requires that the outcome of any analysis should not overly depend on a single data record. It follows that even if a user had opted to be included in the database, there would not be a significant change in any computation based on the database. Therefore, this assures every record owner that any privacy breach will not be a result of participating in a database.

However, the strong privacy guarantee provided by differential privacy does not come without cost. There are two natural settings of data sanitization under differential privacy: *interactive* and *non-interactive*. In the interactive setting, a sanitization mechanism sits between the users and the database. Queries posed by the users and/or their responses must be evaluated and may be modified by the mechanism in order to protect privacy; in the non-interactive setting, a data publisher computes and releases a sanitized version of a database, possibly a synthetic database, to the public and hence it could be used for any analysis. There have been some lower bound results [34], [37], [38] of differential privacy, indicating that only a limited number of queries could be answered; otherwise, an adversary would be able to precisely reconstruct almost the entire original database, resulting in a serious compromise of privacy. Therefore, most recent works have concentrated on designing various interactive mechanisms that answer only a sublinear number, in the size of the underlying database, of queries *in total*, regardless of the number of users. Once this limit is reached, either the database has to be shut down, or any further query would be rejected. This limitation has greatly hindered their applicability, especially in the scenario where a database is made available to many users who legitimately need to pose a large number of queries. Naturally, one would favor a non-interactive release that could be used to answer an arbitrary large number of queries or for various data analysis tasks.

Blum et al. [12] pointed out that the aforementioned lower bounds could be circumvented in the non-interactive setting at the cost of preserving usefulness for *only* restricted classes of queries. However, they did not provide an efficient algorithm. A series of subsequent works [39], [125], [126] aimed to propose more efficient non-interactive sanitization mechanisms. However, all these works are of runtime complexity *at least* linear in the *output domain size*. These progresses, however, are still not sufficient to handle high-dimensional data, because the output domain sizes of high-dimensional data are typically *exponentially* large. To tackle this technical challenge, in this thesis, we initiate a line of *data-dependent* solutions, which adaptively narrows down the output domain by using noisy information obtained from the underlying database. These data-dependent solutions not only achieve reasonable runtime complexity (e.g., linear in the input data size) but also have a positive impact on the resulting utility as there is no need to add noise to every possible entry in the output domain, which accumulates noise quickly.

Protecting individual privacy is one aspect of the problem of PPDP. Another equally important aspect is preserving utility in sanitized data for data analysis. In this thesis, we preserve data utility of different data types for various data analysis tasks, ranging from more general tasks, such as counting query and cut query, to more concrete tasks, such as frequent sequential pattern mining and frequent itemset mining. We theoretically and experimentally demonstrate that useful high-dimensional data could be released even under rigorous privacy models.

1.1 Contributions

In this thesis, we study the problem of privacy-preserving data publishing over four emerging types of high-dimensional data, namely *set-valued data*, *trajectory data*, *sequential data* and *network data*. The key contributions of this thesis are summarized below.

1.1.1 Set-Valued Data Sanitization

Set-valued data, such as transaction data, web search queries, and click streams, has become a major source for various data mining tasks. All existing sanitization techniques [17], [50], [62], [110], [111], [128], [129] developed for publishing set-valued data are dedicated to partition-based privacy models, which are vulnerable to privacy attacks based on background knowledge. In contrast, differential privacy provides strong privacy guarantees independent of an adversary's background knowledge, computational power or subsequent behavior. Existing data publishing approaches for differential privacy, however, are not adequate in terms of both utility and scalability in the context of set-valued data due to its high dimensionality.

This thesis is the first study of publishing set-valued data via differential privacy. Our work initiates the line of *data-dependent* solutions for achieving differential privacy, which allows differential privacy to be efficiently and effectively applied to different types of data. In particular, we propose a probabilistic top-down partitioning algorithm to generate a differentially private release for set-valued data, which scales linearly with the input data size. We prove that our result is (δ, β)-useful for the class of counting queries, the foundation of many data mining tasks. We show that our approach maintains high utility for counting queries and frequent itemset mining and scales to large datasets through extensive experiments on different real-life set-valued datasets.

1.1.2 Trajectory Data Sanitization

With the increasing prevalence of location-aware devices, trajectory data has been generated and collected in various application domains. Trajectory data carries rich information that is useful for many data analysis tasks. Yet, improper publishing and use of trajectory data could jeopardize individual privacy. In this thesis, we acknowledge the emerging data publishing scenario, in which trajectory data needs to

be published with sensitive attributes, and consequently propose the $(K, C)_L$-*privacy* model, which takes into consideration not only identity linkage attacks on trajectory data, but also attribute linkage attacks via trajectory data. This is the first work to introduce *local suppression* to trajectory data sanitization. Our framework allows the adoption of various data utility metrics for different data mining tasks. As an illustration, we aim at preserving both instances of location-time pairs and frequent sequences in a trajectory database, both being the foundation of many trajectory data mining tasks. Our experiments on both synthetic and real-life datasets suggest that the framework is both effective and efficient to overcome the challenges in trajectory data sanitization. In particular, compared with the previous works in the literature, our proposed local suppression method can significantly improve the data utility in sanitized trajectory data.

1.1.3 Sequential Data Sanitization

As a simplified form of trajectory data, sequential data is being increasingly used in a variety of applications, spanning from genome and web usage analysis to location-based recommendation systems. Publishing sequential data is important, since it enables researchers to analyze and understand interesting patterns. In particular, we are motivated by the data sharing scenario at the *Société de transport de Montréal* (STM), the public transit agency in Montreal area. In this thesis, we propose two alternative solutions for publishing sequential data under the rigorous *differential privacy* model.

Publishing sequential data via prefix tree. We propose an efficient data-dependent yet differentially private sequential data sanitization approach based on a *hybrid-granularity* prefix tree structure. Moreover, as a post-processing step, we make use of the inherent consistency constraints of a prefix tree to conduct constrained inferences, which lead to better utility. To our best knowledge, this is the

first work to introduce a practical solution for publishing large volume of sequential data under differential privacy. We examine data utility in terms of two popular data analysis tasks conducted at the STM, namely counting queries and frequent sequential pattern mining. Extensive experiments on real-life STM datasets confirm that our approach maintains high utility and is scalable to large datasets.

Publishing sequential data via n-grams. Due to its inherent sequentiality and high-dimensionality, it is challenging to apply differential privacy to sequential data. As an alternative, we address this challenge by employing a *variable-length n-gram model*, which extracts the essential information of a sequential database in terms of a set of variable-length n-grams. Our approach makes use of a carefully designed exploration tree structure and a set of novel techniques based on the Markov assumption in order to lower the magnitude of added noise. The published n-grams are useful for many purposes. Furthermore, we develop a solution for generating a synthetic database, which enables a wider spectrum of data analysis tasks. Extensive experiments on real-life datasets demonstrate that our approach substantially outperforms the state-of-the-art techniques.

1.1.4 Network Data Sanitization

With the increasing popularity of information networks, research on privacy-preserving network data publication has received substantial attention recently. Most existing works focus on preventing node re-identification from adversaries with structural background knowledge. In contrast, research on thwarting *edge disclosure* (e.g., inferring if there is a direct link between two individuals) is less fruitful, largely due to lack of a formal privacy model. The recent emergence of *ϵ-differential privacy* has shown great promise for rigorous edge disclosure protection. Yet ϵ-differential privacy is vulnerable to data correlation, which hinders its application to network data that may be inherently correlated.

In this thesis, we propose a stronger variant of ϵ-differential privacy, known as (ϵ, k)-differential privacy, which provides privacy guarantee even when a record is correlated to at most $k-1$ other records. We present the concept of correlated sensitivity, which allows Laplace mechanism and exponential mechanism to be used for achieving (ϵ, k)-differential privacy. We subsequently provide a holistic solution for *non-interactive* network data publication. The basic idea is to adaptively identify dense regions of the adjacency matrix of a network dataset by a data-dependent partitioning process, and then reconstruct a noisy adjacency matrix by a novel use of exponential mechanism, which is of independent interest. To our best knowledge, this is the first work providing an efficient and effective solution for publishing real-life network data in the spirit of differential privacy. Extensive experiments demonstrate that our approach performs well on different types of real-life network datasets.

1.2 Organization of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 introduces four important privacy models, the common anonymization mechanisms and popular utility metrics.
- Chapter 3 provides an in-depth literature review of the state-of-the-art techniques in PPDP for different data types. In particular, we summarize the recent applications of differential privacy. Our survey of recent developments of privacy-preserving data publishing has been published in [46].
- Chapter 4 studies the problem of publishing set-valued data for data mining tasks under the differential privacy model. We propose a probabilistic top-down partitioning algorithm that is scalable to high-dimensional set-valued data while providing guaranteed utility. The results of this chapter have been

published in [25].

- Chapter 5 studies the problem of privacy-preserving trajectory data publishing under a realistic heterogeneous data publishing scenario. We develop a generic sanitization framework for trajectory data, which accommodates various utility metrics. The results of this chapter have been published in [23].

- Chapter 6 proposes two alternative solutions to publishing sequential data under differential privacy. The results of this chapter have been published in [22], [20].

- Chapter 7 presents a stronger variant of ϵ-differential privacy, known as (ϵ, k)-differential privacy, for correlated data, and provides a holistic solution for non-interactive network data publication under (ϵ, k)-differential privacy. The results of this chapter are currently under review in [24].

- Chapter 8 concludes the thesis.

Chapter 2

Background

In privacy-preserving data publishing, there is a fundamental trade-off between privacy and utility [66]. At one extreme, the data holder may publish nothing so that privacy can be perfectly protected, but the resulting data utility is zero. At the other extreme, the data holder may directly publish the raw data without any anonymization attempt so that the data utility can be maximized, but no privacy protection can be guaranteed. Thus, it is of importance for the data holder to find a reasonable trade-off between privacy and utility. This requires the following concepts to be defined: privacy model, anonymization mechanism, and utility metric.

2.1 Privacy Models

In 1977, Dalenius [32] provided a very stringent definition of privacy protection: "access to the published data should not enable the attacker to learn anything extra about any target victim compared to no access to the database, even with the presence of any attacker's background knowledge obtained from other sources". In real-life applications, such an absolute privacy protection is impossible due to the presence of an attacker's background knowledge [36]. For this reason, most literature on privacy-preserving data publishing considers more relaxed, but more

practical notions of privacy protection by assuming that an attacker has limited background knowledge. With his background knowledge, the attacker can perform different kinds of privacy attacks. Accordingly, different kinds of privacy models have been proposed. In this section, we will focus on four most fundamental privacy models, namely, k-anonymity [104], [108], ℓ-diversity [85], confidence bounding [114] and differential privacy [37].

In general, a privacy threat occurs when an attacker is able to link a record owner to a record in a published data table, to a sensitive attribute in a published data table, or to the published data table itself. We call these *identity linkage*, *attribute linkage*, and *membership linkage*, respectively. In the most basic form of PPDP, the data holder has a table of the form

$$D(Explicit\_Identifier, Quasi\_Identifier, Sensitive\_Attributes, Non{-}Sensitive\_Attributes),$$

where $Explicit\_Identifier$ is a set of attributes containing information that explicitly identifies record owners such as name and social security number (SSN); $Quasi\_Identifier$ (QID) is a set of attributes that could potentially identify record owners; $Sensitive\_Attributes$ consist of sensitive person-specific information such as disease, salary, and disability status; and $Non - Sensitive\_Attributes$ contain all attributes that do not fall into the previous three categories [16]. The four sets of attributes are disjoint. Furthermore, most works assume that each record in the table belongs to a distinct record owner, known as *microdata*. Previous works have indicated that simply removing explicit identifiers is insufficient to protect individual privacy. Based on quasi-identifiers, an attacker is still able to perform several types of privacy attacks, as explained below.

Identity linkage attack. In an identity linkage attack, some value *qid* on QID identifies a small number of records in the released table T, called a *group*. If the target victim's QID matches the value *qid*, the victim is vulnerable to being linked

Table 2.1: Raw patient data

Job	**Sex**	**Age**	**Disease**
Engineer	Male	35	Hepatitis
Engineer	Male	38	Hepatitis
Lawyer	Male	38	HIV
Writer	Female	30	Flu
Writer	Female	30	HIV
Dancer	Female	30	HIV
Dancer	Female	30	HIV

to the small number of records in the group. In this case, the adversary faces only a small number of possibilities for the victim's record, and with the help of background knowledge, there is a chance that the adversary could uniquely identify the victim's record from the group.

Example 2.1.1. Suppose that a hospital wants to publish patients' records in Table 2.1 to a research center. The explicit identifiers have been removed. If an adversary knows that the record of Bob, a male lawyer who is 38 years old, is in the table, he can infer that Bob is infected with HIV because there is only one record with $qid = \langle Lawyer, Male, 38 \rangle$. ■

Attribute linkage attack. In an attribute linkage attack, the adversary may not need to precisely identify the record of the target victim, but could still infer his sensitive values from the published dataset T based on the set of sensitive values associated with the group to which the victim belongs. In case some sensitive values predominate in the group, a successful inference becomes relatively easy.

Example 2.1.2. From Table 2.1, an adversary can observe that the sensitive attribute of all records with $qid = \langle Dancer, Female, 30 \rangle$ is of the same value, HIV. Therefore, the adversary can easily infer that the target victim Emily, a 30-year-old female dancer, has HIV with 100% confidence provided that he knows that Emily's record is in Table 2.1, even though he cannot uniquely identify her record. ■

Table 2.2: 3-anonymous patient data by generalization

Job	Sex	Age	Disease
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	Hepatitis
Professional	Male	[35-40)	HIV
Artist	Female	[30-35)	Flu
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV
Artist	Female	[30-35)	HIV

Membership linkage attack. Both identity linkage and attribute linkage assume that an adversary already assures that the victim's record is in the released table T. However, in some cases, the presence (or the absence) of the victim's record in T already reveals the victim's sensitive information. Suppose a hospital releases a data table with a particular type of disease. Identifying the presence of the victim's record in the table is already damaging. A membership linkage occurs if an adversary can confidently infer the presence or the absence of the victim's record in the released table.

2.1.1 k-Anonymity

To prevent identity linkage attacks through QID, Samarati and Sweeney [104], [108] proposed the notion of k-anonymity: if a record in the table has some value qid, then at least $k-1$ other records should also have the value qid. In other words, the minimum group size on QID is at least k. A table satisfying this requirement is called k-*anonymous*. In a k-anonymous table, each record is indistinguishable from at least $k-1$ other records with respect to QID. Consequently, the probability of linking a victim to a specific record through QID is at most $1/k$.

Example 2.1.3. Table 2.2 shows a 3-anonymous table by generalizing $QID = \{Job, Sex, Age\}$ from Table 2.1 using the taxonomy trees in Figure 2.1. It has two distinct QID groups: $\langle Professional, Male, [35-40)\rangle$ and $\langle Artist, Female, [30-$

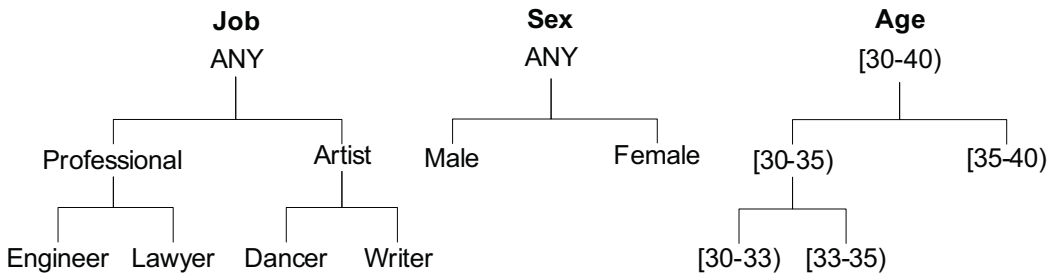


Figure 2.1: Taxonomy trees for Job, Sex, Age.

35)⟩. Since each group contains at least 3 records, the table is 3-anonymous. ■

The k-anonymity model assumes that QID is known to the data publisher. Most works consider a single QID containing all attributes that can be potentially used to launch a privacy attack. The more attributes included in QID, the more protection k-anonymity would provide. On the other hand, this also implies that more information distortion is needed to achieve k-anonymity because the records in a group have to agree on more attributes.

2.1.2 ℓ-Diversity

k-anonymity protects identity linkage attacks, but fails to prevent attribute linkage attacks. Consequently, Machanavajjhala et al. [85] proposed the diversity principle, called *ℓ-diversity*, to thwart attribute linkage attacks. The ℓ-diversity model requires every *qid* group to contain at least ℓ "well-represented" sensitive values. Based on different interpretations of well-representedness, there are several instantiations of this principle. The simplest understanding of "well-represented" is to ensure that there are at least ℓ distinct values for the sensitive attribute in each *qid* group. This *distinct ℓ-diversity* privacy model automatically satisfies k-anonymity, where $k = \ell$, because each *qid* group contains at least ℓ records. Distinct ℓ-diversity cannot prevent probabilistic inference attacks because some sensitive values are naturally more frequent than others in a group, enabling an adversary to conclude that a record in the group is very likely to have those values. For example, Flu is more

common than HIV. This motivates the following two stronger notions of ℓ-diversity.

A table is *entropy ℓ-diverse* if for every *qid* group

$$-\sum_{s \in S} P(qid, s) \log(P(qid, s)) \geq \log(\ell) \tag{2.1}$$

where S is a sensitive attribute, $P(qid, s)$ is the fraction of records in a *qid* group having the sensitive value s. The left-hand side, called the entropy of the sensitive attribute, has the property that more evenly distributed sensitive values in a *qid* group produce a larger value. Therefore, a larger threshold value ℓ implies less certainty of inferring a particular sensitive value in a group. Note that the inequality does not depend on the choice of the log base.

Example 2.1.4. Consider Table 2.2. For the first group $\langle Professional, Male, [35-40)\rangle$, $-\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} = \log(1.9)$, and for the second group $\langle Artist, Female, [30-35)\rangle$, $-\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = \log(1.8)$. So the table satisfies entropy ℓ-diversity if $\ell \leq 1.8$. ■

One limitation of entropy ℓ-diversity is that it does not provide a probability based risk measure, which tends to be more intuitive to a human data publisher. For example, being entropy 1.8-diverse in Example 2.1.4 does not convey the risk level that the attacker has 75% probability of succeeding in inferring the sensitive value HIV where 3 out of the 4 record owners in the qid group have HIV. Also, it is difficult to specify different protection levels based on varying sensitivity and frequency of sensitive values.

The *recursive (c, ℓ)-diversity* makes sure that the most frequent value does not appear too frequently, and the less frequent values do not appear too rarely. Let m be the number of sensitive values in a *qid* group. Let f_i denote the frequency of the i^{th} most frequent sensitive value in a *qid* group. A *qid* group is $(c, \ell) - diverse$ if the frequency of the most frequent sensitive value is less than the sum of the frequencies of the $m-\ell+1$ least frequent sensitive values multiplying by some publisher-specified

constant c, i.e., $f_1 < c\sum_{i=\ell}^{m} f_i$. A table is said to have recursive (c, ℓ)-diversity if all of its groups have (c, ℓ)-diversity. Similarly, the recursive (c, ℓ)-diversity is not intuitive for a data publisher to specify the desired level of privacy protection.

2.1.3 Confidence Bounding

Wang et al. [114] proposed an alternative privacy model to protect attribute linkage attacks. They considered bounding the confidence of inferring a sensitive value from a *qid* group by specifying one or more *privacy templates* of the form, $\langle QID \rightarrow s, h\rangle$, where s is a sensitive value, QID is a quasi-identifier, and h is a threshold. Let $Conf(QID \rightarrow s)$ be $maxconf(qid \rightarrow s)$ over all *qid* groups on QID, where $conf(qid \rightarrow s)$ denotes the percentage of records containing s in the *qid* group. A table satisfies $\langle QID \rightarrow s, h\rangle$ if $Conf(QID \rightarrow s) \leq h$. In other words, $\langle QID \rightarrow s, h\rangle$ bounds the attacker's confidence of inferring the sensitive value s in any group on QID to at most h.

For example, with $QID = \{Job, Sex, Age\}$, $\langle QID \rightarrow HIV, 10\%\rangle$ states that the confidence of inferring HIV from any group on QID is no more than 10%. For the data in Table 2.2, this privacy template is violated because the confidence of inferring HIV is 75% in the group $\langle Artist, Female, [30-35)\rangle$.

The confidence measure has two advantages over recursive (c, ℓ)-diversity and entropy ℓ-diversity. First, the confidence measure is more intuitive because the risk is measured by the probability of inferring a sensitive value. The data publisher relies on this intuition to specify the acceptable maximum confidence threshold. Second, it allows the flexibility for the data publisher to specify a different threshold h for each combination of QID and s according to the perceived sensitivity of inferring s from a group on QID. The recursive (c, ℓ)-diversity cannot be used to bound the frequency of sensitive values that are not the most frequent. Confidence bounding provides greater flexibility than ℓ-diversity in this aspect.

2.1.4 ϵ-Differential Privacy

All aforementioned privacy models are called *partition-based* models. They provide privacy protection by enforcing certain *syntactic* requirements on the released data. Recent research indicates that partition-based privacy models are vulnerable to an adversary's background knowledge. In contrast, *differential privacy* [37] is a more *semantic* definition, which provides strong privacy guarantees independent of an adversary's background knowledge. Differential privacy requires that the removal or addition of a single database record does not significantly affect the outcome of any analysis. It ensures a data record owner that any privacy breach will not be a result of participating in the database since anything that is learnable from the database with his record is also learnable from the one without his record. Formally, differential privacy [37] is defined as follow. Here the parameter, ϵ, specifies the degree of privacy offered.

Definition 2.1 (ϵ-differential privacy)**.** A privacy mechanism $\mathcal{A}$ gives ϵ-differential privacy if for any dataset $\mathcal{D}_1$ and $\mathcal{D}_2$ differing on at most one record, and for any possible sanitized dataset $\widetilde{\mathcal{D}} \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(\mathcal{D}_1) = \widetilde{\mathcal{D}}] \leq e^{\epsilon} \times Pr[\mathcal{A}(\mathcal{D}_2) = \widetilde{\mathcal{D}}] \tag{2.2}$$

where the probability is taken over the randomness of $\mathcal{A}$. ■

One salient merit of differential privacy is its composition properties, which provide privacy guarantees in case of sequential release. Any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence, which is known as *sequential composition* [87]. The implication is that differential privacy is robust to collusions among adversaries.

Theorem 2.1. *Let $\mathcal{A}_i$ each provide ϵ_i-differential privacy. A sequence of $\mathcal{A}_i(\mathcal{D})$ over the dataset $\mathcal{D}$ provides $(\sum_i \epsilon_i)$-differential privacy.* ■

In some special cases, in which a sequence of computations is conducted on *disjoint* datasets, the privacy cost does not accumulate, but depends only on the worst guarantee of all computations. This is known as *parallel composition* [87]. This property could and should be used to obtain good performance.

Theorem 2.2. *Let $\mathcal{A}_i$ each provide ϵ_i-differential privacy. A sequence of $\mathcal{A}_i(\mathcal{D}_i)$ over a set of disjoint datasets $\mathcal{D}_i$ provides $(max(\epsilon_i))$-differential privacy.* ■

2.2 Anonymization Mechanisms

Normally, a given raw dataset is very unlikely to satisfy a specified privacy model. Certain anonymization mechanisms need to be applied to the raw dataset, making it less precise, in order to achieve the privacy model. This is usually done by applying a sequence of anonymization operations, which naturally leads to the trade-off between privacy and utility. It is worth mentioning that there may exist more than one anonymization mechanism to achieve a specific privacy model. However, in many cases, it is important to choose a right anonymization mechanism in order to obtain a better trade-off. So far, four kinds of anonymization mechanisms have been widely used, namely *generalization*, *suppression*, *bucketization* and *perturbation*.

2.2.1 Generalization

The generalization mechanism generates anonymous releases by replacing some attribute values by more general values. For a categorical attribute, a specific value can be replaced with a general one according to a given taxonomy; for a numerical attribute, exact values can be replaced with an interval that covers the exact values. Usually, no-predetermined taxonomy is given for a numerical attribute. We have seen taxonomy trees for both categorical and numerical attributes in Figure 2.1. In Example 2.1.3, we achieved 3-anonymity by generalizing QID according to the taxonomy trees in Figure 2.1.

Generalization can be performed using either *global recoding scheme* or *local recoding scheme*. The global recoding scheme further includes *full-domain generalization scheme*, *subtree generalization scheme* and *sibling generalization scheme* as explained below.

Full-domain generalization scheme [74], [104], [108]. In this scheme, all values of an attribute are generalized to the same level of the taxonomy tree. For example, in Figure 2.1, if *Lawyer* and *Engineer* are generalized to *Professional*, then it also requires generalizing *Dancer* and *Writer* to *Artist*. The search space for this scheme is much smaller than the search spaces for the other schemes below, but the data distortion is the largest because of the same granularity level requirement on all paths of a taxonomy tree.

Subtree generalization scheme [9], [47], [48], [64], [115]. In this scheme, at a non-leaf node, either all child values or none are generalized. For example, in Figure 2.1, if *Engineer* is generalized to $Professional$, this scheme also requires the other child node, *Lawyer*, to be generalized to $Professional$, but *Dancer* and $Writer$, which are child nodes of *Artist*, can remain ungeneralized.

Sibling generalization scheme [74]. This scheme is similar to the subtree generalization, except that some siblings may remain ungeneralized. A parent value is then interpreted as representing all missing child values. For example, in Figure 2.1, if *Engineer* is generalized to $Professional$, and *Lawyer* remains ungeneralized, $Professional$ is interpreted as all jobs covered by $Professional$ except for *Lawyer*. This scheme produces less distortion than subtree generalization scheme because it only needs to generalize the child nodes that violate the specified threshold.

Cell generalization scheme [74], [120], [127]. In all of the above schemes, if a value is generalized, all its instances in the raw dataset are generalized. Therefore, such schemes are called *global recoding*. In cell generalization, also known as *local recoding*, some instances of a value may remain ungeneralized while other instances

Table 2.3: 2-anonymous patient data by suppression

Job	**Sex**	**Age**	**Disease**
*	Male	*	Hepatitis
*	Male	*	Hepatitis
*	Male	*	HIV
Writer	Female	30	Flu
Writer	Female	30	HIV
Dancer	Female	30	HIV
Dancer	Female	30	HIV

are generalized. For example, in Table 2.1 *Engineer* in the first record is generalized to $Professional$, while *Engineer* in the second record can remain ungeneralized. Compared with global recoding schemes, this scheme is more flexible; therefore, it produces a smaller data distortion. Nonetheless, it is important to note that the utility of data could be adversely affected by this flexibility, which causes a data exploration problem: most standard data mining methods treat *Engineer* and $Professional$ as two independent values, but, in fact, they are not. For example, building a decision tree from such a generalized table may result in two branches, $Professional \rightarrow class1$ and $Engineer \rightarrow class2$. It is unclear which branch should be used to classify a new engineer. Though very important, this aspect of data utility has been ignored by all works that employed the local recoding scheme. Data produced by global recoding does not suffer from this data exploration problem.

2.2.2 Suppression

Suppression is a straightforward anonymization mechanism. Unlike generalization, the suppression mechanism does not require any given taxonomy. It produces a release candidate by replacing some attribute values by a special symbol (e.g., "*" or "Any"), which indicates that the value has been suppressed. Table 2.3 presents a 2-anonymous table from Table 2.1 using suppression, in which certain values are

Table 2.4: Bucketized patient data

Job	Sex	Age	BID
Engineer	Male	35	1
Engineer	Male	38	1
Lawyer	Male	38	1
Writer	Female	30	2
Writer	Female	30	2
Dancer	Female	30	2
Dancer	Female	30	2

BID	Disease
1	HIV
1	Hepatitis
1	Hepatitis
2	HIV
2	Flu
2	HIV
2	HIV

replaced by wildcard values, "*". Analogous to generalization, there are also different schemes for suppression. *Record suppression* [9], [64], [74], [104] refers to the operation that suppresses an entire record. *Value suppression* [113], [114] refers to the operation that suppresses every instance of a given value in a table. *Cell suppression* (or *local suppression*) [31], [92] refers to the operation that suppresses some instances of a given value in a table. Cell suppression results in less data distortion, but usually requires greater computational complexity.

2.2.3 Bucketization

The basic idea of the bucketization mechanism is to break the correlation between quasi-identifiers and sensitive values. It first partitions the records in the original data table into *non-overlapping* buckets, each of which is assigned a unique BID. For each bucket, it randomly permutes the sensitive attribute values, and then publishes its projection on the quasi-identifier attributes and also its projection on the permuted sensitive attribute. Table 2.4 presents a release candidate of the bucketization mechanism from Table 2.1. After bucketization, the sensitive value of a victim becomes indistinguishable from the rest in the same bucket.

The main limitation of this mechanism is that its application requires clearly defined sensitive attributes and non-overlapping buckets. This requirement prevents it from being applied to certain types of data, for example, set-valued data and

trajectory data. Moreover, since the quasi-identifiers are published without any modification, an adversary is likely to be able to perform an identity linkage by joining some external tables.

2.2.4 Perturbation

Perturbation mechanisms have been used for a long period of time in the field of statistical disclosure control. Adam and Wortmann [2] have provided a complete summary of perturbation mechanisms that have been widely employed. In this section, we focus on two standard perturbation mechanisms that are used for achieving differential privacy, namely *Laplace mechanism* and *exponential mechanism*. A fundamental concept of both mechanisms is the *global sensitivity* of a function [37] that maps underlying datasets to (vectors of) reals.

Definition 2.2 (Global Sensitivity)**.** For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$GS(f) = \max_{\mathcal{D}_1, \mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_1 \tag{2.3}$$

for all $\mathcal{D}_1, \mathcal{D}_2$ differing in at most one record. ■

Roughly speaking, functions with lower sensitivity are more tolerant towards changes of a dataset and, therefore, allow more accurate differentially private mechanisms.

Laplace Mechanism

For the analysis whose outputs are real, a standard mechanism to achieve differential privacy is to add Laplace noise to the true output of a function. Dwork et al. [37] propose the Laplace mechanism which takes as inputs a database $\mathcal{D}$, a function f, and the privacy parameter ϵ. The noise is generated according to a Laplace

distribution with the probability density function (pdf) $p(x|\lambda) = \frac{1}{2\lambda}e^{-|x|/\lambda}$, where λ is determined by both $GS(f)$ and the desired privacy parameter ϵ.

Theorem 2.3. *For any function $f : \mathcal{D} \rightarrow \mathbb{R}^d$, the mechanism $\mathcal{A}$*

$$\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + Laplace(GS(f)/\epsilon) \tag{2.4}$$

gives ϵ-differential privacy. ■

For example, for a single counting query Q over a dataset $\mathcal{D}$, returning $Q(\mathcal{D}) + Laplace(1/\epsilon)$ maintains ϵ-differential privacy because a counting query has a sensitivity 1.

Exponential Mechanism

For the analysis whose outputs are not real or make no sense after adding noise, McSherry and Talwar [89] propose the exponential mechanism that selects an output from the output domain, $r \in \mathcal{R}$, by taking into consideration its score of a given utility function q in a differentially private manner. The exponential mechanism assigns exponentially greater probabilities of being selected to outputs of higher scores so that the final output would be close to the optimum with respect to q. The chosen utility function q should be insensitive to changes of any particular record, that is, has a low sensitivity. Let the sensitivity of q be $GS(q) = \max_{\forall r, \mathcal{D}_1, \mathcal{D}_2} |q(\mathcal{D}_1, r) - q(\mathcal{D}_2, r)|$.

Theorem 2.4. *Given a utility function $q : (\mathcal{D} \times \mathcal{R}) \rightarrow \mathbb{R}$ for a database $\mathcal{D}$, the mechanism $\mathcal{A}$,*

$$\mathcal{A}(\mathcal{D}, q) = \left\{ return\ r\ with\ probability \propto exp\left(\frac{\epsilon q(\mathcal{D}, r)}{2GS(q)}\right)\right\} \tag{2.5}$$

gives ϵ-differential privacy. ■

2.3 Utility Metrics

Privacy is one aspect of privacy-preserving data publishing; utility is the other. In general, each anonymization operation increases privacy, but decreases utility. To obtain a better trade-off between privacy and utility, we prefer the anonymization operations that increase more privacy at the cost of less utility loss. This requires the quantification of data utility. Consequently, some utility metrics have been proposed for measuring either the usefulness of the anonymized data or the information loss due to the anonymization process. Such utility metrics roughly fall into three categories: *general purpose metrics*, *special purpose metrics* and *trade-off purpose metrics*.

2.3.1 General Purpose Metrics

In many cases, the data publisher does not know how the published data will be analyzed by the recipient. In this case, general purpose metrics are needed. The basic idea of general purpose metrics is to measure the "similarity" between the original data and the anonymized data, which underpins the *principle of minimal distortion* [104], [108]. One of the most intuitive general purpose information metrics is the number of anonymization operations (e.g., generalization or suppression) performed on a dataset [104]. For example, generalizing 10 instances of *Engineer* to $Professional$ causes 10 units of distortion, and further generalizing these instances to ANY_Job causes another 10 units of distortion. In addition, $ILoss$ [124] and *discernibility metric* (DM) [106] are two examples of general purpose metrics that are widely used. $ILoss$ charges a penalty for generalizing a value in a record independently of other records, while the discernibility metric addresses the notion of loss by charging a penalty to each record for being indistinguishable from other records with respect to QID.

2.3.2 Special Purpose Metrics

If the purpose of the data is known at the time of publication, it can be taken into account during the anonymization process to better preserve data utility. For example, if the data is published for modeling the classification of a target attribute in the table, then it is important not to generalize the values whose distinctions are essential for discriminating the class labels in the target attribute. This intuition is reflected in the *classification metric*. Iyengar [64] proposed the classification metric or CM to measure the classification error on the training data. The idea is to charge a penalty for each record suppressed or generalized to a group in which the records class is not the majority class. The intuition is that a record having a non-majority class in a group will be classified as the majority class, which is an error because it disagrees with the record's original class. Special purpose metrics are especially important for differential privacy, under which we can guarantee utility for only restricted classes of data analysis tasks.

2.3.3 Trade-off Purpose Metrics

All above information metrics aim at preserving maximum data usefulness, but the problem is that the anonymization operation that gains maximum information may also lose so much privacy that no other anonymization operation can be performed. The idea of trade-off metrics is to consider both the privacy and information requirements at every anonymization operation and to determine an optimal trade-off between the two requirements [47], [48].

Chapter 3

Literature Review

The privacy concern in data publishing was first arisen in the field of official statistics in the 1960s. In the context of statistical databases, the objective of privacy protection is to protect confidentiality of any individual entity in the database while allowing its users to retrieve aggregate statistics [2]. Recently, with the fast development of data mining techniques, there are increasing demands on publishing entire database records, instead of just aggregate statistics [46]. Such demands obviously present greater challenges to privacy protection, and have spawned a new wave of research on PPDP towards data mining purpose. In this section, we present an in-depth literature review on the recent developments of PPDP.

3.1 Sanitizing Statistical Data

Statistical databases are maintained by various organizations to support their short-term and long-term planning activities. The users of statistical databases are entitled to submit interactive queries for aggregate statistics. The aggregate statistics may disclose an individual's sensitive information that is stored in the database, such as income, disease, and credit rating. Therefore, the problem of sanitizing statistical data, also known as *statistical disclosure control* [2], [13], is to protect confidentiality

based on the interactive query model.

Perturbation is the most important sanitization approach used in statistical disclosure control due to its simplicity, efficiency, and ability to preserve statistical information. The general idea of perturbation is to replace the original data values with some synthetic data values so that the statistical information computed from the perturbed data does not differ significantly from the statistical information computed from the original data. The perturbed data records do not correspond to real-world record owners, so the attacker cannot perform the sensitive linkages or recover sensitive information from the published data. Following the general idea, three concrete sanitization methods have been widely used, namely additive noise, data swapping, and synthetic data generation. Additive noise [2], [13] is often used for hiding sensitive numerical data (e.g., salary). The basic idea is to replace the original sensitive value s with $s + r$ where r is a random value drawn from some distribution. Privacy is measured by how closely the original values of a modified attribute can be estimated [5]. Two papers [44], [70] showed that some simple statistical information, like means and correlations, can be preserved by adding random noise. Data swapping can be used to protect both numerical attributes [102] and categorical attributes [101]. It sanitizes a data table by exchanging values of sensitive attributes among individual records while maintaining the low-order frequency counts or marginals. Synthetic data generation builds a statistical model from the raw data and then samples points from the model. These sampled points form the synthetic data for data publication instead of the original data.

Recently, Dwork et al. [37] proposed an insightful privacy notion based on the principle that the risk to a record owner's privacy should not substantially increase as a result of participating in a statistical database. Instead of comparing the prior and posterior probabilities before and after accessing the published data, Dwork et al. proposed to compare the probability change of an adversary on databases with and without the record owner's data. Consequently, Dwork et al. [37] proposed

a privacy model called ϵ-*differential privacy* to ensure that the outcome of any analysis is insensitive to the removal or addition of a single database record. It follows that even if a user had opted to be included in the database, there would not be a significant change in any computation based on the database. Therefore, this assures every record owner that any privacy breach will not be a result of participating in a database.

Differential privacy lies on a rigorous mathematical foundation, and has been shown to guarantee formal, provable privacy protection independent of an adversary's background knowledge and computational power. These days, differential privacy has gained substantial attention and become the de facto standard for privacy protection. We review the recent applications of differential privacy in Section 3.6.

3.2 Sanitizing Relational Data

The traditional k-anonymity model [104], [108] and its extensions [75], [120], [85], [95], [77], [114] were originally proposed to protect private and sensitive information in the context of relational data. All these privacy models are based on the common assumption that an adversary may use any or even all attributes in the *quasi-identifier* (QID) to launch identity and attribute linkage attacks. k-anonymity [104], [108] prevents identity linkage attacks by requiring every qid group in a table to contain at least k records. Most works on k-anonymity focus on anonymizing a single data table; however, a real-life database usually contains multiple relational tables. Nergiz et al. [95] proposed a privacy model called *MultiR* k-*anonymity* to ensure k-anonymity on multiple relational tables. Their model assumes that a relational database contains a person-specific table PT and a set of tables $T_1, \cdots, T_n$, where PT contains a person identifier Pid and some sensitive attributes, and T_i, for $1 \leq i \leq n$, contains some foreign keys, some attributes in

QID, and sensitive attributes. The general privacy notion is to ensure that for each record owner o contained in the join of all tables $PT \bowtie T_1 \bowtie \cdots \bowtie T_n$, there exist at least $k-1$ other record owners who share the same QID with o.

ℓ-diversity [85] and confidence bounding [114] aim to prevent attribute linkage attacks. ℓ-diversity requires every qid group to contain at least ℓ "well-represented" sensitive values, while confidence bounding limits an adversary's confidence of inferring a sensitive value in any qid group to a certain threshold. (α, k)-anonymity [120] incorporates both k-anonymity and confidence bounding into a single privacy model, requiring every qid in a table T to be shared by at least k records and $conf(qid \rightarrow s) \leq \alpha$ for any sensitive value s, where k and s are data publisher specified thresholds. Li et al. [77] observed that when the overall distribution of a sensitive attribute is skewed, ℓ-diversity does not prevent attribute linkage attacks. Consider a patient table where 95% of records have *Flu* and 5% of records have *HIV*. Suppose that a qid group has 50% of *Flu* and 50% of *HIV* and, therefore, satisfies 2-diversity. However, this group presents a serious privacy threat because any record owner in the group could be inferred as having HIV with 50% confidence, compared to 5% in the overall table. To prevent *skewness attacks*, Li et al. [77] proposed a privacy model, called *t-closeness*, that requires the distribution of a sensitive attribute in any qid group to be close to the distribution of the attribute in the overall table. t-closeness uses the *earth mover's distance* (EMD) to measure the closeness between two distributions of sensitive values, and requires the closeness to be within t. $(\epsilon, \delta)^k$-dissimilarity [116] thwarts both identity and attribute linkage attacks for a much wider range of data models, where the proximity of sensitive values is defined by an arbitrary function.

In the above privacy models, only very limited background knowledge is considered. Yet, recent works have shown the importance of integrating an adversary's background knowledge in privacy quantification. A common challenge faced by all research on integrating background knowledge is to determine what and how much knowledge should be considered. Li and Li [78] modeled an adversary's background

knowledge by mining negative association rules from the data and then using them in the anonymization process. Chen et al. [19] argued that since it is infeasible for a data publisher to anticipate the background knowledge possessed by an adversary, the interesting research direction is to consider only the background knowledge that arises naturally in practice and can be efficiently handled. In particular, three types of background knowledge are considered in [19]: knowledge about the target individual, knowledge about other individuals, and knowledge about the group of individuals having the same sensitive value as that of the target individual. The three-dimensional knowledge is quantified as a (ℓ, k, m) triplet, which indicates that an adversary knows: (1) ℓ sensitive values that the target individual t does not have, (2) the sensitive values of other k individuals, and (3) m individuals having the same sensitive value as that of t. Then, the authors proposed the *skyline privacy criterion*, which allows the data publisher to specify a set of *incomparable* (ℓ, k, m) triplets, called a *skyline*, along with a set of confidence thresholds for a sensitive value δ in order to provide more precise and flexible privacy quantification. The major shortcoming of privacy skyline is its limited expressive power of background knowledge. For example, it fails to express probabilistic background knowledge.

Du et al. [35] specifically modeled an adversary's background knowledge in the form of probabilities, for example, $P(OvarianCancer|Male) = 0$. The primary privacy risks in PPDP come from the linkings between the sensitive attributes (SA) and the quasi-identifiers (QI). Quantifying privacy is, therefore, to derive $P(SA|QI)$ for any instance of SA and QI with the probabilistic background knowledge. Du et al. [35] formulated the derivation of $P(SA|QI)$ as a non-linear programming problem. Currently, the paper is limited in handling only equality background knowledge constraints. Since both papers [19], [35] are unaware of the exact background knowledge possessed by an adversary, Li et al. [79] proposed a generic framework to systematically model different types of background knowledge an adversary may possess. Yet, in this paper the authors limit their scope to background knowledge

that is consistent with the original dataset T. Then, modeling background knowledge is to estimate the adversary's prior belief of the sensitive attribute values over all possible QI values. This can be achieved by identifying the underlying prior belief function that best fits T using a kernel regression estimation method.

3.3 Sanitizing Set-Valued Data

Due to the nature of *high dimensionality* in set-valued data, the extensive research on privacy-preserving data publishing (PPDP) for relational data does not fit well with set-valued data [46]. Some recent papers have started to address the problem of anonymizing set-valued data for the purpose of data mining [17], [50], [62], [110], [111], [128], [129]. These existing works can be broadly divided into two categories according to whether they distinguish the items between *sensitive* and *non-sensitive*.

Ghinita et al. [50] and Xu et al. [128], [129] deliberately divided all items in the universe into either sensitive or non-sensitive, and further assumed that an adversary's background knowledge is strictly confined to non-sensitive items, which are considered quasi-identifiers for launching privacy attacks. In [50], an adversary is modeled with background knowledge of arbitrary number of non-sensitive items. They proposed a bucketization-based approach that limits the probability of inferring a sensitive item to a specified threshold, while preserving correlations among items for frequent pattern mining. In addressing the high dimensionality of set-valued data, Xu et al. [129] bounded the background knowledge of an adversary to at most p non-sensitive items, and intended to prevent both identity attacks and attribute attacks. Specifically, global suppression was employed with the goal of preserving as many item instances as possible. Xu et al. [128] improved the approach proposed in [129] in two ways: instead of preserving item instances, they aimed to preserve frequent itemsets; they presented a border representation, which avoids enumerating an exponential number of moles and nuggets.

All these works suffer from two main drawbacks. First, when an adversary is aware of some, even few, sensitive items, other sensitive items could be learned. Second, in many cases there does not exist a consensus of "sensitive". Items sensitive to someone may not be sensitive to others. Cao et al. [17] addressed the first concern by assuming that an adversary may possess background knowledge on sensitive items and proposed a privacy notion *ρ-uncertainty*, which bounds the confidence of inferring a sensitive item from any subset of items (sensitive or non-sensitive) to ρ. They employed both global suppression (for both sensitive and non-sensitive items) and global generalization (for only non-sensitive items).

In addressing the second concern, He and Naughton [62] and Terrovitis et al. [110], [111] eliminated the distinction between sensitive and non-sensitive. Any item could be both sensitive and non-sensitive at the same time. Incorporating both k-anonymity and confidence bounding in such a setting is a challenging, if not impossible, task due to the inherent conflicting requirements of the two privacy models. As a result, these authors considered only identity attacks. Similar to the idea of [128] and [129], Terrovitis et al. [110] proposed to bound the background knowledge of an adversary by the maximum number m of items and proposed a new privacy model, k^m-anonymity, a relaxation of k-anonymity. They achieved k^m-anonymity by a bottom-up global generalization solution. To improve the utility, recently Terrovitis et al. [111] provided a local recoding method for achieving k^m-anonymity. He and Naughton [62] pointed out that k^m-anonymity provides a weaker privacy protection than k-anonymity and proposed a top-down local generalization solution under k-anonymity. We argue that even k-anonymity provides insufficient privacy protection for set-valued data. Consider an extreme case in which all records are identical. If an adversary assures the presence of the victim in the table, he learns every item of the victim without *any* background knowledge.

3.4 Sanitizing Trajectory and Sequential Data

More broadly, sequential data can be considered as a special kind of trajectory data. Due to the ubiquitousness of trajectory data and sequential data, some recent works [1], [109], [100], [130], [63], [93] have started to study privacy-preserving trajectory data publishing from different perspectives. Abul et al. [1] proposed the (k, δ)-anonymity model based on the inherent imprecision of sampling and positioning systems, where δ represents the possible location imprecision. The core step of their anonymization method is based on *space translation*. The general idea is to modify trajectories so that k different trajectories co-exist in a cylinder of the radius δ. One limitation of this approach lies in the fundamental assumption that the trajectories are of some extent of imprecision, which may not be true for trajectory data from many sources, for example, purchase data, RFID data and transit data.

Terrovitis and Mamoulis [109] modeled an adversary's background knowledge as a set of projections of sequences in a sequential database, and assumed that the data holder has to be aware of all such adversarial knowledge. They consequently proposed a data suppression technique that limits the confidence of inferring the presence of a location in a sequence to a pre-defined probability threshold while minimizing the average difference between the original dataset and the published one. The assumption of knowing all adversarial knowledge before publishing the data is possible in the specific scenario described in their paper, but it is not applicable in general in the context of sequential data.

Pensa et al. [100] proposed a variant of k-anonymity model for sequential data with the goal of preserving frequent sequential patterns and developed the brute force pattern-preserving k-anonymization (BF-P2kA) algorithm, which consists of three steps. In the first step, the sequences in the raw dataset are used to build a prefix tree. Then the prefix tree is pruned to make sure that all branches are with a support greater than k. After this, the pruned infrequent sequences are re-appended to the prefix tree based on *longest common subsequence* (LCS). In the last step, an

anonymous dataset is re-generated based the processed prefix tree.

Yarovoy et al. [130] argued that in moving object databases, there does not exist a fixed set of QID attributes for all moving objects (MOB). They considered timestamps as the QIDs and assumed that an attacker conducts privacy attacks based on an *attack graph*. A moving object database satisfies *MOB k-anonymity* if every node in the attack graph G has at least degree k and G is symmetric. The MOB anonymization algorithm is composed of two steps: identifying anonymization groups and generalizing the groups to common regions according to the QIDs while achieving minimal information loss, which is measured as the reduction in the probability of accurately determining the position of an object over all timestamps between the raw MOD D and its anonymous version D^*. An underlying assumption of MOB k-anonymity is that the data publisher must be aware of the QIDs of all moving objects in the MOD to publish. However, the paper left the acquisition of QIDs for a data publisher unsolved.

Monreale et al. [93] presented an approach based on spatial generalization in order to achieve k-anonymity. The novelty of their approach lies in a generalization scheme that depends on the underlying trajectory dataset rather than a fixed grid hierarchy. Fung et al. [45] proposed a trajectory anonymization algorithm based on *global suppression*, and their utility metric is the number of location-time pair instances suppressed due to anonymization. Hu et al. [63] presented the problem of k-anonymizing a trajectory database with respect to a sensitive event database. The goal is to make sure that every event is shared by at least k users. Specifically, they developed a new generalization mechanism known as *local enlargement*, which achieves better utility than conventional hierarchy- or partition-based generalization.

All these works [1], [109], [100], [130], [63], [93] are limited to privacy protection for only identity linkage attacks based on k-anonymity, however, recently researchers have realized that k-anonymity bears some inherent limitations on trajectory data anonymization. For example, in the emerging data publishing scenario, trajectory

data is published with some other sensitive attributes. Even though an individual can be hidden in a group with size greater than k, an adversary can still infer his sensitive information if the group does not have enough diversity on the sensitive attributes. Therefore, it is valuable to employ more rigorous privacy models to the problem of trajectory data anonymization. In addition, all these approaches are effective in some specific scenarios. A generic anonymization framework that is able to accommodate different data utility requirements is still indispensable.

3.5 Sanitizing Network Data

Since Backstrom et al.'s study [7] of privacy attacks on social networks, the problem of privacy-preserving network data publishing has received increasing attention.

A large line of research studies how to prevent a node from re-identification against an adversary with background knowledge on the network structure (and node attributes). Liu and Terzi [82] proposed the notion of k-degree anonymity, which requires that for every node v there exist at least $k-1$ other nodes with the same degree as v. Zhou and Pei [133] demanded that any vertex cannot be re-identified in an anonymized graph with probability greater than $\frac{1}{k}$ by an adversary equipped with 1-neighborhood background knowledge. Hay et al. [60] generalize a graph by grouping nodes into partitions with size at least k, and only release the size of each partition and the density of edges. Cormode et al. [30] anonymize graphs using label lists based on a critical safety condition and then release only the number of edges among different node classes.

Recently, Zou et al. [134] proposed k-automorphism, which resists any structural attack by enforcing $k-1$ automorphic functions in the published data. Cheng et al. [26] presented the notion of k-security based on k-isomorphism. It requires an input graph to be transformed to k *disjoint* isomorphic subgraphs. The strong point of k-isomorphism is that it prevents not only node re-identification but also edge

disclosure (i.e., an adversary should not be able to determine if two nodes are connected with probability $> \frac{1}{k}$). However, k-isomorphism's ability in preventing edge disclosure is still limited. Yuan et al. [132] introduced a framework that provides personalized privacy protection. Specifically, they defined three levels of protection requirements and combined label generalization and other structure protection techniques in order to achieve improved utility.

Another line of research aims at obfuscating an adversary's certainty of the presence of a link between two targets. Ying and Wu [131] developed randomized edge addition, deletion and switch methods with the goal of preserving spectrum of networks. Liu et al. [83] considered a special situation where edges are weighted. They proposed two privacy preserving strategies, one based on a Gaussian randomization multiplication and the other based on a greedy perturbation algorithm, in order to preserve shortest paths between pairs of nodes. Wu et al. [122] presented a low rank approximation based reconstruction algorithm, which recovers spectral properties of a randomized graph. In general, all these works lack a formal privacy definition and are only resistant to certain types of privacy attacks.

3.6 Applications of Differential Privacy

In the last few years, differential privacy has been gaining considerable attention in various data sharing scenarios. Most of the research on differential privacy concentrates on the *interactive* model with the goal of either reducing the magnitude of added noise [61], [76], [103] or releasing certain data mining results [8], [11], [43], [71], [84]. Lately, several works [12], [39], [125], [126] have started to address the use of differential privacy in PPDP as a substitute for k-anonymity and its extensions. Blum et al. [12] demonstrated that it is possible to release *synthetic* private databases that are useful for all queries over a discretized domain from a concept

class with polynomial VC-dimension[1]. However, their solution is not efficient, taking runtime complexity of $superpolynomial(|\mathcal{C}|, |\mathcal{I}|)$, where $|\mathcal{C}|$ is the size of a concept class and $|\mathcal{I}|$ the size of the item universe. This fact makes their mechanism impractical for real applications. To improve the efficiency, Dwork et al. [39] proposed a recursive algorithm of generating a *synthetic* database with runtime complexity of $polynomial(|\mathcal{C}|, |I|)$. This improvement, however, is still insufficient to handle real-life high-dimensional datasets because $|\mathcal{C}|$ is an exponential function of $|\mathcal{I}|$.

Xiao et al. [126] proposed a two-step algorithm for *relational data*. It first issues queries for *every* possible combination of attribute values to the PINQ interface [87], and then produces a generalized output using the perturbed dataset returned by PINQ. Apparently, this approach is computationally expensive in the context of high-dimensional data. For example, for set-valued data it requires issuing a total of $2^{|\mathcal{I}|} - 1$ queries, where $|\mathcal{I}|$ is the total number of possible items. The works reported in [12], [39], [126] are based on the query model. In contrast, Xiao et al. [125] assumed that their algorithm has direct and unconditional access to the underlying *relational data*. They proposed a wavelet-transformation based approach that lowers the magnitude of noise than adding independent Laplace noise. Similarly, the algorithm needs to process all possible entries in the entire output domain, which causes a scalability problem for high-dimensional data.

With the wide acknowledgment of differential privacy, some papers [58], [66], [54] have started to apply differential privacy to network data from different perspectives. All these works consider a special instantiation of differential privacy, known as *edge-differential privacy* [58], where a neighbor of a network dataset is obtained by either adding/removing an edge or by adding/removing an *isolated* node. Edge-differential privacy protects individual edges from being disclosed. Hay et al. [58] studied the publication of a private estimate of the degree distribution of a network

[1]Vapnik-Chervonenkis (VC) dimension is a measure of the complexity of a concept in the class.

via constrained inferences. Karwa et al. [66] provided efficient algorithms for answering triangles, k-triangles and k-stars with instance-dependent noise. Gupta et al. [54] gave new algorithms for generating a non-interactive release that is useful for cut queries. The key idea is to pair an *iterative database construction* algorithm with a *distinguisher*, which returns a cut query with a significantly different value on an intermediate database, in order to give increasingly accurate approximations with respect to cut queries. However, as explained later in Chapter 7, there are several major limitations that obstruct its application to real-life network datasets.

Chapter 4

Set-Valued Data Sanitization

4.1 Introduction

Set-valued data, such as transaction data, web search queries, and click streams, refers to the data in which each record owner is associated with a set of items drawn from a universe of items [62], [110], [111]. Sharing set-valued data provides enormous opportunities for various data mining tasks in different application domains such as marketing, advertising, and infrastructure management. However, such data often contains sensitive information that could violate individual privacy. Such privacy concerns are even exacerbated in the emerging computing paradigms, for example *cloud computing*. Therefore, set-valued data needs to be sanitized before it can be released to the public. In this chapter, we consider the problem of publishing set-valued data that simultaneously protects individual privacy under the framework of *differential privacy* [37] and provides guaranteed utility to data analysts.

There has been some existing research [17], [50], [62], [110], [111], [128], [129] on publishing set-valued data based on *partition-based privacy models* [49], for example k-anonymity [108] (or its relaxation, k^m-anonymity [110], [111]) and/or confidence bounding [17], [114]. However, the recent discovery of several privacy attacks [67], [94], [119] has questioned the capability of partition-based privacy models on privacy

protection. The vulnerability of partition-based privacy models is largely due to their deterministic nature, which makes them fail to resist an adversary with substantial background knowledge. This fact motivated our use of differential privacy to provide formal, provable privacy guarantees for set-valued data publication. However, the application of differential privacy to set-valued data publishing is very challenging, especially in terms of efficiency due to its high dimensionality. By definition, any differentially private mechanism has to be insensitive to the addition/removal of a single record, which, in the context of set-valued data, could be an *arbitrary* itemset derived from the item universe $\mathcal{I}$. To mask this record, existing techniques [12], [39], [125], [126] *explicitly* consider a total of $2^{|\mathcal{I}|} - 1$ itemsets that can be derived from $\mathcal{I}$. Since $|\mathcal{I}|$ in a real-life application could be over a thousand, these approaches can hardly be applied to a real-life data sharing scenario.

In addressing this problem, we initiate the line of *data-dependent* solutions, which adaptively narrow down the output domain by using noisy answers obtained from the underlying database. A data-dependent solution also has a positive impact on the resulting utility as there is no need to add noise to every possible entry in the output domain. The main technical challenge is how to make use of a specific dataset while satisfying differential privacy. In particular, for set-valued data, we demonstrate that in the presence of a *context-free taxonomy tree* we can efficiently generate a sanitized release of set-valued data in a differentially private manner with guaranteed utility for counting queries and many other data mining tasks. Unlike the use of taxonomy trees in the *generalization* mechanism for partition-based privacy models, where the taxonomy trees are highly specific to a particular application, the taxonomy tree required in our solution does not necessarily need to reflect the underlying semantics and, therefore, is context-free. This feature makes our approach flexible for applying to various kinds of set-valued datasets. In this chapter, we also discuss how to apply the data-dependent idea to other data types.

Contributions. We summarize the contributions of this chapter as follows.

- This is the first study of publishing set-valued data via differential privacy. The previous anonymization techniques [17], [50], [62], [110], [111], [128], [129] developed for publishing set-valued data are dedicated to partition-based privacy models. Due to their deterministic nature, they cannot be used for achieving differential privacy. In this chapter, we propose a *probabilistic* top-down partitioning algorithm that provides provable utility under differential privacy, one of the strongest privacy models.

- This is the first work that proposes an efficient *non-interactive* approach scalable to high-dimensional set-valued data with guaranteed utility under differential privacy. We stress that our goal is to publish the *data*, not data mining results. Publishing data provides much greater flexibilities for data miners than publishing data mining results. We show that a more efficient and effective solution could be achieved by making use of the underlying dataset, instead of explicitly considering *all* possible outputs as used in the existing works [12], [39], [125], [126]. For a set-valued dataset, it could be done by a top-down partitioning process based on a context-free taxonomy tree. The use of a context-free taxonomy tree makes our approach applicable to all kinds of set-valued datasets. We prove that the result of our approach is (δ, β)-useful for counting queries, which guarantees the usefulness for data mining tasks based on counts, e.g., mining frequent patterns and association rules [55]. We argue that the general idea of data-dependent solutions has a wider application, for example, to relational data in which each attribute is associated with a taxonomy tree. This implies that some traditional data publishing methods, such as TDS [48] and Mondrian [75], could be adapted to satisfy differential privacy.

The results of this chapter have been published in [25].

Table 4.1: A sample set-valued dataset

Rec. #	**Items**
1	$\{I_1, I_2, I_3, I_4\}$
2	$\{I_2, I_4\}$
3	$\{I_2\}$
4	$\{I_1, I_2\}$
5	$\{I_2\}$
6	$\{I_1\}$
7	$\{I_1, I_2, I_3, I_4\}$
8	$\{I_2, I_3, I_4\}$

4.2 Preliminaries

4.2.1 Set-Valued Data

Let $\mathcal{I} = \{I_1, I_2, ..., I_{|I|}\}$ be the universe of items, where $|\mathcal{I}|$ is the size of the item universe. The multiset $\mathcal{D} = \{D_1, D_2, ..., D_{|\mathcal{D}|}\}$ denotes a set-valued dataset, where each record $D_i \in \mathcal{D}$ is a non-empty subset of $\mathcal{I}$. Table 4.1 presents an example of set-valued datasets with the item universe $\mathcal{I} = \{I_1, I_2, I_3, I_4\}$.

Given the item universe $\mathcal{I}$, the output domain $\mathcal{O}$ of a set-valued database is composed of all possible itemsets that can be derived from $\mathcal{I}$. Therefore, the size of the output domain $|\mathcal{O}| = \sum_{k=1}^{|\mathcal{I}|} \binom{|\mathcal{I}|}{k} = 2^{|\mathcal{I}|} - 1$.

Example 4.2.1. Given $\mathcal{I} = \{I_1, I_2, I_3\}$, the associated output domain $\mathcal{O} = \{\{I_1\}, \{I_2\}, \{I_3\}, \{I_1, I_2\}, \{I_1, I_3\}, \{I_2, I_3\}, \{I_1, I_2, I_3\}\}$. The size of $\mathcal{O}$ is $2^3 - 1 = 7$. ■

4.2.2 Context-Free Taxonomy Tree

A set-valued dataset could be associated with a single *taxonomy tree*. In the classic generalization mechanism, the taxonomy tree required is highly specific to a particular application. This constraint has been considered a major limitation of applying generalization [4]. The reason of requiring an application-specific taxonomy tree is that the release contains generalized items that need to be *semantically* consistent

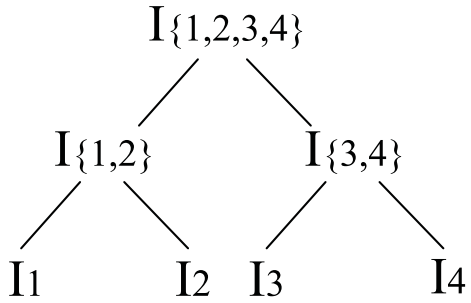


Figure 4.1: A context-free taxonomy tree of the sample data.

with the original items. In our approach, we publish only original items; therefore, the taxonomy tree could be *context free*.

Definition 4.1 (Context-Free Taxonomy Tree)**.** A context-free taxonomy tree is a taxonomy tree, whose internal nodes are a set of their leaves, not necessarily the semantic generalization of the leaves. ■

For example, Figure 4.1 presents a context-free taxonomy tree for Table 4.1, and one of its internal nodes $I_{\{1,2,3,4\}} = \{I_1, I_2, I_3, I_4\}$. We say that an item can be *generalized* to a taxonomy tree node if it is in the node's set. For example, I_1 can be generalized to $I_{\{1,2\}}$ because $I_1 \in \{I_1, I_2\}$.

4.2.3 Utility Metrics

Due to the lower bound results [34], [37], [38], we can only guarantee the utility of restricted classes of queries [12] in the non-interactive setting. In this chapter, we aim to develop a solution for publishing set-valued data that is useful for *counting queries* (also known as *count queries*).

Definition 4.2 (Counting Query)**.** For a given itemset $U \in \mathcal{O}$, a counting query Q over a dataset $\mathcal{D}$ is defined to be $Q(\mathcal{D}) = |\{D \in \mathcal{D} : U \subseteq D\}|$. ■

We choose counting queries because they are crucial to several key data mining tasks over set-valued data, for example, mining frequent patterns and association

rules [55]. We employ (δ, β)-*usefulness* [12] to theoretically measure the utility of sanitized data for counting queries.

Definition 4.3 ((δ, β)-usefulness)**.** A privacy mechanism $\mathcal{A}$ is (δ, β)-useful for queries in class $\mathcal{C}$ if with probability $1 - \beta$, for every $Q \in \mathcal{C}$ and every dataset $\mathcal{D}$, for $\widetilde{\mathcal{D}} = \mathcal{A}(\mathcal{D})$, $|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})| \leq \delta$. ■

(δ, β)-usefulness is effective to give an overall theoretical estimation of utility, but fails to provide intuitive experimental results. Therefore, we experimentally measure the utility of sanitized data for a counting query Q by its *relative error* [125], [123] with respect to the true result over the original database $\mathcal{D}$, which is computed as:

$$error(Q(\widetilde{\mathcal{D}})) = \frac{|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})|}{max\{Q(\mathcal{D}), s\}},$$

where s is a *sanity bound* used to mitigate the influences of queries with extremely small *selectivities* [125], [123]. *Selectivity* is defined as the fraction of records in the dataset satisfying all items in Q [125].

4.3 Sanitization Algorithm

In this section, we present a *Diff*erentially-private sanitization algorithm (given in Algorithm 4.1) that recursively *Part*itions a given set-valued dataset based on a *context-free* taxonomy tree (*DiffPart*).

4.3.1 Partitioning Algorithm

Intuitively, a differentially private release of a set-valued dataset could be generated by adding Laplace noise to a set of counting queries. A simple yet infeasible approach can be achieved by employing Dwork et al.'s method [37]: first generate all distinct itemsets from the item universe; then for each itemset issue a counting query and add Laplace noise to the answer. This approach suffers from two main drawbacks in the

Algorithm 4.1: DiffPart

Input: Raw set-valued dataset $\mathcal{D}$
Input: Fan-out f
Input: Privacy budget ϵ
Output: Sanitized dataset $\widetilde{\mathcal{D}}$

1: $\widetilde{\mathcal{D}} \leftarrow \emptyset$;
2: Construct a taxonomy tree $\mathcal{T}$ with fan-out f;
3: Partition $p \leftarrow$ all records in $\mathcal{D}$;
4: $p.cut \leftarrow$ the root of $\mathcal{T}$;
5: $p.\widetilde{\epsilon} = \epsilon/2$;
6: $p.\alpha = p.\widetilde{\epsilon}/|InternalNodes(p.cut)|$;
7: Add p to an initially empty queue $\mathbb{Q}$;
8: **while** $\mathbb{Q} \neq \emptyset$ **do**
9: Dequeue p' from $\mathbb{Q}$;
10: Sub-partitions $P \leftarrow SubPart\_Gen(p', \mathcal{T})$;
11: **for** each sub-partition $p_i \in P$ **do**
12: **if** p_i is a leaf partition **then**
13: $N_{p_i} = NoisyCount(|p_i|, \epsilon/2 + p_i.\widetilde{\epsilon})$;
14: **if** $N_{p_i} \geq \sqrt{2}C_1/(\epsilon/2 + p_i.\widetilde{\epsilon})$ **then**
15: Add N_{p_i} copies of $p_i.cut$ to $\widetilde{\mathcal{D}}$;
16: **end if**
17: **else**
18: Add p_i to $\mathbb{Q}$;
19: **end if**
20: **end for**
21: **end while**
22: **return** $\widetilde{\mathcal{D}}$;

context of set-valued data. First, it requires a total of $2^{|\mathcal{I}|} - 1$ queries, giving rise to a scalability problem. Second, the noise added to the itemsets that never appear in the original dataset accumulates exponentially, rendering the release useless for data analysis tasks. In fact, these are also the main limitations of other non-interactive approaches [12], [39], [125], [126] when applied to set-valued data. We argue that an efficient solution could be achieved by taking into consideration the underlying dataset. However, attention must be paid because identifying the set of counting queries based on the input dataset may leak its sensitive information and, therefore, violate differential privacy.

We first provide an overview of *DiffPart*. It starts by creating the context-free taxonomy tree. It then generalizes all records to a single partition with a common representation. We call the common representation the *hierarchy cut*, consisting of a set of taxonomy tree nodes. It recursively distributes the records into *disjoint* sub-partitions with more specific representations in a top-down manner based on the taxonomy tree. For each sub-partition, we determine if it is empty *in a noisy way* and further split the sub-partitions considered "non-empty". Our approach stops when no further partitioning is possible in any sub-partition. We call a partition a *leaf partition* if every node in its hierarchy cut is a leaf of the taxonomy tree. Finally, for each leaf partition, the algorithm asks for its noisy size (the noisy number of records in the partition) to construct the release. Our use of a top-down partitioning process is inspired by its use in [62], but with substantial differences. Their approach is used to generate a *generalized* release satisfying k-anonymity while ours is to identify the set of counting queries used to publish differentially private data.

Algorithm 4.1 presents our approach in more detail. It takes as inputs the raw set-valued dataset $\mathcal{D}$, the fan-out f used to construct the taxonomy tree, and also the total privacy budget ϵ specified by the data publisher, and returns a sanitized dataset $\widetilde{\mathcal{D}}$ satisfying ϵ-differential privacy.

Top-down partitioning. The algorithm first constructs the context-free taxonomy tree $\mathcal{T}$ by iteratively grouping f nodes from one level to an upper level until a single root is created. If the size of the item universe is not divided by f, smaller groups can be created.

The initial partition p is created by generalizing all records in $\mathcal{D}$ under a hierarchy cut of a single taxonomy tree node, namely the root of $\mathcal{T}$. A record can be *generalized* to a hierarchy cut if *every* item in the record can be generalized to a node in the cut and *every* node in the cut generalizes some items in the record. For example, the record $\{I_3, I_4\}$ can be generalized to the hierarchy cuts $\{I_{\{3,4\}}\}$ and $\{I_{\{1,2,3,4\}}\}$, but *not* $\{I_{\{1,2\}}, I_{\{3,4\}}\}$. The initial partition p is added to an empty

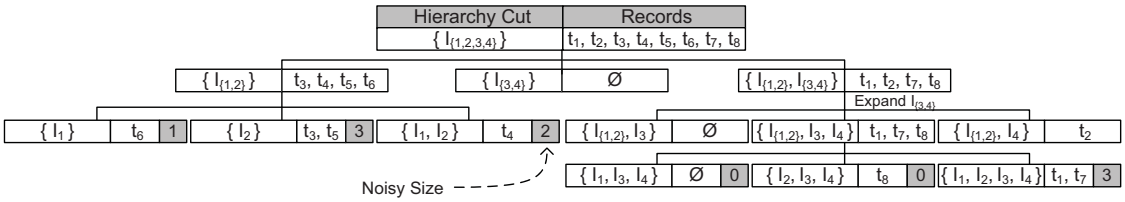


Figure 4.2: The partitioning process.

queue $\mathbb{Q}$.

For each partition in the queue, we need to generate its sub-partitions and identify the non-empty ones for further partitioning. Due to noise required by differential privacy, a sub-partition cannot be *deterministically* identified as non-empty. Probabilistic operations are needed for this purpose. For each operation, a certain portion of privacy budget is required to obtain the noisy size of a sub-partition based on which we decide whether it is "empty". Algorithm 4.1 keeps partitioning "non-empty" sub-partitions until leaf partitions are reached.

Example 4.3.1. Given the dataset in Table 4.1 and a fan-out value 2, a possible taxonomy tree is presented in Figure 4.1, and a possible partitioning process is illustrated in Figure 4.2. Partitions $\{I_{\{3,4\}}\}$, $\{I_{\{1,2\}}, I_3\}$ and $\{I_{\{1,2\}}, I_4\}$ are considered "empty" and, therefore, not further partitioned. ■

Privacy budget allocation. The use of the total privacy budget ϵ needs to be carefully allocated to each probabilistic operation to avoid unexpected termination of the algorithm. Since the operations are used to determine the noisy sizes of the sub-partitions resulted from partition operations, a naive allocation scheme is to bound the maximum number of partition operations needed in the entire algorithm and assign an equal portion to each of them. This approach, however, does not perform well. Instead, we propose a more sophisticated adaptive scheme. We reserve $\epsilon/2$ to obtain the noisy sizes of leaf partitions, which are used to construct the release, and use the rest $\epsilon/2$ to guide the partitioning process. For each partition, we independently calculate the maximum number of partition operations further

needed and assign privacy budget to partition operations based on the number.

The portion of privacy budget assigned to a partition operation is further allocated to the resulting sub-partitions to check their noisy sizes (to see if they are "empty"). Since all sub-partitions from the same partition operation contain *disjoint* records, due to the *parallel composition* property [87], the portion of privacy budget could be used *in full* on each sub-partition. This scheme guarantees that more specific partitions always obtain more privacy budget, complying with the rationale that more general partitions contain more records and, therefore, are more resistant to a smaller privacy budget. We prove this statement after introducing how to calculate the maximum number of partition operations in Theorem 4.1.

Theorem 4.1. *Given a non-leaf partition p with a hierarchy cut and an associated taxonomy tree $\mathcal{T}$, the maximum number of partition operations needed to reach leaf partitions is $|InternalNodes(cut)| = \sum_{u_i \in cut} |InternalNodes(u_i, \mathcal{T})|$, where $|InternalNodes(u_i, \mathcal{T})|$ is the number of internal nodes of the subtree of $\mathcal{T}$ rooted at u_i.* ■

Proof. Given a partition p, our algorithm selects one non-leaf taxonomy tree node from its hierarchy cut to expand at a time. Our algorithm stops when every non-leaf taxonomy tree node in p's hierarchy cut is specialized to a leaf node. For a non-leaf node u in the hierarchy cut, *in the worst case*, it will be replaced by the combination containing all its children. If the children are not leaf node, they need to be split, and *in the worst case* again, it will be replaced by the combination containing all its children. That is, we need to go through all internal nodes of the subtree of $\mathcal{T}$ rooted at u. Therefore, in order to make all non-leaf nodes in p's hierarchy cut to leaf nodes, we need, in the worst case, $\sum_{u_i \in cut} |InternalNodes(u_i, \mathcal{T})|$ partitionings (partition operations).

Take the dataset in Table 4.1 as an example. Consider a partition with the hierarchy cut $\{I_{\{1,2,3,4\}}\}$. After the first partitioning, the sub-partition with the hierarchy cut $\{I_{\{1,2\}}, I_{\{3,4\}}\}$ represents the worst case. Suppose $I_{\{1,2\}}$ is selected to

split, the sub-partition with the hierarchy cut $\{I_1, I_2, I_{\{3,4\}}\}$ presents the worst case. After that, we need one more split on $I_{\{3,4\}}$. Therefore, in the worst case, the total number of partition operations required is 3, which is the number of internal nodes of the taxonomy tree in Figure 4.1. □

Now we prove that our adaptive allocation scheme always assigns more privacy budget to more specific partitions. Let n_i be the maximum number of partition operations calculated according to Theorem 4.1 and m be the total number of partition operations needed to reach leaf partitions. Let $\frac{\epsilon}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{1}{n_{m-1}}$ be the privacy budget assigned to a partition and $\frac{\epsilon}{2}\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m}$ the privacy budget assigned to its sub-partitions, which are more specific. We have $n_i \geq n_{i+1}+1$ because the maximum number of partition operations further needed for a partition is always one more than that of its sub-partitions (we need *at least* one more partition operation to split it to its sub-partitions). We can observe the following.

$$\begin{aligned}
\frac{\epsilon}{2}\prod_{i=1}^{m-1}(1-\frac{1}{n_i})\cdot\frac{1}{n_m} &= \frac{\epsilon}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{n_{m-1}-1}{n_{m-1}}\cdot\frac{1}{n_m} \\
&\geq \frac{\epsilon}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{n_m}{n_{m-1}}\cdot\frac{1}{n_m} \\
&= \frac{\epsilon}{2}\prod_{i=1}^{m-2}(1-\frac{1}{n_i})\cdot\frac{1}{n_{m-1}}
\end{aligned}$$

Using transitivity, we conclude that more specific partitions always receive more privacy budget. Each partition tracks its unused privacy budget $\widetilde{\epsilon}$ and calculates the portion of privacy budget α for the next partition operation. Any privacy budget left from the partitioning process is added to leaf partitions.

Example 4.3.2. For the partitioning process illustrated in Figure 4.2, partitions $\{I_1, I_2\}$, $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, $\{I_{\{1,2\}}, I_3, I_4\}$, and $\{I_1, I_2, I_3, I_4\}$ receive privacy budget $5\epsilon/6$, $\epsilon/6$, $\epsilon/6$ and $2\epsilon/3$, respectively. ■

Algorithm 4.2: SubPart_Gen (Sub-partition generation)

Input: Partition p
Input: Taxonomy tree $\mathcal{T}$
Output: Noisy non-empty sub-partitions V of p

```
1: Initialize a vector V;
2: Select a node u from p.cut to partition;
3: Generate all non-empty sub-partitions S;
4: Allocate records in p to S;
5: for each sub-partition s_i ∈ S do
6:    N_{s_i} = NoisyCount(|s_i|, p.α);
7:    if N_{s_i} ≥ √2 C_2 × height(p.cut)/p.α then
8:       s_i.ε̃ = p.ε̃ − p.α;
9:       s_i.α = s_i.ε̃/|InternalNodes(s_i.cut)|;
10:      Add s_i to V;
11:   end if
12: end for
13: j = 1;
14: l = number of u's children;
15: while j ≤ 2^l − |S| do
16:   N_j = NoisyCount(0, p.α);
17:   if N_j ≥ √2 C_2 × height(p.cut)/p.α then
18:      Randomly generate an empty sub-partition s'_j;
19:      s'_j.ε̃ = p.ε̃ − p.α;
20:      s'_j.α = s'_j.ε̃/|InternalNodes(s'_j.cut)|;
21:      Add s'_j to V;
22:   end if
23:   j++;
24: end while
25: return V;
```

Sub-partition generation. "Non-empty" sub-partitions can be identified by either *exponential mechanism* or *Laplace mechanism*. For exponential mechanism, we can get the noisy number N of non-empty sub-partitions, and then use exponential mechanism to extract N sub-partitions by using the number of records in a sub-partition as the score function. This approach, however, does not take advantage of the fact that all sub-partitions contain *disjoint* datasets, resulting in a relatively small privacy budget for each operation and thus less accurate results. For this reason, we employ Laplace mechanism for generating sub-partitions, whose details

are presented in Algorithm 4.2.

For a *non-leaf* partition, we generate a candidate set of taxonomy tree nodes from its hierarchy cut, containing all non-leaf nodes that are of the largest height in $\mathcal{T}$, and then *randomly* select a node u from the set to expand, generating a total of 2^l sub-partitions, where $l \leq f$ is the number of u's children in $\mathcal{T}$. The sub-partitions can be exhaustively generated by replacing u by the combinations of its children. For example, the partition $\{I_{\{1,2\}}\}$ generates three sub-partitions: $\{I_1\}$, $\{I_2\}$ and $\{I_1, I_2\}$. This technique, however, is inefficient.

We propose an efficient implementation by *separately* handling non-empty and empty sub-partitions of a partition p. Non-empty sub-partitions, usually of a small number, need to be explicitly generated. We issue a counting query for the noisy size of each sub-partition by Laplace mechanism. We use the noisy size to make our decision. We consider a sub-partition "non-empty" if its noisy size $\geq \sqrt{2}C_2 \times height(p.cut)/p.\alpha$. We design the threshold as a function of the standard deviation of the noise and the height of p's hierarchy cut, the largest height of all nodes in p's hierarchy cut. The rationale of taking into consideration the height is that more general partitions should have more records to be worth being partitioned. A constant C_2 is added to the function for the reason of efficiency: we want to prune empty sub-partitions as early as possible. While this heuristic is arbitrary, it provides good experimental results on different real-life datasets.

For empty sub-partitions, we do not explicitly generate all possible ones, but employ a *test-and-generate* method: generate a uniformly random empty sub-partition without replacement only if the noisy count of an empty sub-partition's true count 0 is greater than the threshold. To satisfy differential privacy, empty and non-empty sub-partitions must use the *same* threshold. A C_2 value that is slightly greater than 1 can effectively prune most empty sub-partitions without jeopardizing non-empty ones.

For a *leaf* partition, we use the reserved $\epsilon/2$ plus the privacy budget left from

the partitioning process to obtain its noisy size. To minimize the effect of noise, we add a leaf partition p only if its noisy size $\geq \sqrt{2}C_1/(\epsilon/2 + p.\widetilde{\epsilon})$. Typically, C_1 is a constant in the range of $[1, C_2]$. We argue that since the data publisher has full access to the raw dataset, she could try different C_1 and C_2 values and publish a reasonably good release. We consider how to automatically determine C_1 and C_2 values in future work.

We illustrate how *DiffPart* works in Example 4.3.3.

Example 4.3.3. Given the sample dataset in Table 4.1, a fan-out value 2, and the total privacy budget ϵ, *DiffPart* works as follows (see Figure 4.2 for an illustration). It first creates the context-free taxonomy tree $\mathcal{T}$ illustrated in Figure 4.1 and generalizes all records to a single partition with the hierarchy cut $\{I_{\{1,2,3,4\}}\}$. A portion of privacy budget $\epsilon/6$ is allocated to the first partition operation because there are 3 internal nodes in $\mathcal{T}$ (and $\epsilon/2$ is reserved for leaf partitions).

The algorithm then creates three sub-partitions with the hierarchy cuts $\{I_{\{1,2\}}\}$, $\{I_{\{3,4\}}\}$, and $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, respectively, by replacing the node $I_{\{1,2,3,4\}}$ by different combinations of its children, leading records #3, 4, 5, and 6 to the sub-partition $\{I_{\{1,2\}}\}$ and records #1, 2, 7 and 8 to the sub-partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$. Suppose that the noisy sizes indicate that these two sub-partitions are "non-empty". Further splits are needed on them. There is no need to explore the sub-partition $\{I_{\{3,4\}}\}$ any more as it is considered "empty".

The portions of privacy budget for the next partition operations are *independently* calculated for the two partitions. For the partition $\{I_{\{1,2\}}\}$, there is at most one more partition operation and, therefore, it gets the privacy budget $\epsilon/3$; for the partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, $\epsilon/6$ is allocated as there are still two internal nodes in its hierarchy cut. A further split of $\{I_{\{1,2\}}\}$ creates three leaf partitions, $\{I_1\}$, $\{I_2\}$, and $\{I_1, I_2\}$. For the partition $\{I_{\{1,2\}}, I_{\{3,4\}}\}$, assume that $I_{\{3,4\}}$ is randomly selected to expand. This generates three sub-partitions: $\{I_{\{1,2\}}, I_3\}$, $\{I_{\{1,2\}}, I_4\}$, and $\{I_{\{1,2\}}, I_3, I_4\}$ with record #2 in $\{I_{\{1,2\}}, I_4\}$, and records #1, 7, 8 in $\{I_{\{1,2\}}, I_3, I_4\}$.

Assume that the partition $\{I_{\{1,2\}}, I_3, I_4\}$ is considered "non-empty". One more partition operation is needed and $\epsilon/6$ privacy budget is allocated.

After the last partitioning, we get three more leaf partitions with the hierarchy cuts $\{I_1, I_3, I_4\}$, $\{I_2, I_3, I_4\}$ and $\{I_1, I_2, I_3, I_4\}$. For all leaf partitions, we use the reserved $\epsilon/2$ plus the privacy budget left from the partitioning process to calculate their noisy sizes. This implies $5\epsilon/6$ for $\{I_1\}$, $\{I_2\}$, and $\{I_1, I_2\}$, and $2\epsilon/3$ for $\{I_1, I_3, I_4\}$, $\{I_2, I_3, I_4\}$ and $\{I_1, I_2, I_3, I_4\}$. ■

One interesting observation is that with the partitioning process, the hierarchy cuts of the sub-partitions resulted from the same partition operation become more similar. For this reason, to some extent the effect of noise for counting queries is mitigated (recall that the mean of noise is 0).

4.3.2 Analysis

Privacy analysis. The privacy guarantee of our solution is given in Theorem 4.2.

Theorem 4.2. *Algorithm 4.1 together with Algorithm 4.2 satisfies ϵ-differential privacy.* ■

Proof. In essence, the only information obtained from the underlying dataset is the noisy sizes of the partitions (or equivalently, the noisy answers of a set of counting queries). Due to noise, any itemset from the universe may appear in the sanitized release. In the previous work [87], it has been proven that partitioning a dataset by *explicit* user inputs does not violate differential privacy. However, the actual partitioning result should not be revealed as it violates differential privacy. This explains why we need to consider every possible sub-partition and use its noisy size to make decision.

Let a sequence of partitionings that consecutively distributes the records in the initial partition to leaf partitions be a *partitioning chain*. Due to Theorem 2.2, the privacy budget used in each partitioning chain is independent of those of other

chains. Therefore, if we can prove that the total privacy budget used in each partitioning chain is less than or equal to ϵ, we get the conclusion that Algorithm 4.1 together with Algorithm 4.2 satisfies ϵ-differential privacy.

Let m be the total number of partitionings in a partitioning chain and n_i the maximum number of partitionings calculated according to Theorem 4.1. We can formalize the proposition to be the following equivalent problem.

$$\epsilon \geq \underbrace{\frac{\epsilon}{2} \cdot \frac{1}{n_1}}_{\text{first partitioning}} + \underbrace{\frac{\epsilon}{2} \cdot (1 - \frac{1}{n_1}) \cdot \frac{1}{n_2}}_{\text{second partitioning}} + \cdots + \underbrace{\frac{\epsilon}{2} \prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot \frac{1}{n_m} + \frac{\epsilon}{2}}_{\text{last partitioning}}$$

$$\textit{Subject to } n_i \geq n_{i+1} + 1 \textit{ and } n_m = 1.$$

Each item of the right hand side (RHS) of the above equation represents the portion of privacy budget allocated to a partition operation. The entire RHS gives the total privacy budget used in the partitioning chain. We prove the correctness of the equation below.

The above equation can be rewritten as the following equivalent form:

$$\frac{1}{n_1} + \sum_{i=1}^{m-1} (\prod_{j=1}^{i} (1 - \frac{1}{n_j}) \cdot \frac{1}{n_{i+1}}) \leq 1$$

$$\textit{Subject to } n_i \geq n_{i+1} + 1 \textit{ and } n_m = 1.$$

We add one more non-negative item $\prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot (1 - \frac{1}{n_m})$ to the left hand side of the above equation. We obtain the following.

$$\frac{1}{n_1} + \sum_{i=1}^{m-1} (\prod_{j=1}^{i} (1 - \frac{1}{n_j}) \cdot \frac{1}{n_{i+1}}) + \prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot (1 - \frac{1}{n_m})$$

$$
\begin{aligned}
&= \frac{1}{n_1} + \sum_{i=1}^{m-2} (\prod_{j=1}^{i} (1 - \frac{1}{n_j}) \cdot \frac{1}{n_{i+1}}) + \prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot \frac{1}{n_m} + \prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot (1 - \frac{1}{n_m}) \\
&= \frac{1}{n_1} + \sum_{i=1}^{m-2} (\prod_{j=1}^{i} (1 - \frac{1}{n_j}) \cdot \frac{1}{n_{i+1}}) + \prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \\
&= \frac{1}{n_1} + \sum_{i=1}^{m-2} (\prod_{j=1}^{i} (1 - \frac{1}{n_j}) \cdot \frac{1}{n_{i+1}}) + \prod_{i=1}^{m-2} (1 - \frac{1}{n_i}) \cdot (1 - \frac{1}{n_{m-1}}) \\
&= \cdots \\
&= \frac{1}{n_1} + (1 - \frac{1}{n_1}) \\
&= 1
\end{aligned}
$$

This completes the proof. Therefore, our approach satisfies ϵ-differential privacy. □

Since $n_m = 1$, we can get that the item added above $\prod_{i=1}^{m-1} (1 - \frac{1}{n_i}) \cdot (1 - \frac{1}{n_m}) = 0$. This indicates that our allocation scheme makes full use of the total privacy budget.

Utility analysis. We theoretically prove that Algorithm 4.1 guarantees that the sanitized dataset $\widetilde{\mathcal{D}}$ is (δ, β)-useful for counting queries.

Theorem 4.3. *The result of Algorithm 4.1 is (δ, β)-useful for counting queries.* ■

Proof. Given any counting query Q that covers up to m distinct itemsets in the entire output domain, the accurate answer of Q over the input dataset $\mathcal{D}$ is $Q(\mathcal{D}) = \sum_{i=1}^{m} Q(I_i)$, where I_i is the itemset covered by Q; the answer of Q over $\widetilde{\mathcal{D}}$ is $Q(\widetilde{\mathcal{D}}) = \sum_{i=1}^{m} (Q(I_i) + N_i)$, where N_i is the noise added to I_i. By the definition of (δ, β)-usefulness, to prove Theorem 4.3 is to prove that with a probability $1 - \beta$,

$$
\begin{aligned}
|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})| &= |\sum_{i=1}^{m}(Q(I_i) + N_i) - \sum_{i=1}^{m} Q(I_i)| \\
&= |\sum_{i=1}^{m} N_i| \\
&\leq \sum_{i=1}^{m} |N_i| \\
&\leq \delta
\end{aligned}
$$

We have the following observations.

- For I_i such that $I_i \notin \mathcal{D} \cap I_i \notin \widetilde{\mathcal{D}}$, $N_i = 0$. Let the size of such I_i be $m' \leq m$.
- For I_i such that $I_i \in \mathcal{D} \cap I_i \in \widetilde{\mathcal{D}}$, $N_i \sim Lap(1/\bar{\epsilon})$, where $\bar{\epsilon} = \epsilon/2 + \widetilde{\epsilon}$.
- For I_i such that $I_i \notin \mathcal{D} \cap I_i \in \widetilde{\mathcal{D}}$, $N_i \sim Lap(1/\bar{\epsilon})$, where $\bar{\epsilon} = \epsilon/2 + \widetilde{\epsilon}$.
- For I_i such that $I_i \in \mathcal{D} \cap I_i \notin \widetilde{\mathcal{D}}$, $N_i \sim Lap(1/\phi) + \gamma$, where $\phi = \epsilon/(2 \cdot |InternalNodes(\mathcal{T})|) \leq \bar{\epsilon}$ (the smallest privacy budget used in the entire partitioning process) and $\gamma = \sqrt{2}C_2 \log_f |I|/\phi$ is introduced by the threshold in Algorithm 4.1 and Algorithm 4.2.

  Therefore, we need to prove that with probability $1 - \beta$,

$$
\begin{aligned}
\sum_{i=1}^{m} |N_i| = \sum_{i=1}^{m-m'} |N_i| \quad &\leq \sum_{i=1}^{m-m'} (|Y_i| + \gamma) \\
&\leq \sum_{i=1}^{m-m'} |Y_i| + (m - m') \cdot \gamma \\
&\leq \delta
\end{aligned}
$$

where Y_i is a random variable i.i.d from $Lap(1/\phi)$. If every $|Y_i| \leq \delta_1$ where $\delta_1 = \frac{\delta}{m-m'} - \gamma$, we have $\sum_{i=1}^{m} |N_i| \leq \delta$. Let us call the event that any single $|Yi| > \delta_1$ a

FAILURE. We can calculate

$$Pr[FAILURE] = 2\int_{\delta_1}^{\infty} \frac{\phi}{2} exp(-\phi x)dx = exp(-\phi\delta_1)$$

Since every Y_i is independent and identically distributed, we have

$$\begin{aligned} Pr[\sum_{i=1}^{m} |N_i| \leq \delta] &= Pr[\sum_{i=1}^{m-m'} |Y_i| \leq \delta - (m-m')\cdot\gamma] \\ &\geq (1 - Pr[FAILURE])^{m-m'} \\ &\geq (1 - exp(-\phi\delta_1))^{m-m'} \end{aligned}$$

In [126], it has been proven that

$$(1 - exp(-\phi\delta_1))^{m-m'} \geq 1 - (m-m')exp(-\phi\delta_1)$$

Therefore, we get

$$\begin{aligned} Pr[\sum_{i=1}^{m} |N_i| \leq \delta] &\geq 1 - (m-m')exp(-\phi\delta_1) \\ &\geq 1 - (m-m')exp(\phi\gamma - \frac{\phi\delta}{m-m'}) \end{aligned}$$

This completes the proof. □

Complexity analysis. The runtime complexity of Algorithm 4.1 and Algorithm 4.2 is $O(|\mathcal{D}| \cdot |\mathcal{I}|)$, where $|\mathcal{D}|$ is the number of records in the input dataset $\mathcal{D}$ and $|\mathcal{I}|$ the size of the item universe. The main computational cost comes from the distribution of records from a partition to its sub-partitions. The complexity of distributing the records for a single partition operation is $O(|\mathcal{D}|)$ because a partitioning can affect at most $|\mathcal{D}|$ records. According to Theorem 4.1, the maximum number of partitionings needed for the entire process is the number of internal nodes in the taxonomy tree

$\mathcal{T}$. For a taxonomy tree with a fan-out $f \geq 2$, the number of internal nodes is $\frac{|\mathcal{I}|-1}{f-1}$. Therefore, the overall complexity of our approach is $O(|\mathcal{D}| \cdot |\mathcal{I}|)$.

Applicability. It is worthwhile discussing the applicability of our approach in the context of *relational data*. The core of our idea is to limit the output domain by taking into consideration the underlying dataset. In this chapter, we propose a probabilistic top-down partitioning process based on a context-free taxonomy tree in order to adaptively narrow down the output domain. For relational data, (categorical) attributes are usually associated with taxonomy trees. Therefore, a similar probabilistic partitioning process could be used. The difference is that the partitioning process needs to be conducted by considering the correlations among multiple taxonomy trees. In this case, exponential mechanism could be used in each partition operation to choose an attribute to split. Different heuristics (e.g. information gain, gini index or max) could be used as the score function. Following the idea, we maintain that our idea could adapt existing deterministic sanitization techniques, such as TDS [48] and Mondrian [75], to satisfy differential privacy. This approach would outperform existing works [12], [39], [125], [126] on publishing relational data in the framework of differential privacy in terms of both utility and efficiency for the same reasons explained in this chapter.

4.4 Experimental Evaluation

In the experiments, we examine the performance of our algorithm in terms of *utility* for different data mining tasks, namely *counting queries* and *frequent itemset mining*, and *scalability* of handling large set-valued datasets. We compare our approach (*DiffPart*) with Dwork et al.'s method (introduced in Section 4.3.1 and referred to as *Basic* in the following) to show the significant improvement of *DiffPart* on both utility and scalability. The implementation was done in C++, and all experiments were conducted on an Intel Core 2 Duo 2.26GHz PC with 2GB RAM.

Table 4.2: Experimental dataset statistics.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	$max\|D\|$	$avg\|D\|$
MSNBC	989,818	17	17	1.72
STM	1,210,096	1,012	64	4.82

Two *real-life* set-valued datasets, *MSNBC* [1] and *STM* [2], are used in the experiments. *MSNBC* originally describes the URL categories visited by users in time order. We converted it into set-valued data by ignoring the sequentiality, where each record contains a set of URL categories visited by a user. *MSNBC* is of a small universe size. We deliberately choose it so that we can compare *DiffPart* to *Basic*. *STM* records the sets of subway and/or bus stations visited by passengers in Montréal area within a week. It is of a relatively high universe size, for which *Basic* (and the methods in [12], [39], [125], [126]) fails to sanitize. The characteristics of the datasets are summarized in Table 4.2, where $max|D|$ is the maximum record size and $avg|D|$ the average record size.

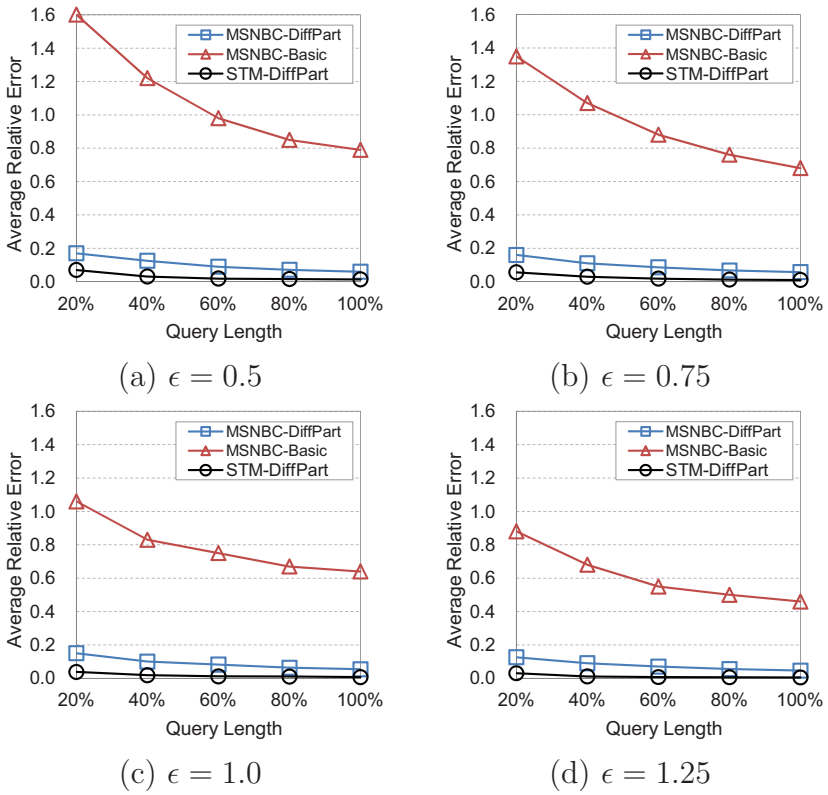
4.4.1 Data Utility

Following the evaluation scheme from previous works [125], [123], we measure the utility of a counting query Q over the sanitized dataset $\widetilde{\mathcal{D}}$ by its *relative error* with respect to the actual result over the raw dataset $\mathcal{D}$. Specifically, the relative error of Q is computed as $\frac{|Q(\widetilde{\mathcal{D}})-Q(\mathcal{D})|}{max\{Q(\mathcal{D}),s\}}$, where s is a *sanity bound* that weakens the influence of the queries with extremely small selectivities. In our experiments, s is set to 0.1% of the dataset size, the same as [125].

Counting Query. In our first set of experiments, we examine the relative error of counting queries with respect to different privacy budgets. For each dataset, we randomly generate $50,000$ counting queries with varying numbers of items. We call

[1] *MSNBC* is publicly available at UCI machine learning repository (http://archive.ics.uci.edu/ml/index.html).

[2] *STM* is provided by the *Société de transport de Montréal* (STM) (http://www.stm.info).



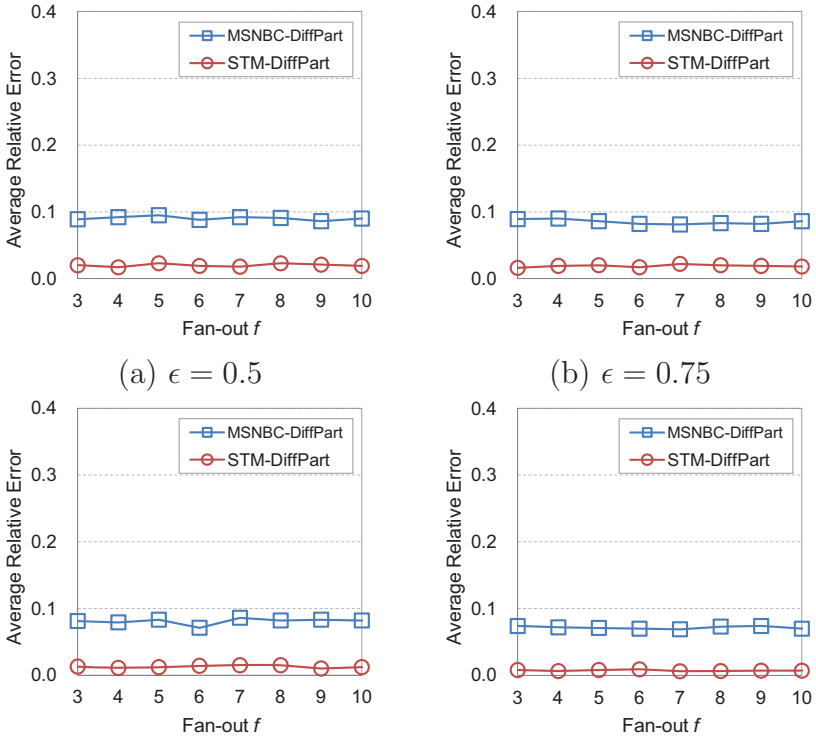
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 4.3: Average relative error vs. privacy budget.

the number of items in a query the *length* of the query. We divide the query set into 5 subsets such that the query length of the i-th subset is uniformly distributed in $[1, \frac{i \cdot max|D|}{5}]$ and each item is randomly drawn from $\mathcal{I}$. In the following figures, all relative error reported is the average of 10 runs.

Figure 4.3 shows the average relative error under varying privacy budget ϵ from 0.5 to 1.25 with fan-out $f = 10$ for each query subset. The X-axes represent the maximum query length of each subset in terms of the percentage of $max|D|$. The relative error decreases when the privacy budget increases because less noise is added. The error of *Basic* is significantly larger than that of *DiffPart* in all cases. When the query length decreases, the performance of *Basic* deteriorates substantially because the queries cover exponentially more itemsets that never appear in the original dataset and, therefore, contain much more noise. In contrast, our approach is more



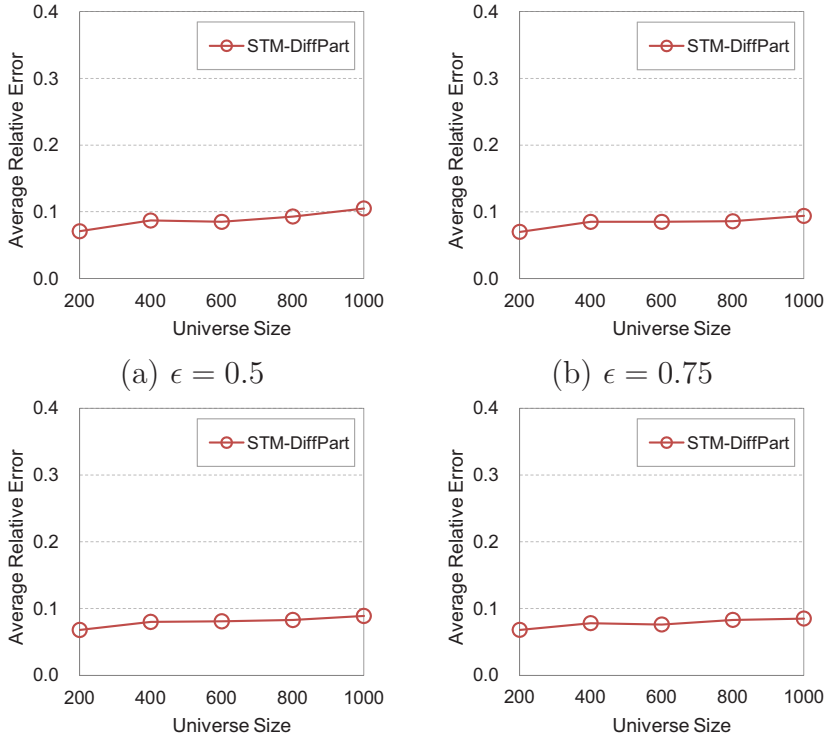
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 4.4: Average relative error vs. fan-out.

stable with different query lengths. It is foreseeable that queries with a length greater than $max|D|$ result in less error. In addition to better utility, *DiffPart* is more efficient than *Basic*, which fails to sanitize the STM dataset due to its large universe size.

Figure 4.4 illustrates the average relative error under different values of fan-out f with privacy budget ϵ ranging from 0.5 to 1.25 while fixing the query length to be $60\% \cdot max|D|$. In general, *DiffPart* generates relatively stable results for different fan-out values. For smaller fan-out values, each partitioning receives less privacy budget; however, there are more levels of partitionings, which increases the chance of pruning more empty partitions. The fact makes the relative error of smaller fan-out values comparable to that of larger fan-out values. The insensitivity of our approach to different fan-out values is a desirable property, which makes a data



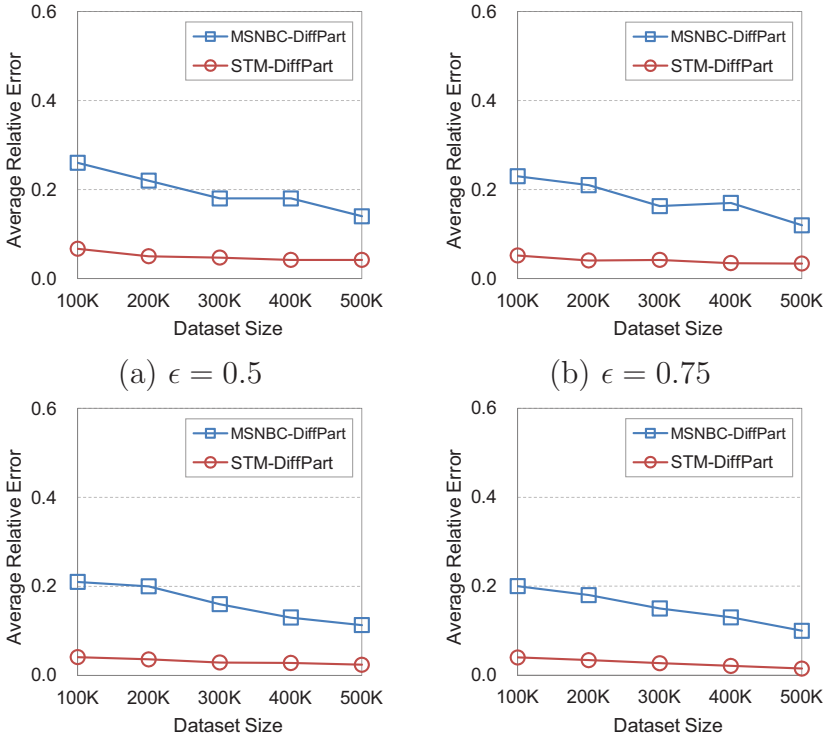
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $epsilon = 1.25$

Figure 4.5: Average relative error vs. universe size.

publisher easier to obtain a good release.

Figure 4.5 presents the average relative error under different universe sizes with privacy budget ϵ varying from 0.5 to 1.25. We set the fan-out $f = 10$ and fix the query length to 10 (we deliberately choose a small length to make the difference more observable). Since $MSNBC$ is of a small universe size, we only examine the performance of *DiffPart* on STM. We generate the test sets by limiting STM's universe size. After reducing the universe size, the sizes of the test sets also decrease. To make a fair comparison, we fix the dataset size under different universe sizes to 800,000. We can observe that the average relative error decreases when the universe size becomes smaller, because there is a greater chance to have more records falling into a partition, making the partition more resistant to larger noise. We can also observe that the datasets with smaller universe sizes obtain more stable relative



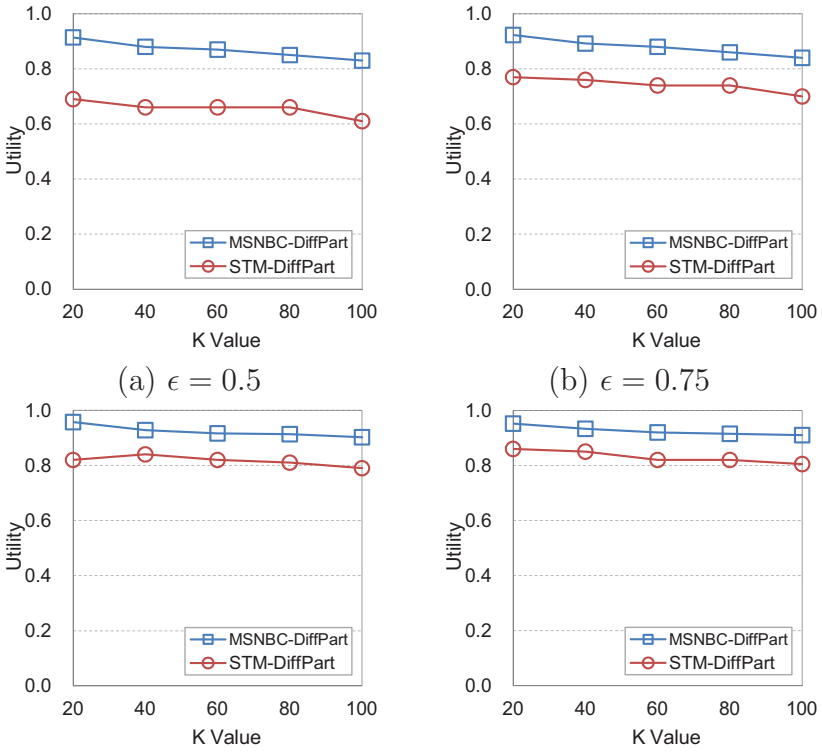
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 4.6: Average relative error vs. dataset size.

error under varying privacy budgets. This is due to the same reason that smaller universe sizes result in partitions with larger sizes, which are less sensitive to varying privacy budgets.

In theory, a dataset has to be large enough to obtain good utility under differential privacy. We experimentally study how the utility varies under different dataset sizes on the two real-life set-valued datasets. We generate the test datasets by randomly extracting records from the two datasets. The results are presented in Figure 4.6, where ϵ varies from 0.5 to 1.25, $f = 10$, and the query length is $60\% \cdot max|D|$. It can be observed that the two datasets behave differently to varying dataset sizes. The relative error of $MSNBC$ improves significantly when the privacy budget increases, while the change of STM's error is small. This indicates



(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 4.7: Utility for frequent itemset mining.

the fact that when the dataset size is not large enough, the distribution of the underlying records is key to the performance. In addition, we can observe that when the privacy budget is small, the error is more sensitive to the dataset size. It is because the number of records in a partition needs to be greater than the magnitude of noise (which is inversely proportion to the privacy budget) in order to obtain good utility.

Frequent itemset mining. We further validate the utility of sanitized data by frequent itemset mining, which is a more concrete data mining task. Given a positive number K, we calculate the top K most frequent itemsets on the raw dataset $\mathcal{D}$ and the sanitized dataset $\widetilde{\mathcal{D}}$, respectively, and examine their similarity. Let $F_K(\mathcal{D})$ denote the set of top K itemsets calculated from $\mathcal{D}$ and $F_K(\widetilde{\mathcal{D}})$ the set from $\widetilde{\mathcal{D}}$. For a frequent itemset $F_i \in F_K(\mathcal{D})$, let $sup(F_i, F_K(\mathcal{D}))$ denote its support in $F_K(\mathcal{D})$ and

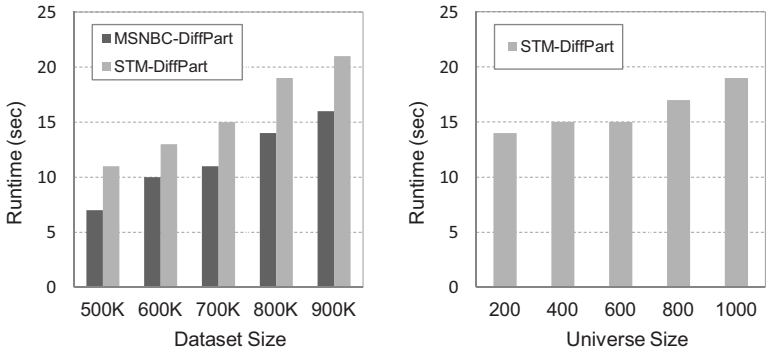
$sup(F_i, F_K(\widetilde{\mathcal{D}}))$ denote its support in $F_K(\widetilde{\mathcal{D}})$. If $F_i \notin F_K(\widetilde{\mathcal{D}})$, $sup(F_i, F_K(\widetilde{\mathcal{D}})) = 0$. We define the utility metric to be

$$1 - \frac{\sum_{F_i \in F_K(\mathcal{D})} \frac{|sup(F_i, F_K(\mathcal{D})) - sup(F_i, F_K(\widetilde{\mathcal{D}}))|}{sup(F_i, F_K(\mathcal{D}))}}{K},$$

where 1 means that $F_K(\mathcal{D})$ is identical to $F_K(\widetilde{\mathcal{D}})$ (even the support of every frequent itemset); 0 means that $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$ are totally different. Specifically, we employ *MAFIA* [3] to mine frequent itemsets.

In Figure 4.7, we study the utility of sanitized data for frequent itemset mining under different privacy budgets and different K values with $f = 10$. We observe two general trends from the experimental results. First, the privacy budget has a direct impact on frequent itemset mining. A higher budget results in better utility since the partitioning process is more accurate and less noise is added to leaf partitions. The differences of the supports of top K frequent itemsets between $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$ actually reflect the performance of *DiffPart* for counting queries of extremely small length (because the top-K frequent itemsets are usually of a small length). We can observe that the utility loss (the difference between $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$) is less than 30% except the case $\epsilon = 0.5$ for STM. Second, utility decreases when K value increases. When K value is small, in most cases the sanitized datasets are able to give the identical top-K frequent itemsets as the raw datasets, and the utility loss is mainly caused by the differences of the supports. When K value becomes larger, there are more false positives (itemsets wrongly included in the output) and false drops (itemsets mistakenly excluded), resulting in worse utility. Nevertheless, the utility loss is still less than 22% when $K = 100$ and $\epsilon \geq 1.0$ on both datasets.

[3] A maximal frequent itemset mining tool, available at http://himalaya-tools.sourceforge.net/Mafia/



(a) Runtime vs. $|\mathcal{D}|$ (b) Runtime vs. $|\mathcal{I}|$

Figure 4.8: Runtime vs. different parameters.

4.4.2 Scalability

We study the scalability of *DiffPart* over large datasets. According to the complexity analysis in Section 4.3.2, dataset size and universe size are the two factors that dominate the complexity. Therefore, we present the runtime of *DiffPart* under different dataset sizes and universe sizes in Figure 4.8. Figure 4.8.a presents the runtime of *DiffPart* under different dataset sizes. We generate the test sets in a similar setting to that of Figure 4.6 and set $\epsilon = 1.0$, $f = 10$. As expected, the runtime is linear to the dataset size. Figure 4.8.b studies how the runtime varies under different universe sizes, where $\epsilon = 1.0$ and $f = 10$. Since $MSNBC$ is of a small universe size, we only examine the runtime of *DiffPart* on STM. We generate the test sets in a similar setting to that of Figure 4.5. It can be observed again that the runtime scales linearly with the universe size. In summary, our approach scales well to large set-valued datasets. It takes less than 35 seconds to sanitize the STM dataset, whose $|\mathcal{D}| = 1,210,096$ and $|\mathcal{I}| = 1,012$.

4.5 Summary

In this chapter, we propose a probabilistic top-down partitioning algorithm for publishing set-valued data in the framework of differential privacy. Compared to the existing works on set-valued data publishing, our approach provides stronger privacy protection with guaranteed utility. Our work also contributes to the research of differential privacy by demonstrating that an efficient *data-dependent non-interactive* solution could be achieved by carefully making use of the underlying dataset. Our experimental results on real-life datasets demonstrate the effectiveness and efficiency of our approach.

Chapter 5

Trajectory Data Sanitization

5.1 Introduction

Recently, the prevalence of various location-aware devices, such as RFID tags, cell phones, GPS navigation systems, and point of sale terminals, has made trajectory data ubiquitous in various domains. The fact has stimulated extensive trajectory data mining research [52], [72], [73], resulting in many important real-life applications, such as city traffic management [81], homeland security [80], and location-based advertising [117].

Having access to high-quality trajectory data is the prerequisite for effective data mining. However, trajectory data often contain detailed information about individuals, and disclosing such information may reveal their lifestyles, preferences, and sensitive personal information. Moreover, for many applications, trajectory data need to be published with other attributes, including sensitive ones, thus incurring the privacy concern of inferring individuals' sensitive information via trajectory data. This emerging data publishing scenario, however, has not been well studied in existing works. Such privacy concerns often limit trajectory data holders' enthusiasm in providing data for further research and applications. Example 5.1.1 illustrates the potential privacy threats due to trajectory data publishing.

Table 5.1: Raw trajectory database T

Rec. #	**Path**	**Diagnosis**	**...**
1	$a1 \rightarrow d2 \rightarrow b3 \rightarrow e4 \rightarrow f6 \rightarrow e8$	HIV	...
2	$d2 \rightarrow c5 \rightarrow f6 \rightarrow c7 \rightarrow e9$	Fever	...
3	$b3 \rightarrow c7 \rightarrow e8$	Hepatitis	...
4	$b3 \rightarrow e4 \rightarrow f6 \rightarrow e8$	Flu	...
5	$a1 \rightarrow d2 \rightarrow c5 \rightarrow f6 \rightarrow c7$	HIV	...
6	$c5 \rightarrow f6 \rightarrow e9$	Hepatitis	...
7	$f6 \rightarrow c7 \rightarrow e8$	Fever	...
8	$a1 \rightarrow d2 \rightarrow f6 \rightarrow c7 \rightarrow e9$	Flu	...

Example 5.1.1. A hospital has employed a RFID patient tagging system in which patients' trajectory data, personal data, and medical data are stored in a central database [97]. The hospital intends to release such data (Table 5.1) to data miners for research purposes. A trajectory is a sequence of spatio-temporal *pairs* in the form of $(loc_i t_i)$. For example, *Record*#3 indicates that the tagged patient visited locations b, c, and e at timestamps 3, 7, and 8, respectively, and has hepatitis (other information is omitted for the purpose of illustration). With adequate background knowledge, an adversary can perform two kinds of privacy attacks on the trajectory database.

Identity linkage attack: If a trajectory in the database is so specific that not many patients can match it, there is a chance that with the help of background knowledge an adversary could uniquely identify the victim's record and, therefore, his sensitive information. Suppose an adversary knows that the record of the target victim, Claude, is in Table 5.1, and that Claude visited locations d and e at timestamps 2 and 4, respectively. The adversary can associate *Record*#1 with Claude and in turn identify Claude as an HIV patient because *Record*#1 is the only record containing both $d2$ and $e4$.

Attribute linkage attack: If a sensitive value occurs frequently with some sequences of pairs, it is possible to infer the sensitive value from these sequences even

though the record of the victim cannot be uniquely identified. Suppose the adversary knows that another victim, Bill, visited $a1$ and $f6$. The adversary can infer that Bill has HIV with $2/3 = 67\%$ confidence because two of the three records ($Records\#1, 5, 8$) containing $a1$ and $f6$ have the sensitive value HIV. ■

A trajectory database (e.g., Table 5.1) may contain other attributes, such as gender, age, and nationality. Although they are not explicit identifiers, an adversary may utilize combinations of these attributes as quasi-identifiers (QID) to identify the records and sensitive information of target victims. To thwart privacy threats due to QIDs, many privacy models, such as k-anonymity [105], ℓ-diversity [85], and confidence bounding [114], have been proposed in the context of relational data. These privacy models are effective for relational data anonymization; however, they fail to address the new challenges of trajectory data anonymization, as described below.

High dimensionality: Trajectory data are usually high-dimensional and cannot be effectively handled by traditional k-anonymity and its extensions due to the *curse of high dimensionality* [3]. Consider a transit system with 300 stations operating 24 hours a day. The corresponding trajectory database would have $300 \times 24 = 7200$ dimensions, because a trajectory could be represented in a tabular format with 7200 attributes filled with 0/1 values. Since k-anonymity and its extensions require every trajectory to be shared by at least k records and/or impose the diversity of sensitive values in every trajectory group, most data have to be suppressed in order to meet these kinds of restrictive privacy requirements.

Sparseness: Trajectory data are usually sparse. Consider passengers in transit systems. Among all available locations and all possible timestamps, they may visit only a few locations at a few timestamps, making the trajectory of each individual relatively short. Anonymizing such short trajectories in a high-dimensional space poses great challenges for traditional anonymization techniques because the trajectories may have little overlap. Enforcing k-anonymity could lower the data utility

Table 5.2: $(2, 50\%)_2$-privacy preserved database T'

Rec. #	**Path**	**Diagnosis**	**...**
1	$b3 \rightarrow e4 \rightarrow f6 \rightarrow e8$	HIV	...
2	$d2 \rightarrow c5 \rightarrow f6 \rightarrow c7 \rightarrow e9$	Fever	...
3	$c7 \rightarrow e8$	Hepatitis	...
4	$b3 \rightarrow e4 \rightarrow f6 \rightarrow e8$	Flu	...
5	$d2 \rightarrow c5 \rightarrow f6 \rightarrow c7$	HIV	...
6	$c5 \rightarrow f6 \rightarrow e9$	Hepatitis	...
7	$f6 \rightarrow c7 \rightarrow e8$	Fever	...
8	$d2 \rightarrow f6 \rightarrow c7 \rightarrow e9$	Flu	...

significantly.

Sequentiality: Time contains important information for trajectory data mining, but it also brings new privacy threats. Consider two trajectories $b3 \rightarrow e6$ and $e3 \rightarrow b6$. They have the same locations and timestamps but in a different order and, thus, are different from each other. An adversary could exploit such difference in order to increase the chance of a successful linkage attack. Therefore, traditional k-anonymity is not applicable to trajectory data, and anonymizing trajectory data requires additional efforts.

Trade-off between privacy and utility. One common assumption of k-anonymity and its extensions is that an adversary may use any or *even all* attributes in QIDs to perform linkage attacks. Yet this common assumption may be overly restrictive in the context of trajectory data. In a real-life attack, it is very unlikely that an adversary can identify all the visited locations along with the timestamps of a victim because it requires significant efforts to collect every piece of such background information. If the adversary is able to learn all such information, it is also possible that he can learn the victim's sensitive information. Thus, in the context of trajectory data, it is reasonable to derive a practical privacy model based on the assumption that an adversary's background knowledge on a target victim is bounded by at most L location-time pairs. We call such bounded background knowledge *L-knowledge*.

Based on this observation, we propose a new privacy model called $(K, C)_L$-*privacy* that requires any subsequence q of any adversary's L-knowledge to be shared by either 0 or at least K records in a trajectory database T and the confidence of inferring any sensitive value in S from q to be at most C, where L and K are positive integer thresholds, C is a real number threshold in the range of $[0, 1]$, and S is a set of sensitive values specified by the data holder. $(K, C)_L$-privacy guarantees that the probability of succeeding in an identity linkage attack is $\leq 1/K$ and the probability of succeeding in an attribute linkage attack is $\leq C$. Table 5.2 presents an example of an anonymous database satisfying $(2, 50\%)_2$-privacy from Table 5.1, in which every sequence q with maximum length 2 is shared by at least 2 records and the confidence of inferring any sensitive value in $S = \{HIV, Hepatitis\}$ from q is $\leq 50\%$.

Protecting privacy is one aspect of anonymizing trajectory data. Another aspect is preserving data utility in the anonymous data for data mining. The anonymized data may be used for different data mining tasks; therefore, we propose a generic framework to accommodate different utility requirements. As an illustration, in this chapter we aim to preserve both instances of location-time pairs and frequent sequences in a trajectory database. The ratio of suppressed instances is a general measure of anonymized data quality for a wide range of trajectory data mining tasks [72], [73]; the ratio of suppressed frequent sequences is a direct indication of anonymized data quality for trajectory pattern mining [52].

Generalization, *bucketization*, and *suppression* are the most widely used anonymization mechanisms. Generalization requires the use of taxonomy trees, which are highly specific to a particular application [4]. In many trajectory data applications, such domain specific taxonomy trees are not available. This fact largely hinders generalization's applicability on trajectory data anonymization. Bucketization merely breaks the correlation between trajectory data and sensitive attributes, and publishes trajectory data without any modification, which fails to protect identity linkage attacks on trajectory data. In addition, a *condensation* approach [4] is

proposed for multi-dimensional data publishing. However, it does not prevent from attribute linkage attacks in general. Specifically, for trajectory data, its complexity grows exponentially due to the high dimensionality. Furthermore, there lacks a way of measuring the similarity of trajectories, which is essential to the condensation approach. Therefore, in our solution, we employ suppression, both local and global suppressions, to eliminate privacy threats from a trajectory database. The introduction of local suppression results in significant data utility improvements for trajectory data anonymization. In global suppression, if a location-time pair p is selected to be suppressed from a trajectory database T, then all instances of p are removed from T, whereas in local suppression, some instances of p may remain intact in T while other instances are removed. Global suppression punishes all records containing p even if the privacy leakage is caused by only one instance of p in one record. In contrast, local suppression eliminates the exact instances that cause privacy breaches without penalizing others. Thus, local suppression preserves much better data utility compared to global suppression.

Contributions. In this chapter, we acknowledge the emerging data publishing scenario, in which trajectory data need to be published with sensitive attributes. This naturally requires the prevention from both identity linkage attacks and attribute linkage attacks, which has not been studied in existing works. Based on the practical assumption that an adversary has only limited background knowledge on a target victim, we propose the $(K, C)_L$-privacy model for trajectory data anonymization, which takes into consideration not only identity linkage attacks on trajectory data, but also attribute linkage attacks via trajectory data. We present an anonymization framework that supports both local suppression and global suppression with the goal of preserving data utility for data mining. This is, to the best of our knowledge, the first study introducing local suppression to trajectory data anonymization. In this section, we tailor our anonymization framework to preserve both instances of location-time pairs and frequent sequences in trajectory data. The framework itself

is open to different data mining workloads by incorporating different data utility metrics. We provide comprehensive experimental evaluations on both synthetic and real-life trajectory datasets. The experimental results demonstrate that our proposed algorithm is both effective and efficient to address the special challenges in trajectory data anonymization. In particular, local suppression is shown to be essential to enhance the resulting data utility when combined with $(K, C)_L$-privacy. The results of this chapter have been published in [23].

5.2 Problem Definition

5.2.1 Trajectory Database

A typical trajectory system generates a sequence of sensory data records of the general form $\langle ID, loc, t\rangle$, where each record indicates that the record owner (or the object) having the unique identifier ID was detected in location loc at time t. For example, in transportation systems, a record represents that a passenger was present in station loc at time t, where ID could be the passenger's transportation card number. Different types of trajectory data can be easily converted into the general form by pre-processing steps. For example, GPS data, a typical type of trajectory data, is of the form $\langle ID, (X\ coordinate, Y\ coordinate), timestamp\rangle$, which can be converted by substituting the grid ID/name containing a point for $(X\ coordinate, Y\ coordinate)$. By selecting a proper granularity of a location, this general form is suitable to represent various kinds of trajectory data for different data mining tasks.

The trajectory of a specific record owner, representing the owner's movement history, is composed of a sequence of (loc, t) *pairs*. A *trajectory*, denoted by $(loc_1t_1) \rightarrow \cdots \rightarrow (loc_nt_n)$, can be constructed by grouping the sensory data records $\langle ID, loc, t\rangle$ by ID and sorting them by the timestamps. The timestamps in a trajectory are always increasing.

In addition to trajectory data, a *trajectory database* may also contain other attributes that are associated with the record owners. Formally, a trajectory database contains a collection of data records in the form of

$$(loc_1 t_1) \rightarrow \cdots \rightarrow (loc_n t_n) : s_1, \ldots, s_p : d_1, \ldots, d_m$$

where $(loc_1 t_1) \rightarrow \cdots \rightarrow (loc_n t_n)$ is a trajectory, $s_i \in S_i$ are the sensitive attributes with values from the domain S_i, and $d_i \in D_i$ are the quasi-identifiers (QIDs) of the record owner with the values from the domain D_i. Given a trajectory database, an adversary can perform privacy attacks via either trajectories or QID attributes. Anonymization on relational QID attributes has been extensively studied in previous works [48], [75], [85], [105], [124]. This chapter focuses on addressing the privacy threats posed by trajectories.

5.2.2 Privacy Threats

Suppose a data holder wants to publish a trajectory database T to some recipients for data mining. Explicit identifiers, e.g., name, SSN, and ID, have been removed. One recipient, the adversary, seeks to identify the record or sensitive values of some target victim V in T. As explained before, we assume that the adversary knows at most L spatio-temporal pairs that the victim V has previously visited. Such background knowledge about the victim V is denoted by $\kappa_V = (loc_1 t_1) \rightarrow \cdots \rightarrow (loc_z t_z)$, where $z \leq L$. Using the background knowledge κ_V, the adversary could identify a group of records in T, denoted by $T(\kappa_V)$, that "matches" κ_V. A record *matches* κ_V if κ_V is a subsequence of the trajectory in the record. For example, in Table 5.1, if $\kappa_V = d2 \rightarrow e4$, then *Record*#1 matches κ_V, but *Record*#2 does not. Given the background knowledge κ_V, an adversary could identify and utilize $T(\kappa_V)$ to perform two types of privacy attacks:

1. *Identity linkage attack*: $T(\kappa_V)$ is a set of candidate records that contains the victim V's record. If the group size of $T(\kappa_V)$, denoted by $|T(\kappa_V)|$, is small,

then the adversary may identify V's record from $T(\kappa_V)$ and, therefore, V's sensitive value.

2. *Attribute linkage attack*: Given $T(\kappa_V)$, the adversary may infer that V has sensitive value s with confidence $Conf(s|T(\kappa_V)) = \frac{|T(\kappa_V \bigcup s)|}{|T(\kappa_V)|}$, where $T(\kappa_V \bigcup s)$ denotes the set of records containing both κ_V and s. $Conf(s|T(\kappa_V))$ is the percentage of the records in $T(\kappa_V)$ containing s. The privacy of V is at risk if $Conf(s|T(\kappa_V))$ is high.

   Example 5.1.1 illustrates these two types of attacks.

5.2.3 Privacy Requirement

An adversary's background knowledge κ could be any non-empty subsequence q with $|q| \leq L$ of any trajectory in the trajectory database T. Intuitively, $(K, C)_L$-*privacy* requires that every subsequence q with $|q| \leq L$ in T is shared by at least a certain number of records, and that the confidence of inferring any sensitive value via q cannot be too high.

Definition 5.1 ($(K, C)_L$-privacy)**.** Let L be the maximum length of the background knowledge. Let S be a set of sensitive values of the sensitive attributes of a trajectory database T selected by the data holder. T satisfies $(K, C)_L$-*privacy* if and only if for any subsequence q in T with $0 < |q| \leq L$,

1. $|T(q)| \geq K$, where K is a positive integer specifying the anonymity threshold, *and*

2. $Conf(s|T(q)) \leq C$ for any $s \in S$, where $0 \leq C \leq 1$ is a real number specifying the confidence threshold. ■

The $(K, C)_L$-privacy model has several desirable properties. First, it is a generalized version of several existing privacy models: k-anonymity [105] is a special case

of the $(K, C)_L$-privacy model with $L = |d|$ and $C = 100\%$, where $|d|$ is the number of dimensions in a given database. ℓ-diversity [85] is a special case of $(K, C)_L$-privacy model with $L = |d|$, and $\ell = 1/C$. Confidence bounding [114] is a special case of the $(K, C)_L$-privacy model with $L = |d|$ and $K = 1$. (α, k)-anonymity [121] is also a special case of $(K, C)_L$-privacy with $L = |d|$, $K = k$, and $C = \alpha$. Second, it is intuitive for a data holder to impose different types and levels of privacy protection by specifying different L, K, and C thresholds.

It is worth noting that $(K, C)_L$-privacy is a stronger privacy notion than other existing privacy models for trajectory data [1], [100], [109], [130] in the sense that $(K, C)_L$-privacy thwarts both identity linkages on trajectory data and attribute linkages via trajectory data. It is vital to thwart attribute linkage attacks in trajectory data publishing because more and more trajectory data mining tasks will resort to both trajectory data and other personal information. For example, Utsunomiya et al. [112] conducted an interesting passenger classification analysis using both passengers' trajectory data and personal information. A recent investigation [99] further indicates that there is a need to enrich trajectory data by incorporating sociodemographic data for data mining tasks.

5.2.4 Utility Requirement

Since we aim at presenting a framework that allows the adoption of various data utility metrics for different data mining tasks, we illustrate the preservation of two different kinds of utility metrics, both *instances of location-time pairs* and *frequent sequences* in a trajectory database. The ratio of suppressed instances is a general measure of the usefulness of anonymized data for a wide range of trajectory data mining tasks [72], [73]. In addition, previous works [48], [79] suggest that anonymization algorithms can be tailored to better preserve utility if the utility requirement is known in advance. We also preserve frequent sequences specifically for trajectory

pattern mining [52]. However, extracting all possible frequent sequences in a trajectory database is computationally expensive. It is even exacerbated when dealing with large datasets with long frequent sequences because all subsequences of a frequent sequence are also frequent. A more feasible solution is to preserve *maximal frequent sequences* (*MFS*).

Definition 5.2 (Maximal frequent sequence)**.** For a given minimum support threshold $K' > 0$, a sequence q is *maximal frequent* in a trajectory database T if q is frequent and no super sequence of q is frequent in T. ■

The set of MFS in T, denoted by $U(T)$, is much smaller than the set of frequent sequences (FS) in T given the same K', but still contains the essential information of FS. Any subsequence of an MFS is also an FS. Once all the MFS have been determined, the support count of any particular FS can be computed by scanning $U(T)$ once.

We emphasize that although in this section we aim at preserving instances and MFS, the $(K, C)_L$-privacy model and the anonymization framework presented in Section 5.3 are independent of the underlying utility metric and are flexible enough to serve other utility requirements. The only change is to replace the greedy function guiding the anonymization process, which will be further explained in Section 5.3.

5.2.5 Problem Statement

To achieve $(K, C)_L$-privacy for a given trajectory database T, our proposed framework conducts a sequence of local and global suppressions to remove all privacy threats from T while preserving as much data utility as possible. *Global suppression* eliminates *all* instances of a pair p from T if some instances of p cause privacy breaches, while *local suppression* eliminates only the instances of p that cause privacy breaches and leaves others intact. Finding an optimal solution based on suppression for $(K, C)_L$-privacy, however, is *NP-hard* (see Section 5.3 for proof). Thus, we

propose a greedy algorithm to efficiently identify a reasonably "good" sub-optimal solution.

Definition 5.3 (Trajectory data anonymization)**.** Given a trajectory database T, a $(K, C)_L$-privacy requirement, a utility metric, and a set of sensitive values S, the task of *trajectory data anonymization* is to generate a transformed version of T that satisfies $(K, C)_L$-privacy while maintaining the maximum utility with respect to the utility metric by a sequence of local and global suppressions. ■

5.3 Anonymization Algorithm

Our proposed anonymization algorithm consists of two phases. First, identify all violating sequences that breach a given $(K, C)_L$-privacy requirement in a trajectory database. Second, perform a sequence of local and global suppressions to anonymize the trajectory database while maintaining as much data utility as possible.

5.3.1 Identifying Violating Sequences

An adversary may use any non-empty sequence with length not greater than L as background knowledge to launch a linkage attack. Thus, given a $(K, C)_L$-privacy requirement, any subsequence q with $0 < |q| \leq L$ in a trajectory database T is a *violating sequence* if its group $T(q)$ does not satisfy Condition 1, Condition 2, or both in $(K, C)_L$-privacy in Definition 5.1.

Definition 5.4 (Violating sequence)**.** Let q be a subsequence of a trajectory in T with $0 < |q| \leq L$. q is a *violating sequence* with respect to a $(K, C)_L$-privacy requirement if $|T(q)| < K$ *or* $Conf(s|T(q)) > C$ for any sensitive value $s \in S$. ■

Example 5.3.1. Given $L = 2$, $K = 2$, $C = 50\%$, and the sensitive value set $S = \{HIV, Hepatitis\}$. In Table 5.1, the sequence $q_1 = a1 \rightarrow b3$ is a violating sequence because $|T(q_1)| = 1 < K$; the sequence $q_2 = a1 \rightarrow d2$ is also a violating

sequence because $Conf(HIV|T(q_2)) = 2/3 = 67\% > C$. However, the sequence $q_3 = b3 \rightarrow c7 \rightarrow e8$ is *not* a violating sequence even though $|T(q_3)| = 1 < K$ and $Conf(Hepatitis|T(q_3)) = 100\% > C$ because $|q_3| = 3 > L$. ■

To satisfy a given $(K, C)_L$-privacy requirement on a trajectory database T, it is sufficient if all violating sequences in T with respect to the privacy requirement are removed, because all possible channels for identity and attribute linkages are eliminated. A naive approach is to first enumerate all possible violating sequences and then remove them. This approach is infeasible because of the huge number of violating sequences. Consider a violating sequence q with $|T(q)| < K$. Any super sequence of q, denoted by q'', with $|T(q'')| > 0$ in T is also a violating sequence because $|T(q'')| \leq |T(q)| < K$. To overcome the bottleneck of violating sequence enumeration, our insight is that a few "minimal" violating sequences exist among the violating sequences, and it is sufficient to achieve $(K, C)_L$-privacy by removing *only* the minimal violating sequences.

Definition 5.5 (Minimal violating sequence)**.** A violating sequence q is a *minimal violating sequence* (MVS) if every proper subsequence of q is not a violating sequence. ■

Example 5.3.2. Given $L = 2$, $K = 2$, $C = 50\%$, and $S = \{HIV, Hepatitis\}$. In Table 5.1, the sequence $q_1 = d2 \rightarrow e4$ is an MVS because $|T(q_1)| = 1 < K$, and none of its proper subsequences, $d2$ and $e4$, is a violating sequence. In contrast, the sequence $q_2 = a1 \rightarrow d2$ is a violating sequence, but *not* an MVS, because one of its proper subsequences, $a1$, is a violating sequence. ■

The set of MVS is much smaller than the set of violating sequences; therefore, we can efficiently identify all privacy threats by generating all MVS. A trajectory database T satisfies $(K, C)_L$-privacy if and only if T contains no MVS.

Theorem 5.1. *A trajectory database T satisfies $(K, C)_L$-privacy if and only if T contains no minimal violating sequence.* ■

Algorithm 5.1: Identify Minimal Violating Sequences (MVS)

Input: Raw trajectory database T
Input: Thresholds L, K and C
Input: Sensitive values S
Output: Minimal violating sequences $V(T)$

1: $C_1 \leftarrow$ all distinct pairs in T;
2: $i = 1$;
3: **while** $i \leq L$ and $C_i \neq \emptyset$ **do**
4: Scan T once to compute $|T(q)|$ and $Conf(s|T(q))$, for $\forall q \in C_i$, $\forall s \in S$;
5: **for** each sequence $q \in C_i$ with $|T(q)| > 0$ **do**
6: **if** $|T(q)| \geq K$ and $Conf(s|T(q)) \leq C$ for all $s \in S$ **then**
7: Add q to U_i;
8: **else**
9: Add q to V_i;
10: **end if**
11: **end for**
12: i++;
13: Generate candidate set C_i by $U_{i-1} \bowtie U_{i-1}$;
14: **for** each sequence $q \in C_i$ **do**
15: **if** q is a super sequence of any $v \in V_{i-1}$ **then**
16: Remove q from C_i;
17: **end if**
18: **end for**
19: **end while**
20: **return** $V(T) = V_1 \cup \cdots \cup V_{i-1}$;

Proof. Suppose a database T does not satisfy $(K, C)_L$-privacy even if T contains no MVS. By Definition 5.1, T must contain some violating sequences. According to Definition 5.5, a violating sequence must be an MVS itself or contain an MVS, which contradicts the initial assumption. Therefore, T must satisfy $(K, C)_L$-privacy. □

Hence, our first step is to efficiently identify all the MVS, $V(T)$, in the given trajectory database T. Algorithm 5.1 presents the details of generating $V(T)$. Based on Definition 5.5, we generate all MVS of size $i+1$, denoted by V_{i+1}, by incrementally extending non-violating sequences of size i, denoted by U_i, with an additional pair. *This needs to take into consideration the sequentiality of trajectory data.* Line 1 loads all distinct pairs in T as the initial candidate set C_1. Line 4 scans T once to compute

$|T(q)|$ and $Conf(s|T(q))$ for every sequence $q \in C_i$, and for every sensitive value $s \in S$. If a sequence q is not violating, it is added to the non-violating sequence set U_i for generating the next candidate set C_{i+1} (Line 7); otherwise, q is added to the MVS set (Line 9). The next candidate set C_{i+1} is generated in two steps. First, conduct a self-join of U_i (Line 13). Second, remove all super sequences of the identified MVS from C_{i+1} (Lines 14-18). The second step significantly reduces the minimal violating sequence search space. Two sequences $q_x = (loc_1^x t_1^x) \rightarrow \cdots \rightarrow (loc_i^x t_i^x)$ and $q_y = (loc_1^y t_1^y) \rightarrow \cdots \rightarrow (loc_i^y t_i^y)$ can be joined if the first $i-1$ pairs are identical and $t_i^x < t_i^y$. The joined result is $(loc_1^x t_1^x) \rightarrow \cdots \rightarrow (loc_i^x t_i^x) \rightarrow (loc_i^y t_i^y)$. The definition of join-compatibility makes sure that every potential candidate sequence would be generated exactly once.

Example 5.3.3. Given $L = 2$, $K = 2$, $C = 50\%$, and the sensitive value set $S = \{HIV, Hepatitis\}$, the MVS set generated from Table 5.1 is $V(T) = \{a1, d2 \rightarrow b3, d2 \rightarrow e4, d2 \rightarrow e8, b3 \rightarrow c7\}$. ■

5.3.2 Removing Violating Sequences

The second step is to remove all identified minimal violating sequences using suppression with the goal of preserving as much data utility as possible. However, finding an optimal solution is *NP-hard*.

Theorem 5.2. *Given a trajectory database T and a $(K, C)_L$-privacy requirement, it is NP-hard to find the optimal anonymization solution.* ■

Proof. The problem of finding the optimal anonymization solution can be converted into the *vertex cover problem* [27]. The vertex cover problem is a well-known problem in which, given an undirected graph $G = (V, E)$, it is NP-hard to find the smallest set of vertices S such that each edge has at least one endpoint in S. To reduce our problem into the vertex cover problem, we only consider the set of MVS of length 2. Then, the set of candidate pairs represents the set of vertices V and the set of

MVS is analogous to the set of edges E. Hence, the optimal vertex cover, S, means finding the smallest set of candidate pairs that must be suppressed to obtain the optimal anonymous dataset T'. Given that it is NP-hard to determine S, it is also NP-hard to find the optimal set of candidate pairs for suppression. □

Therefore, we propose a greedy algorithm that employs both local and global suppressions to eliminate all identified MVS, $V(T)$, with respect to the given $(K, C)_L$-privacy requirement in order to efficiently identify a reasonably "good" solution. Generally, suppressing a pair p from $V(T)$ increases privacy and decreases data utility. So our goal is to design a greedy function, $Score(p)$, that guides us to find the sub-optimal trade-off between privacy and data utility. In this section, we define our greedy function as follow:

$$Score(p) = \frac{PrivGain(p)}{UtilityLoss(p) + 1}$$

where $PrivGain(p)$ is the number of MVS that can be eliminated by suppressing p, and $UtilityLoss(p)$ is the number of either instances or MFS that are lost due to suppressing p, depending on the given utility metric. Since suppressing p may not cause utility loss in terms of MFS, we add 1 to the denominator to avoid the division by zero error. The function considers both privacy and utility simultaneously by selecting the anonymization operation with the maximum privacy gain per unit of utility loss. Considering only privacy gain or utility loss would lead to inferior performances according to our tests. Again, our anonymization algorithm is independent of the underlying data utility metric. To optimize the data utility for other data mining workloads, we can simply re-design the meaning of $UtilityLoss(p)$.

A key to an efficient solution is to ensure that no new MVS will be generated in the anonymizing process. Upon satisfying the requirement, the identified MVS $V(T)$ always decreases monotonically. A suppression-based algorithm is guaranteed to achieve $(K, C)_L$-privacy within less than $|V(T)|$ iterations. One nice property of

global suppression is that it does not generate any new MVS during the anonymizing process.

Theorem 5.3. *A global suppression does not generate any new minimal violating sequence with respect to a* $(K, C)_L$*-privacy requirement.* ■

Proof. Suppose a pair p is globally suppressed from a given trajectory database T. The database after the global suppression is denoted by T'.

- For any sequence q in T not containing an instance of p, we have $|T'(q)| = |T(q)|$ and $Conf(s|T'(q)) = Conf(s|T(q))$. Identically, for any subsequence q' of q, which does not contain p either, we have $|T'(q')| = |T(q')|$ and $Conf(s|T'(q')) = Conf(s|T(q'))$. So q cannot be a new minimal violating sequence in T'.

- For any sequence q in T that contains an instance of p, q no longer exists in T', so q cannot be a new minimal violating sequence.

Therefore, no sequence in T will become a new MVS in T'. □

However, *local suppression* does not share the same property. For example, locally suppressing $c7$ from $Record\#3$ in Table 5.1 will generate a new MVS $c7 \rightarrow e8$ because in the resulting database T', $|T'(c7 \rightarrow e8)| = 1 < K$. Identifying the values of all newly generated MVS requires expensive computational cost. Moreover, there is no guarantee that the anonymization algorithm can converge within a bounded number of iterations, $|V(T)|$. Therefore, it is beneficial to perform local suppressions only when no new MVS will be generated. Such a local suppression is called a *valid* local suppression.

Definition 5.6 (Valid local suppression)**.** A local suppression over a trajectory database is *valid* if it does not generate any *new* MVS. ■

Algorithm 5.2: Check if a local suppression is valid

Input: Trajectory database T
Input: Thresholds L, K, C, and sensitive values S
Input: A pair p in an MVS m
Output: A *boolean* value indicating if locally suppressing p from m is valid

1: $P \leftarrow$ distinct pair p' such that $p' \in T(m) \wedge p' \in (T(p) - T(m))$;
2: $V' \leftarrow$ all size-one MVS and the MVS containing p, $V(p)$;
3: Remove all pairs, except p, in V' from P;
4: $Q \leftarrow$ all possible sequences with size $\leq L$ generated from P after removing super sequences of the sequences in $V(T) - V(p)$;
5: Scan $T(p) - T(m)$ once to compute $|q|$ and $Conf(s|T(q))$ for each $q \in Q$ and for every sensitive value $s \in S'$, where S' is the subset of S in $T(p) - T(m)$;
6: **for** each sequence q with $|q| > 0$ **do**
7: **if** $|q| < K$ or $Conf(s|T(q)) > C$ for any $s \in S'$ **then**
8: **return** $false$;
9: **end if**
10: **end for**
11: **return** $true$;

An intuitive way to check if a local suppression is valid is to re-invoke Algorithm 5.1 and compare $V(T)$ and $V(T')$. However, it is extremely costly. Instead, Algorithm 5.2 presents an efficient approach to avoid the computational cost of calculating the values of all newly generated MVS. It significantly narrows down the checking space to a very small set of sequences that may be affected by a local suppression by carefully using the properties of MVS.

Theorem 5.4. *Algorithm 5.2 is sufficient to check if a local suppression is valid.* ■

Proof. Suppose a pair p in an MVS m is locally suppressed from a given trajectory database T. The resulting database is denoted by T'. For any sequence q in T not containing an instance of p, we have $|T'(q)| = |T(q)|$ and $Conf(s|T'(q)) = Conf(s|T(q))$. Identically, for any subsequence q' of q, we have $|T'(q')| = |T(q')|$ and $Conf(s|T'(q')) = Conf(s|T(q'))$. So q cannot be a new MVS in T'. If there is a new MVS, it must contain p. Since p is eliminated from the records containing m, $T(m)$, we only need to consider the sequences in $T(p)-T(m)$, where $T(p)$ denotes the records containing p. For a sequence q in T containing an instance of p, if $q \notin T(m)$,

we have $|T'(q)| = |T(q)|$ and $Conf(s|T'(q)) = Conf(s|T(q))$ and, therefore, such q cannot be a new MVS. q is possible to be a new MVS only if $q \in T(m)$ *and* $q \in (T(p) - T(m))$ (Line 1). Since we only care about *new* MVS, we could further filter out all identified MVS and their super sequences. For the remaining sequences, if none of them is a violating sequence, it is sufficient to ensure that there is no new MVS by Definition 5.5 (Line 4-11). □

Example 5.3.4. Consider Table 5.1 with $L = 2$, $K = 2$, $C = 50\%$, and the sensitive value set $S = \{HIV, Hepatitis\}$. For the local suppression of $d2$ in MVS $d2 \rightarrow e4$, we get $P = \{d2, f6\}$ and $V' = \{a1, d2 \rightarrow b3, d2 \rightarrow e4, d2 \rightarrow e8\}$. Since all sequences in $Q = \{d2, f6, d2 \rightarrow f6\}$ are not violating sequences, this local suppression is *valid*. ■

Algorithm 5.3 presents the entire anonymization algorithm. Line 1 calls Algorithm 5.1 to generate the MVS set $V(T)$. For preserving MFS, Line 2 is needed, which calls the MFS mining algorithm to build a MFS-tree with a UL table that keeps track of the occurrences of all candidate pairs in the MFS-tree. We adapt *MAFIA* [15], originally designed for mining maximal frequent itemsets, to mine MFS. For all instances of all pairs in $V(T)$, their scores for local and global suppressions are calculated and stored in *Score* table based on Algorithm 5.2 (Line 3). Different instances of a pair in $V(T)$ have different entries in *Score* table. Only valid local suppressions are assigned scores. The global suppression scores of all instances of a pair are the same. Lines 4-17 iteratively select a pair p with the highest score in *Score* table to suppress. According to whether the highest score is obtained from *local suppression* or *global suppression*, our algorithm performs different strategies. For local suppression, the algorithm identifies the set of MVS, denoted by V', that will be eliminated due to locally suppressing p, and removes the instances of p from the records $T(m)$. One extra step is performed for MFS to update the supports of MFS in the MFS-tree (Line 9). For global suppression, the algorithm removes all the MVS containing p, and suppresses all instances of p from T. For preserving

Algorithm 5.3: Trajectory Database Anonymizer

Input: Raw trajectory database T
Input: Thresholds L, K, C, (K')
Input: Sensitive values S
Output: Anonymous T' satisfying the given $(K, C)_L$-privacy requirement

1: Generate $V(T)$ by Algorithm 5.1;
2: Generate MFS by MFS algorithm and build MFS-tree;
3: Build *Score* table by Algorithm 5.2;
4: **while** *Score* table $\neq \emptyset$ **do**
5: Select a pair p with the highest score from its MVS m;
6: **if** p is obtained from *local suppression* **then**
7: $V' \leftarrow$ each MVS m' such that $p \in m' \wedge T(m') = T(m)$;
8: Suppress the instances of p from $T(m)$;
9: Delete the MFS containing p if their supports are $< K'$ after suppression, otherwise update their supports;
10: **else**
11: $V' \leftarrow V(p)$;
12: Suppress all instances of p in T;
13: Delete all MFS containing p from MFS-tree;
14: **end if**
15: Update the $Score(p')$ if both p and p' are in V' (or in the same MFS);
16: $V(T) = V(T) - V'$;
17: **end while**
18: **return** the suppressed T as T';

MFS, the MFS containing p are removed from the MFS tree (Line 13). Line 15 updates the *Score* table, which requires two tasks: 1) checking if the pairs affected by the current suppression are valid for future local suppressions; and 2) calculating the scores for such pairs. Specifically, for preserving MFS, a special data structure, *MFS-tree*, is created to facilitate the anonymization.

Definition 5.7 (MFS-tree)**.** MFS-tree is a tree structure that represents each MFS as a tree path from root to leaf. The support of each MFS is stored at its leaf node. Each node keeps track of a count of MFS sharing the same prefix. The count at the root is the total number of MFS. MFS-tree has a UL table that keeps the total occurrences of every candidate pair p. Each candidate pair p in the UL table has a link, denoted by $Link_p$, that links up all the nodes in MFS-tree containing p. ■

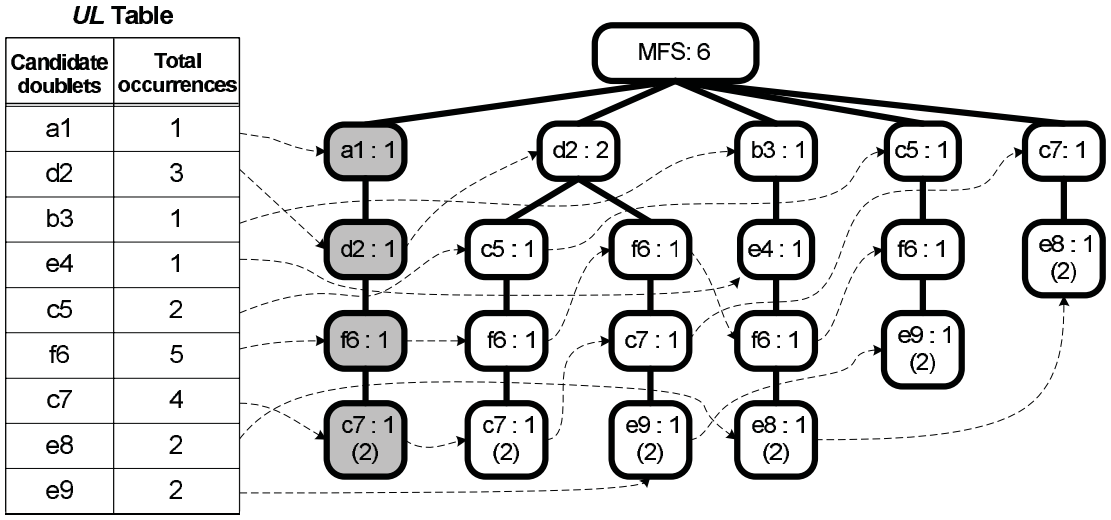


Figure 5.1: MFS-tree for efficient *Score* updates

Example 5.3.5. Figure 5.1 presents the MFS-tree generated from Table 5.1 with $K' = 2$. To find all the MFS containing $f6$, simply follow $Link_{f6}$, starting from the $f6$ entry in the UL table. ■

5.3.3 Complexity Analysis

Our anonymization algorithm consists of two steps. In the first step, we identify all MVS. The most expensive operation is scanning the raw trajectory database T once for all sequences in each candidate set C_i. The cost is $\sum_{i=1}^{L} |C_i|i$, where $|C_i|$ is the size of candidate set C_i. The size of C_1 is the number of distinct pairs in T whose upper limit is $|d|$, the number of dimensions. Since C_2 is generated by self-joining all pairs in U_1, whose size is less than or equal to $|C_1|$, its upper bound is $|d|(|d|-1)/2$. However, when $i \geq 3$, the sizes of the candidate sets do not increase significantly for two reasons: 1) All candidates are generated by self-joining, which requires that only if two sequences share the same prefix, their resulting sequence can be considered a future candidate. When i is relatively large, the chance of finding two such sequences decreases significantly. 2) The pruning process in Algorithm 5.1 also greatly reduces the candidate search space. Therefore, a good approximation is $C \approx |d|^2$. However,

in the worst case, the computational cost of the first step is bounded by $O(|d|^L|T|)$, where $|T|$ is the number of records in T. In the second step, we construct the *Score* table, and then remove all MVS iteratively. The most costly operation is to check if the instances of the pairs in $V(T)$ are valid to be locally suppressed. The number of instances of pairs in $V(T)$ is less than $\sum_{i=1}^{L} |C_i|i$, and thus also bounded by $|d|^L$. For every instance in $V(T)$, we need to invoke Algorithm 5.2 *at most* twice. For each invocation, in the worst case, it has to go through all records in T. So the cost of the second step is still bounded by $O(|d|^L|T|)$. By incorporating both steps, the complexity of the entire algorithm is $O(|d|^L|T|)$. The scalability of our algorithm is further demonstrated in Section 5.4.2.

5.4 Experimental Evaluation

In this section, we examine the performance of our anonymization framework in terms of *utility loss* due to the anonymization and *scalability* for handling large datasets. For preserving *instances*, the utility loss is defined as $\frac{N(T)-N(T')}{N(T)}$, where $N(T)$ and $N(T')$ are the numbers of instances of pairs in the original dataset T and the anonymous dataset T', respectively; for preserving *MFS*, the utility loss is defined as $\frac{|U(T)|-|U(T')|}{|U(T)|}$, where $|U(T)|$ and $|U(T')|$ are the numbers of MFS in T and T', respectively. The formulas respectively measure the percentage of instances and MFS that are lost due to suppressions. Lower utility loss implies better resulting data quality. We cannot directly compare our algorithm with previous works [1], [100], [109], [130] on trajectory data anonymization because none of them can prevent from both identity and attribute linkage attacks. Instead, we compare our local suppression method with the global suppression method described in our technical report [91]. In the following experiments, we show that applying local suppression along with $(K, C)_L$-privacy would significantly lower utility loss in the context of trajectory data.

Table 5.3: Experimental dataset statistics

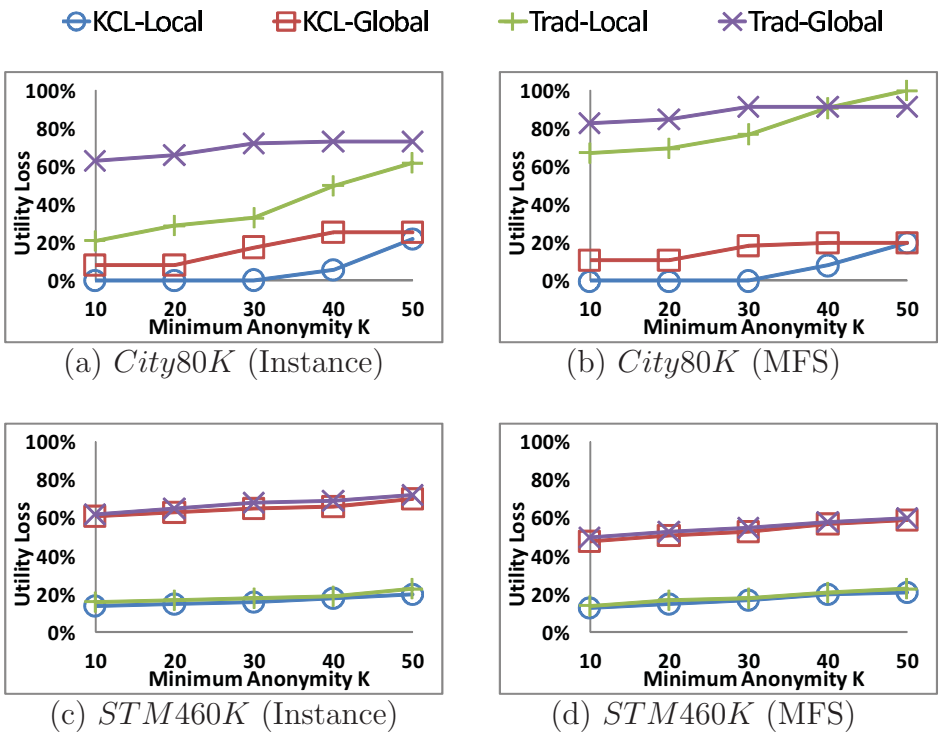
Datasets	Records $\|T\|$	Dimensions $\|d\|$	Data size (K bytes)	Sensitive set cardinality	Data type
City80K	80,000	624	2,297	1/5	Synthetic
STM460K	462,483	3,264	9,810	6/24	Real-life

Two datasets, $City80K$ and $STM460K$, are used in the experiments. $City80K$ is a *synthetic* dataset simulating the routes of 80,000 pedestrians roaming in a metropolitan area of 26 blocks in 24 hours. The sensitive attribute of $City80K$ contains a total of five possible values, one of which is considered as sensitive. $STM460K$ is a *real-life* dataset provided by *Société de transport de Montréal* (STM), the public transit agency in Montréal. It contains the transit data of 462,483 passengers among 68 subway stations within 48 hours, where the time granularity is set to hour level. The passengers' fare types are currently considered as the sensitive attribute. It contains 24 distinct values and 6 of them are considered as sensitive. The properties of the two experimental datasets are summarized in Table 5.3.

5.4.1 Utility Loss

To fully study the effectiveness of our anonymization algorithm, we evaluate the utility loss in terms of varying K, C, L values. Specifically, for preserving MFS, we also study the effect of varying K' values. Instead of examining the effect of L separately, we show the benefit of a reasonable L value over the traditional k-anonymity (confidence bounding) in combination with other parameters. In Figures 5.2-5.4, the following legends are used: *KCL-Local* uses local suppression for $(K, C)_L$-privacy; *KCL-Global* uses global suppression for $(K, C)_L$-privacy [91]; *Trad-Local* uses local suppression for traditional k-anonymity (confidence bounding); *Trad-Global* uses global suppression for traditional k-anonymity (confidence bounding).

Effect of K. We vary the parameter K from 10 to 50 while fixing $L = 3$, $C = 60\%$, and $K' = 800$, on both $City80K$ and $STM460K$ to study the effect of K on

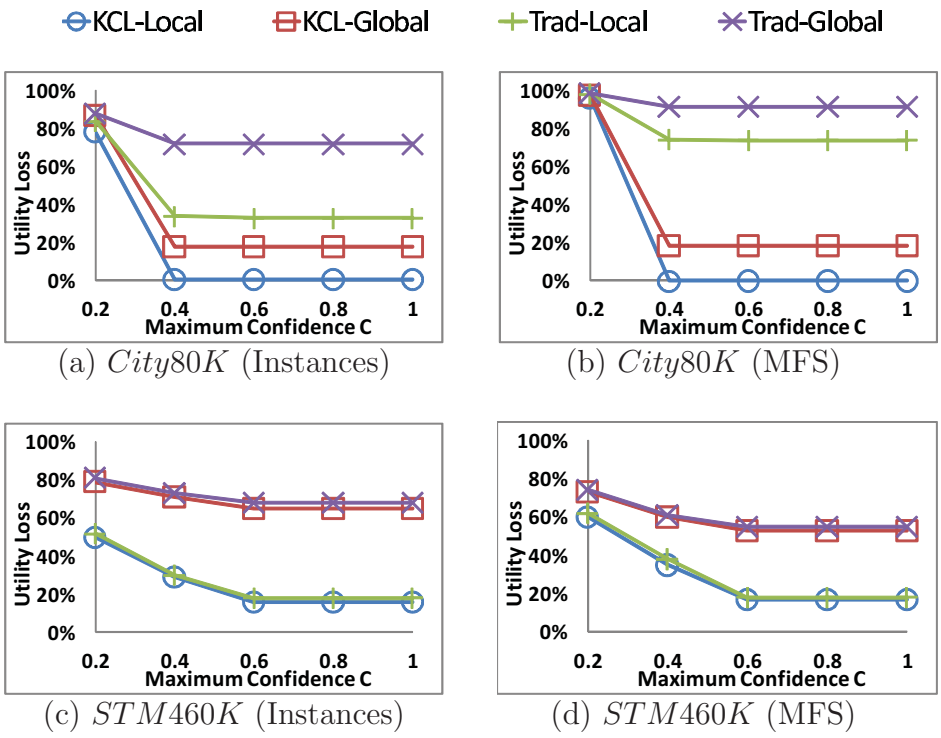


(a) $City80K$ (Instance) (b) $City80K$ (MFS)

(c) $STM460K$ (Instance) (d) $STM460K$ (MFS)

Figure 5.2: Utility loss vs. K ($L = 3, C = 60\%, K' = 800$)

$(K, C)_L$-privacy model under the two different utility metrics, the results of which are demonstrated in Figure 5.2. Recall that k-anonymity is achieved in our framework by setting $L = |d|$ and $C = 100\%$, where $|d|$ is the number of dimensions in the given dataset. Comparing the utility loss of the schemes based on $(K, C)_L$-privacy to the ones based on k-anonymity unveils the utility improvement due to the assumption of L-knowledge; comparing the schemes using local suppression to those using only global suppression unveils the utility enhancement due to the employment of local suppression. Overall, *KCL-Local* performs significantly better than *KCL-Global*. In particular, it achieves 75% improvement for instance and 68% improvement for MFS on the real dataset $STM460K$. However, local suppression itself is not sufficient to guarantee good data utility. When local suppression is applied to k-anonymity, the resulting utility loss is still relatively high on $City80K$. It is interesting to see that on $STM460K$ the utility loss under $(K, C)_L$-privacy and k-anonymity is very close. This is due to the fact that most MVS of $STM460K$ are of size-3 or



(a) $City80K$ (Instances) (b) $City80K$ (MFS)

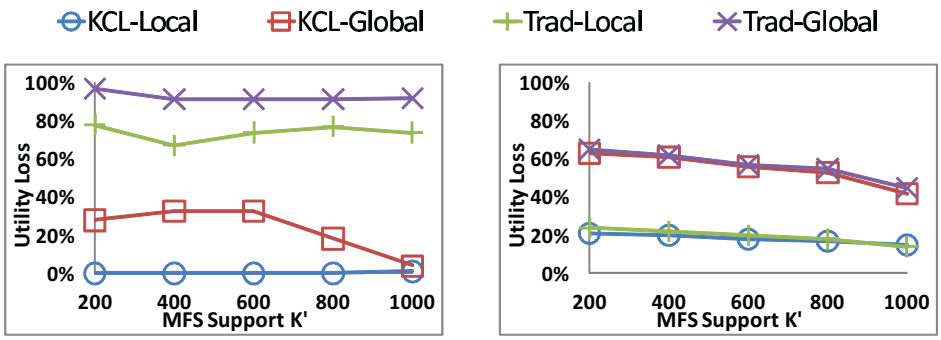
(c) $STM460K$ (Instances) (d) $STM460K$ (MFS)

Figure 5.3: Utility loss vs. C ($L = 3, K = 30, K' = 800$)

less. Nevertheless, Figure 5.2 suggests that when combined with local suppression, $(K, C)_L$-privacy can significantly lower the utility loss than can k-anonymity, in the context of trajectory data.

Effect of C. Figure 5.3 shows the impact of C on the utility loss while fixing $L = 3$, $K = 30$, and $K' = 800$, which allows us to examine the effect of attribute linkages. Since k-anonymity is unable to prevent attribute linkages, confidence bounding [114] is used to compare with $(K, C)_L$-privacy. Recall that confidence bounding is achieved under $(K, C)_L$-privacy by setting $L = |d|$. When C is small, the utility loss is high for all anonymization schemes because approximately 20% of the records of $City80K$ and 25% of the records of $STM460K$ contain a sensitive value. However, as C increases, the utility loss becomes less sensitive to C. The result also suggests that applying local suppression under $(K, C)_L$-privacy results in substantially lower utility loss.

Effect of K'. For preserving MFS, we study the relationship between K' and



(a) $City80K$ (b) $STM460K$

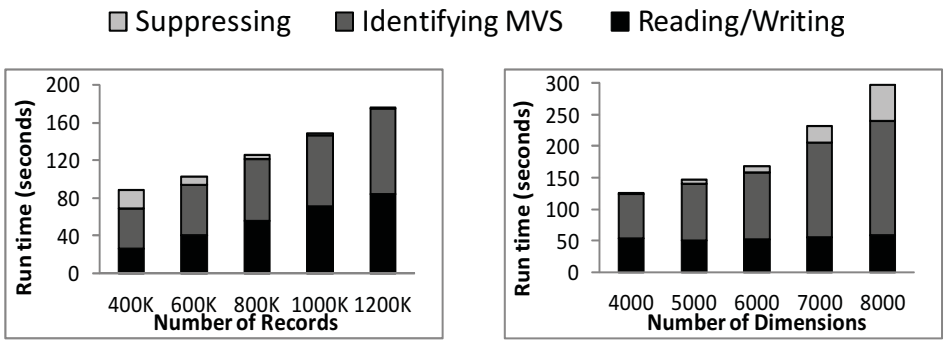
Figure 5.4: Utility loss vs. K' ($L = 3, K = 30, C = 60\%$)

the utility loss by fixing $L = 3$, $K = 30$, and $C = 60\%$ in Figure 5.4. Generally, as K' increases, the utility loss decreases. When K' gets larger, the size of MFS becomes smaller, which, in turn, makes the MFS set and MVS set have less overlap. Hence, suppressions have less influence on MFS. We also observe that local suppression is less sensitive to varying K' values due to the fact that local suppression allows decreasing the support of an MFS rather than always totally eliminating an MFS.

5.4.2 Scalability

Since the computational complexity of our algorithm is dominated by $|d|$, the number of dimensions, and $|T|$, the number of records, we study the scalability of our anonymization framework in terms of $|d|$ and $|T|$ on relatively large trajectory datasets generated with similar settings as $City80K$. Since using local suppression results in better data utility, we only evaluate the scalability of applying local suppression for preserving MFS (using only global suppression requires less computing resources), where the following parameters are used: $L = 3$, $K = 30$, $C = 60\%$, and $K' = 800$.

Effect of $|T|$. Figure 5.5 (a) presents the run time of processing datasets with 4000 dimensions and size ranging from 400,000 to 1,200,000. We can observe that the time spent on reading raw datasets and writing the anonymized datasets is



(a) *Runtime vs. # of records* (b) *Runtime vs. # of dimensions*

Figure 5.5: Scalability

proportional to the dataset sizes. The time of identifying MVS sets also increases linearly, which confirms our complexity analysis. With the increase of the data size, the time spent on suppressions, however, drops substantially. When the number of records increases, there is a much greater chance for a sequence q to satisfy $|T(q)| \geq K$; therefore, the size of MVS decreases significantly, so it takes much less time to perform all suppressions.

Effect of $|d|$. In Figure 5.5 (b), we increase the dimensions on datasets of 1 million records. The time spent reading raw data and writing anonymized data is insensitive to the number of dimensions of the given dataset. However, as the number of dimensions increases, it takes more time to generate the MVS set because the size of each candidate set increases. The size of the resulting MVS set also increases due to the increased sparseness. Thus, the time spent on suppressing all identified MVS also increases.

Overall, our anonymization framework is able to efficiently process large trajectory datasets. The total run time of anonymizing 1 million records with 8000 dimensions is still less than 300 seconds.

5.5 Summary

In this chapter, we summarize the special challenges of trajectory data anonymization and show that traditional k-anonymity and its extensions are not effective in the context of trajectory data. Based on the practical assumption of L-knowledge, we achieve a $(K, C)_L$-privacy model on trajectory data without paying extra utility and computation costs due to over-sanitization. This is the first work that introduces local suppression to trajectory data anonymization to enhance the resulting data utility. Consequently, we propose an anonymization framework that is able to remove all privacy threats from a trajectory database by both local and global suppressions. This framework is independent of the underlying data utility metrics and, therefore, is suitable for different trajectory data mining workloads. Our experimental results on both synthetic and real-life datasets demonstrate that combining $(K, C)_L$-privacy and local suppression is able to significantly improve the anonymized data quality.

Though we adopt a stronger privacy notion than other existing works, in the context of trajectory data, by taking into consideration the possibility of inferring record owners' sensitive information via trajectory data, the specificity of trajectory data enables adversaries to perform other kinds of privacy attacks, especially when they are equipped with different types of background knowledge.

Chapter 6

Sequential Data Sanitization

6.1 Introduction

Sequential data, which can be considered a simplified form of trajectory data, has been used in a variety of applications, spanning from genome and web usage analysis to location-based recommendation systems. Publishing such datasets is important since they can help us analyze and understand interesting sequential patterns. For example, mobility traces have become widely collected in recent years and have opened the possibility to improve our understanding of large-scale transportation networks. Similar to trajectory data, raw sequential data may enable an adversary to learn sensitive information about a victim. The privacy concern of publishing sequential data is best exemplified by the case of the *Société de transport de Montréal* (STM, http://www.stm.info), the public transit agency in Montreal area.

Over the last few years, smart card automated fare collection (SCAFC) systems have been increasingly deployed in transportation systems as a secure method of user validation and fare collection. These systems generate and collect passengers' transit data every day, which, after being anonymized, needs to be shared for various reasons, such as administrative regulations, profit sharing and data analysis. Transit data usually contains individual-specific sensitive information, and

publishing raw data would directly violate passengers' privacy. In 2008, the STM deployed SCAFC systems in its transportation network. Transit information, such as smart card number and station ID, is collected when a passenger swipes his smart card at a SCAFC terminal, and is then stored in a central database management system, where the transit information of a passenger is organized as a sequence of stations in time order, a kind of *sequential data* (see a formal definition in Section 6.2.1). The deployment of SCAFC systems allows the seamless integration with other transit networks of neighboring cities, for example, the *Agence métropolitaine de transport* (AMT), which consequently requires data sharing among several collaborating parties. In addition, periodically, the IT department of the STM shares transit data with other departments, e.g., the marketing department, for basic data analysis, and publishes its transit data to external research institutions for more complex data analysis tasks, such as marketing analysis [112], customer behavior analysis [14], and demand forecasting [112].

According to its preliminary research [14], [86], [18], the STM can substantially benefit from transit data analysis at strategic, tactical, and operational levels. Yet, it has also realized that the nature of transit data is raising major privacy concerns on the part of card users in information sharing [86]. This fact has been an obstacle to conducting further data analysis much less performing regular commercial operations. In this chapter, we aim to provide practical solutions to such a real-life transit data sharing scenario. We point out that our solutions also benefit many other sectors, for example cell phone communication and credit card payment, which have been facing a similar dilemma in sequential data publishing and individual privacy protection.

Previous efforts have been made in addressing the problem of transit data publication at the STM. In Chapter 5, we proposed the local suppression technique based on the $(K, C)_L$-privacy model. However, its privacy property is still dubious

when facing a strong adversary (e.g., one knows more than L-knowledge). It is therefore urgent to respond to the failure of existing sanitization techniques by developing new schemes with proven privacy guarantees. For this reason, we employ *differential privacy* [37], one of the strongest privacy models. Differential privacy provides provable privacy guarantees independent of an adversary's background knowledge and computational power.

Traditional differentially private *non-interactive* approaches [12], [39], [125] are *data-independent* in the sense that all possible entries in the output domain need to be *explicitly* considered no matter what the underlying database is. For high-dimensional data, such as sequential data, this is computationally infeasible. Consider a transit database $\mathcal{D}$ with all stations drawn from a universe of size m. Suppose the maximum length of a record (the number of stations in a record) in $\mathcal{D}$ is l. These approaches need to generate $\sum_{i=1}^{l} m^i = \frac{m^{l+1}-m}{m-1}$ output entries. For a STM transit database with $m = 1,000$ and $l = 20$, it requires to generate 10^{60} entries. Hence, these approaches are not computationally applicable with today's systems to real-life transit databases.

To tackle the challenge, we develop *data-dependent* solutions by extending the ideas proposed in two very recent papers [90], [25]. The general idea of a data-dependent solution is to adaptively narrow down the output domain by using *noisy* answers obtained from the underlying database. However, the methods in [90], [25] cannot be directly applied to sequential data for two reasons. First, the inherent sequentiality of sequential data is not considered in [90], [25]. Second, the methods only work for *sets*, yet a record in a sequential database may contain a *bag* of locations. Therefore, non-trivial efforts are needed to develop a differentially private data publishing approach for sequential data.

Protecting individual privacy is one aspect of sanitizing data. Another equally important aspect is preserving utility in sanitized data for data analysis. In this chapter, we consider two important data mining tasks conducted at the STM,

namely *count queries* (see a formal definition in Section 6.2.4) and *frequent sequential pattern mining* [6]. Count queries, as a general data analysis task, are the building block of many advanced data mining tasks. In the STM case, with accurate answers to count queries, data recipients can answer questions, such as "how many passengers have visited both stations *Guy-Concordia* and *McGill* [1]". Frequent sequential pattern mining, as a concrete data mining task, helps, for example, the STM better understand passengers' transit patterns and consequently allows the STM to adjust its network geometry and schedules in order to better utilize its existing resources.

Contributions. In this chapter, we propose two alternative solutions that pioneer the use of differential privacy for publishing sequential data. The first solution makes use of a *hybrid-granularity* prefix tree while the second solution makes use of a *variable-length n-gram model*. We summarize the major contributions of this chapter as follows.

Publishing sequential data via prefix tree. This is the first work that introduces a practical solution for publishing large volume of real-life sequential data via differential privacy in the *non-interactive* setting.

- We study the real-life transit data sharing scenario at the STM and propose an efficient sanitization algorithm to generate a differentially private sequential data release by making use of a *hybrid-granularity* prefix tree. We design a statistical process for efficiently constructing such a noisy prefix tree under Laplace mechanism, which is vital to the scalability of our solution. We emphasize that our approach can be seamlessly extended to trajectory data (see Section 6.3.1).

- We make use of two sets of inherent constraints of a prefix tree to conduct constrained inferences, which helps generate a more accurate release.

[1] *Guy-Concordia* and *McGill* are two metro stations on the green line of the STM metro network.

- We conduct an extensive experimental study over different real-life STM datasets. We examine utility of sanitized data for two different data mining tasks performed by the STM, namely count queries (a generic data analysis task) and frequent sequential pattern mining (a concrete data mining task). Experimental results demonstrate that our approach maintains high utility and is scalable to large volume of real-life sequential data.

Publishing sequential data via n-grams. To further improve data utility of sanitized data, we propose the use of a *variable-length n-gram model*, which outperforms the state-of-the-art techniques.

- For the first time, we introduce the n-gram model as an effective means of achieving differential privacy in the context of sequential data. To better suit differential privacy, we propose the use of a novel *variable-length* n-gram model, which balances the trade-off between information extracted from the underlying database and the magnitude of Laplace noise added. The variable-length n-gram model intrinsically fits differential privacy in the sense that it retains the essential information of a sequential database by identifying a set of high-quality n-grams whose counts are large enough to resist Laplace noise.

- We develop a series of techniques to guarantee good utility under the variable-length n-gram model, including an adaptive privacy budget allocation scheme, a formal choice of a threshold value, and the enforcement of consistency constraints. These techniques make use of the inherent Markov assumption in an n-gram model. In addition, we develop an efficient method to generate a synthetic dataset from released n-grams, enabling a wider spectrum of data analysis tasks.

- We conduct an extensive experimental study on the variable-length n-gram model over real-life sequential datasets, which provides important insights for

Table 6.1: Sample sequential database

Rec. #	**Sequence**
1	$L_1 \rightarrow L_2 \rightarrow L_3$
2	$L_1 \rightarrow L_2$
3	$L_3 \rightarrow L_2 \rightarrow L_1$
4	$L_1 \rightarrow L_2 \rightarrow L_4$
5	$L_1 \rightarrow L_2 \rightarrow L_3$
6	$L_3 \rightarrow L_2$
7	$L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_1$
8	$L_3 \rightarrow L_1$

future work. In particular, we demonstrate that our solution substantially outperforms the state-of-the-art solutions [88], [21] in terms of count query and frequent sequential pattern mining.

The results of this chapter have been published in [22], [20].

6.2 Preliminaries

6.2.1 Sequential Data

Let $\mathcal{L} = \{L_1, L_2, \cdots, L_{|\mathcal{L}|}\}$ be the universe of locations, where $|\mathcal{L}|$ is the size of the universe. Without loss of generality, we consider locations as discrete spatial areas in a map. For example, in a transportation system, $\mathcal{L}$ represents all stations in the transit network. This assumption also applies to many other types of sequential data, e.g., purchase records, where a location is a store's address, or web browsing histories, where a location is a URL. We model a *sequence* as a time ordered list of locations drawn from the universe.

Definition 6.1 (Sequence)**.** A sequence D of length $|D|$ is a time ordered list of locations $D = l_1 \rightarrow l_2 \rightarrow \cdots \rightarrow l_{|D|}$, where $\forall 1 \leq i \leq |D|$, $l_i \in \mathcal{L}$. ■

A location may occur multiple times in D, and may occur consecutively in D. Therefore, given $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$, $D = L_1 \rightarrow L_2 \rightarrow L_2$ is a valid sequence. A

sequential database is composed of a multiset of sequences; each sequence represents the movement history of a record owner. A formal definition is given below.

Definition 6.2 (Sequential Database)**.** A sequential database $\mathcal{D}$ of size $|\mathcal{D}|$ is a multiset of sequences $\mathcal{D} = \{D_1, D_2, \cdots, D_{|\mathcal{D}|}\}$. ■

Table 6.1 presents a sample sequential database with $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$.

6.2.2 Prefix Tree

A sequential (or trajectory) database can be represented in a more compact way in terms of a *prefix tree*. A prefix tree groups sequences with the same prefix into the same branch. We first define a prefix of a sequence below.

Definition 6.3 (Sequence Prefix)**.** A sequence $S = s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_{|S|}$ is a prefix of a sequence $T = t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_{|T|}$, denoted by $S \preceq T$, if and only if $|S| \leq |T|$ and $\forall 1 \leq i \leq |S|$, $s_i = t_i$. ■

For example, $L_1 \rightarrow L_2$ is a prefix of $L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_3$, but $L_1 \rightarrow L_4$ is *not*. Note that a sequence prefix is a sequence per se. Next, we formally define a prefix tree below.

Definition 6.4 (Prefix Tree)**.** A prefix tree $\mathcal{PT}$ of a sequential database $\mathcal{D}$ is a triplet $\mathcal{PT} = (V, E, \mathit{Root}(\mathcal{PT}))$, where V is the set of nodes labeled with locations, each corresponding to a unique sequence prefix in $\mathcal{D}$; E is the set of edges, representing transitions between nodes; $\mathit{Root}(\mathcal{PT}) \in V$ is the virtual root of $\mathcal{PT}$. The unique sequence prefix represented by a node $v \in V$, denoted by $prefix(v, \mathcal{PT})$, is an ordered list of locations starting from $\mathit{Root}(\mathcal{PT})$ to v. ■

Each node $v \in V$ of $\mathcal{PT}$ keeps a doublet in the form of $\langle tr(v), c(v) \rangle$, where $tr(v)$ is the set of sequences in $\mathcal{D}$ having the prefix $prefix(v, \mathcal{PT})$, that is, $\{D \in \mathcal{D} : prefix(v, \mathcal{PT}) \preceq D\}$, and $c(v)$ is a noisy version of $|tr(v)|$ (e.g., $|tr(v)|$ plus Laplace

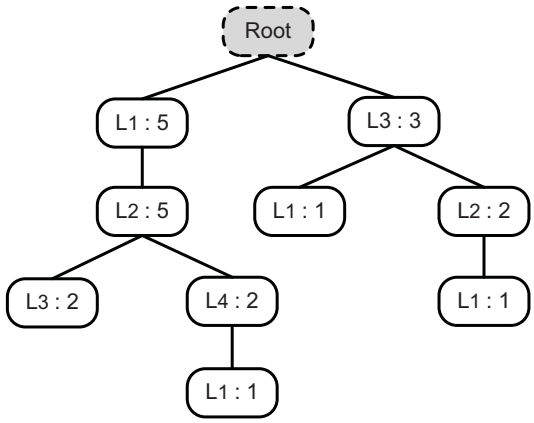


Figure 6.1: The prefix tree of the sample data in Table 6.1

noise). $tr(Root(\mathcal{PT}))$ contains all sequences in $\mathcal{D}$. We call the set of all nodes of $\mathcal{PT}$ at a given depth i a *level* of $\mathcal{PT}$, denoted by $level(i, \mathcal{PT})$. $Root(\mathcal{PT})$ is at depth zero. Figure 6.1 illustrates the prefix tree of the sample database in Table 6.1, where each node v is labeled with its location and $|tr(v)|$.

6.2.3 N-Gram Model

An n-gram model is a type of probabilistic prediction model based on an $(n-1)$-order *Markov* model. It can compactly model large-scale sequential data and provide scalable trade-off between storage and accuracy. N-gram models have been proven to be very robust in modeling sequential data and have been widely used in probability, communication theory, computational linguistics (e.g., statistical natural language processing), computational biology (e.g., biological sequence analysis), and data compression.

N-gram models estimate the probability of the next location for a given sequence by making use of the *Markov independence assumption* (of order $n-1$) that the occurrence of each location in a sequence depends only on the previous $n-1$ locations (instead of *all* previous locations), where n is typically a small value (e.g., 3-5). Let the probability that a sequence $L_1 \rightarrow L_2 \rightarrow \ldots \rightarrow L_i$, where $L_j \in \mathcal{L}$ $(\forall 1 \leq j \leq i)$

Table 6.2: Another sample sequential dataset

Rec. #	**Sequence**
1	$L_2 \to L_3 \to L_1$
2	$L_2 \to L_3$
3	$L_3 \to L_2$
4	$L_2 \to L_3 \to L_1$
5	$L_3 \to L_2 \to L_1$
6	$L_2 \to L_3 \to L_1 \to L_2 \to L_3$
7	$L_3 \to L_2$
8	$L_3 \to L_1 \to L_2 \to L_3$

Table 6.3: 1-grams

Gram	**#**	**Pr**
L_1	5	0.21
L_2	9	0.38
L_3	10	0.41

Table 6.4: 2-grams

Gram	**#**	**Pr**	**Gram**	**#**	**Pr**	**Gram**	**#**	**Pr**
$L_1 \to L_1$	0	0	$L_2 \to L_1$	1	0.11	$L_3 \to L_1$	4	0.4
$L_1 \to L_2$	2	0.4	$L_2 \to L_2$	0	0	$L_3 \to L_2$	3	0.3
$L_1 \to L_3$	0	0	$L_2 \to L_3$	6	0.67	$L_3 \to L_3$	0	0
$L_1 \to \&$	3	0.6	$L_2 \to \&$	2	0.22	$L_3 \to \&$	3	0.3

and $i \geq n$, is followed by $L_{i+1} \in \mathcal{L}$ be denoted by $P(L_{i+1}|L_1 \to L_2 \to \ldots \to L_i)$. Then, under the n-gram model, we have:

$$P(L_{i+1}|L_1 \to L_2 \to \ldots \to L_i) :\approx P(L_{i+1}|L_{i-n+2} \to L_{i-n+3} \to \ldots \to L_i).$$

In the sequel, the probability $P(L_{i+1}|L_j \to L_{j+1} \to \cdots \to L_i)$ is shortly denoted by $P(L_{i+1}|L_i^j)$.

N-gram models provide a trade-off between storage and accuracy: a larger n value retains more information of the dataset, but it requires more storage and time to process. For example, Tables 6.3 and 6.4 show the set of all unigrams and 2-grams, respectively, along with their counts and probabilities for the sample dataset in Table 6.2, where & is a special symbol representing the termination of a sequence. Consider the calculation of the (approximated) number of occurrences of $L_3 \to L_1 \to L_2 \to L_3$, whose true number is 2. Using 2-grams, one possible approximation is $\#(L_3 \to L_1) \cdot P(L_2|L_1) \cdot P(L_3|L_2) = 4 \cdot 0.4 \cdot 0.67 = 1.07$. In contrast, using 3-grams,

a better approximation can be $\#(L_3 \rightarrow L_1 \rightarrow L_2) \cdot P(L_3|L_1 \rightarrow L_2) = 2 \cdot 1.0 = 2.0$. However, this better scheme requires to process all 3-grams at the cost of storage and time.

6.2.4 Utility Requirements

In the STM case, sanitized data is mainly used to perform two data mining tasks, namely *count query* and *frequent sequential pattern mining* [6]. Count queries, as a general data analysis task, are the building block of many data mining tasks. We formally define count queries over a sequential database below.

Definition 6.5 (Count Query)**.** For a given set of locations $\mathbb{L}$ drawn from the universe $\mathcal{L}$, a count query Q over a database $\mathcal{D}$ is defined to be $Q(\mathcal{D}) = |\{S \in \mathcal{D} : \mathbb{L} \subseteq ls(S)\}|$, where $ls(S)$ returns the set of locations in S. ■

Note that sequentiality among locations is not considered in count queries, because the major users of count queries are, for example, the personnel of the marketing department of the STM, who are merely interested in users' presence in certain stations for marketing analysis, known as *passenger counting*, but *not* the sequentiality of visiting [2]. Instead, the preservation of sequentiality in sanitized data is examined by frequent sequential pattern mining. The utility of a count query Q over the sanitized database $\widetilde{\mathcal{D}}$ is similarly measured by its *relative error* [125], [123], [25], which is computed as:

$$error(Q(\widetilde{\mathcal{D}})) = \frac{|Q(\widetilde{\mathcal{D}}) - Q(\mathcal{D})|}{max\{Q(\mathcal{D}), s\}},$$

where s is a *sanity bound* used to mitigate the influences of queries with extremely small *selectivities* [125], [123], [25].

[2] A variant of count query that considers sequentiality is employed in Section 6.4.3 to demonstrate the utility of sanitized data for a broad spectrum of mobility trace analysis tasks.

For frequent sequential pattern mining, we measure the utility of sanitized data in terms of *true positive* (TP), *false positive* (FP) and *false drop* (FD) [40]. Given a positive number k, we denote the set of top k most frequent sequential patterns with size greater than 1 [3] on the original database $\mathcal{D}$ by $\mathcal{F}_k(\mathcal{D})$ and the set of frequent sequential patterns on the sanitized database $\widetilde{\mathcal{D}}$ by $\mathcal{F}_k(\widetilde{\mathcal{D}})$. True positive is the number of frequent sequential patterns in $\mathcal{F}_k(\mathcal{D})$ that are correctly identified in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\mathcal{D}) \cap \mathcal{F}_k(\widetilde{\mathcal{D}})|$. False positive is the number of infrequent sequential patterns in $\mathcal{D}$ that are mistakenly included in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\widetilde{\mathcal{D}}) - \mathcal{F}_k(\mathcal{D}) \cap \mathcal{F}_k(\widetilde{\mathcal{D}})|$. False drop is the number of frequent sequential patterns in $\mathcal{F}_k(\mathcal{D})$ that are wrongly omitted in $\mathcal{F}_k(\widetilde{\mathcal{D}})$, that is, $|\mathcal{F}_k(\mathcal{D}) \cup \mathcal{F}_k(\widetilde{\mathcal{D}}) - \mathcal{F}_k(\widetilde{\mathcal{D}})|$. Since in our setting $|\mathcal{F}_k(\mathcal{D})| = |\mathcal{F}_k(\widetilde{\mathcal{D}})| = k$, false positives always equal false drops.

6.3 Publishing Sequential Data via Prefix Tree

6.3.1 Sanitization Algorithm

We first provide an overview of our two-step sanitization algorithm in Algorithm 6.1. Given a raw sequential dataset $\mathcal{D}$, a privacy budget ϵ, a user specified height of the prefix tree h and a location taxonomy tree $\mathcal{T}$, it returns a sanitized dataset $\widetilde{\mathcal{D}}$ satisfying ϵ-differential privacy. *BuildNoisyPrefixTree* constructs a *noisy hybrid-granularity* prefix tree $\mathcal{PT}$ of $\mathcal{D}$ using a set of count queries based on the given taxonomy tree $\mathcal{T}$, which defines multiple levels of granularities over the location universe. It can be either public knowledge or generated from the location universe on-the-fly by specifying a fan-out value. In the STM case, we use a two-level taxonomy tree where each station can be generalized to the metro/bus line on which it locates. For the simplicity of illustration, we give our algorithm based on a two-level taxonomy tree. The extension to a multiple-level taxonomy tree is straightforward. *GeneratePrivateRelease* employs utility boosting techniques on $\mathcal{PT}$ based on two sets of

[3] Nearly all single locations form a size-1 frequent sequential pattern.

Algorithm 6.1: Prefix Tree Sequential Data Sanitization

Input: Raw sequential dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Input: Location taxonomy tree $\mathcal{T}$
Output: Sanitized dataset $\widetilde{\mathcal{D}}$

1: Noisy prefix tree $\mathcal{PT} \leftarrow BuildNoisyPrefixTree(\mathcal{D}, \epsilon, h, \mathcal{T})$;
2: Sanitized dataset $\widetilde{\mathcal{D}} \leftarrow GeneratePrivateRelease(\mathcal{PT})$;
3: **return** $\widetilde{\mathcal{D}}$;

consistency constraints, and then generates a differentially private release.

Noisy Prefix Tree Construction

Our strategy for *BuildNoisyPrefixTree* is to recursively group sequences in $\mathcal{D}$ into *disjoint* sub-datasets based on their prefixes. Algorithm 6.2 presents the details of *BuildNoisyPrefixTree*. We first create a prefix tree $\mathcal{PT}$ with a virtual root *Root* (Lines 2-3). To build $\mathcal{PT}$, we employ a *uniform* privacy budget allocation scheme, that is, divide the total privacy budget ϵ into equal portions $\bar{\epsilon} = \frac{\epsilon}{h}$, each is used for constructing a level of $\mathcal{PT}$ (Line 4). In Lines 6-26, we iteratively construct each level of $\mathcal{PT}$ in a noisy way.

To satisfy differential privacy, we need to guarantee that every sequence that can be derived from the location universe (either in or not in $\mathcal{D}$) has a non-zero probability to appear in the noisy prefix tree. Therefore, at each level, for each node, we need to consider *every* possible location as its potential child. Our goal is to identify the children that are associated with non-zero number of sequences (referred to as *non-empty node*) so that we can continue to expand them. Here decisions have to be made based on noisy counts.

In order to achieve good utility, it is critical to prune out nodes associated with zero number of sequences (*empty node*) reliably as early as possible. For this reason, instead of using a simple prefix tree, we divide a level of $\mathcal{PT}$ into two sub-levels with different location granularities. The first sub-level consists of nodes associated with

Algorithm 6.2: *BuildNoisyPrefixTree*

Input: Raw sequential dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Input: Location taxonomy tree $\mathcal{T}$
Output: Noisy prefix tree $\mathcal{PT}$

```
1: i = 0;
2: Create a prefix tree PT with a virtual root Root;
3: Add all sequences in D to tr(Root);
4: ε̄ = ε/h;
5: Calculate ε̄_1 and ε̄_2 s.t. ε̄_1 + ε̄_2 = ε̄;
6: while i < h do
7:    for each non-generalized node v ∈ level(i) do
8:       U_g ← the set of generalized nodes from T;
9:       for each node u ∈ U_g do
10:         Add sequences S with prefix(u) ⪯ S to tr(u);
11:         c(u) = NoisyCount(|tr(u)|, ε̄_1);
12:         if c(u) ≥ θ_g then
13:            Add u to PT;
14:            U_ng ← u's non-generalized children in T;
15:            for each node w ∈ U_ng do
16:               Add sequences S with prefix(w) ⪯ S to tr(w);
17:               c(w) = NoisyCount(|tr(w)|, ε̄_2);
18:               if c(w) ≥ θ_ng then
19:                  Add w to PT;
20:               end if
21:            end for
22:         end if
23:      end for
24:   end for
25:   i++;
26: end while
27: return PT;
```

generalized location information (*generalized node*), and then, depending on noisy counts of these nodes, we decide whether to further expand them to create the second sub-level in which nodes are associated with non-generalized locations (e.g., ask the noisy counts of passengers in a metro line and then decide whether to ask counts of each station on this line). $\bar{\epsilon}$ is then allocated to the two sub-levels as a function of the fan-out f of the location taxonomy tree $\mathcal{T}$: the first sub-level receives $\bar{\epsilon}_1 = \frac{2\bar{\epsilon}}{f}$ and the second receives $\bar{\epsilon}_2 = \frac{(f-2)\bar{\epsilon}}{f}$. One important observation is that all nodes on the same sub-level are associated with *disjoint* sequence subsets, and therefore the privacy budget allocated to a sub-level can be used *in full* for each node in it. We provide formal analysis on utility improvement of a hybrid-granularity prefix tree after presenting Theorem 6.1.

For a dataset with a very large location universe $\mathcal{L}$, processing all locations explicitly may be slow. We provide an efficient implementation by *separately* handling potential non-empty and empty nodes. For a non-empty node u, we add Laplace noise to $|tr(u)|$ and use the noisy answer $c(u)$ to decide if it is non-empty. If $c(u)$ is greater than or equal to the pre-defined threshold θ, we deem that u is non-empty and insert u to $\mathcal{PT}$. In the STM case, the threshold of a non-generalized node $\theta_{ng} = \frac{2\sqrt{2}}{\bar{\epsilon}_2}$ (two times of the standard deviation of noise) while the threshold of a generalized node $\theta_g = \frac{4\sqrt{2}}{\bar{\epsilon}_1}$. Intuitively, this setting more reliably eliminates empty nodes while having very limited effect on non-empty nodes. Since non-empty nodes are typically of a small number, this process can be done efficiently.

For empty nodes, we need to conduct a series of *independent* boolean tests, each calculates $NoisyCount(0, \bar{\epsilon}')$ to check if it passes θ, where $\bar{\epsilon}'$ is the privacy budget assigned to a node (either $\bar{\epsilon}_1$ or $\bar{\epsilon}_2$). The number of empty nodes that pass θ, k, follows the *binomial distribution* $B(m, p_\theta)$, where m is the total number of empty nodes we need to check and p_θ is the probability for a single experiment to succeed. We design a statistical process for Laplace mechanism to directly extract k empty nodes without explicitly processing every empty node. This is inspired by [29], in

which a statistical process is designed for *geometric mechanism* [51].

Theorem 6.1. *Independently conducting m pass/not pass experiments based on Laplace mechanism with privacy budget $\bar{\epsilon}$ and threshold θ is equivalent to the following steps: (1) generate a value k from the binomial distribution $B(m, p_\theta)$, where $p_\theta = \frac{exp(-\bar{\epsilon}\theta)}{2}$; (2) select k uniformly random empty nodes without replacement with noisy counts sampled from the distribution*

$$P(x) = \begin{cases} 0 & \forall x < \theta \\ 1 - exp(\bar{\epsilon}\theta - \bar{\epsilon}x) & \forall x \geq \theta \end{cases} \blacksquare$$

Proof. The probability of a single experiment passing the threshold θ is

$$Pr[PASS] = \int_\theta^\infty \frac{\bar{\epsilon}'}{2} exp(-\bar{\epsilon}'x)dx = \frac{exp(-\bar{\epsilon}'\theta)}{2}.$$

Since the experiments are independent, the number of successful experiments, k, follows the binomial distribution $B(m, \frac{exp(-\bar{\epsilon}'\theta)}{2})$. Once k is determined, we can uniformly at random select k empty nodes. The probability density function of the noisy counts x for the k empty nodes, conditional on $x \geq \theta$, is:

$$p(x|x \geq \theta) = \begin{cases} 0 & \forall x < \theta \\ \frac{\frac{\bar{\epsilon}'}{2} exp(-\bar{\epsilon}'x)}{p_\theta} = \bar{\epsilon}' exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x) & \forall x \geq \theta \end{cases}$$

The corresponding cumulative distribution function is:

$$P(x) = \begin{cases} 0 & \forall x < \theta \\ \int_\theta^x \bar{\epsilon}' exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x)dx = 1 - exp(\bar{\epsilon}'\theta - \bar{\epsilon}'x) & \forall x \geq \theta \end{cases}$$

□

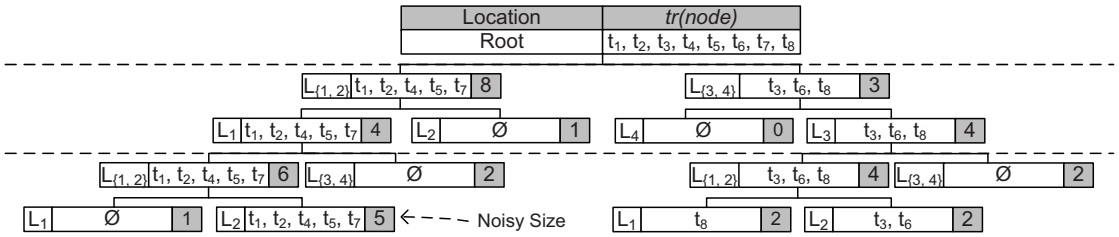


Figure 6.2: The noisy hybrid-granularity prefix tree of the sample data

Now we give a theoretical analysis on the utility improvement due to a hybrid-granularity prefix tree in terms of reduction of number of empty non-generalized nodes that are mistakenly generated. This number directly reflects the level of noise in sanitized data.

Theorem 6.2. *For an empty node v at level i, a noisy hybrid-granularity prefix tree of height h reduces the number of empty non-generalized nodes mistakenly generated due to identifying v as non-empty by a factor of $O(2^{h-i}exp(4\sqrt{2}(h-i)))$.* ■

Proof. From Theorem 6.1, we learn that $p_{\theta_{ng}} = \frac{exp(-2\sqrt{2})}{2}$ and $p_{\theta_g} = \frac{exp(-4\sqrt{2})}{2}$. Consider an empty node v at level i. In a simple noisy prefix tree, if v is mistakenly considered as non-empty, the expected value of number of descendants of v, which are all empty nodes, is $\mathbb{E}_1 = (|\mathcal{L}|p_{\theta_{ng}})^{h-i}$. In a hybrid-granularity prefix tree, the expected value of number of descendants of v is

$$\mathbb{E}_2 = (fp_{\theta_{ng}})(\frac{|\mathcal{L}|}{f}p_{\theta_g} \cdot fp_{\theta_{ng}})^{h-i} = (fp_{\theta_{ng}})(|\mathcal{L}|p_{\theta_g}p_{\theta_{ng}})^{h-i}.$$

where f is the fan-out of the location taxonomy tree. This implies a reduction of

$$\frac{\mathbb{E}_1}{\mathbb{E}_2} = \frac{1}{fp_{\theta_{ng}}p_{\theta_g}^{h-i}} = O(2^{h-i}exp(4\sqrt{2}(h-i))).$$

□

Example 6.3.1. Consider the sequential database $\mathcal{D}$ in Table 6.1, the height $h = 2$, and the calculated threshold $\theta = 3$. Suppose that L_1 and L_2 can be generalized to

$L_{\{1,2\}}$ and L_3 and L_4 can be generalized to $L_{\{3,4\}}$. The construction of a possible noisy hybrid-granularity prefix tree $\mathcal{PT}$ is illustrated in Figure 6.2. A path of $\mathcal{PT}$ may be of a length shorter than h if it has been considered "empty" before h is reached. ■

Private Release Generation

We can generate the sanitized database by traversing $\mathcal{PT}$ once in *postorder* (ignore generalized nodes), calculating the number n of sequences terminated at each non-generalized node v and appending n copies of $prefix(v, \mathcal{PT})$ to the output. However, due to the noise added to ensure differential privacy, we may not be able to obtain a meaningful and consistent release. For example, in Figure 6.2, consider the root-to-leaf path $Root(\mathcal{PT}) \rightarrow L_1 \rightarrow L_2$. We have $c(L_2) > c(L_1)$, which is counterintuitive because it is not possible, in general, to have more sequences with the prefix $prefix(u, \mathcal{PT})$ than with the prefix $prefix(v, \mathcal{PT})$, where u is a child of v in $\mathcal{PT}$. If we leave such inconsistencies unsolved, the resulting release may not be meaningful and therefore provides poor utility.

Definition 6.6 (Consistency Constraint)**.** In a prefix tree, there exist two sets of consistency constraints:

1. For any root-to-leaf path p, $\forall v_i \in p, |tr(v_i)| \leq |tr(v_{i+1})|$, where v_i is a child of v_{i+1};
2. For each node v, $|tr(v)| \geq \sum_{u \in children(v)} |tr(u)|$. ■

Our goal is to enforce such consistency constraints on the noisy prefix tree (with all generalized nodes removed [4]) in order to produce a consistent and more accurate private release. We adapt the constrained inference technique proposed in [61] to adjust the noisy counts of nodes in the noisy prefix tree so that the constraints defined in Definition 6.6 are respected. Note that the technique proposed

[4]The utility improvements of constrained inferences on a prefix tree with and without generalized nodes are almost identical.

in [61] cannot be directly applied to our case because: 1) the noisy prefix tree has an irregular structure (rather than a complete tree with a fixed degree); 2) the noisy prefix tree has different constraints $|tr(v)| \geq \sum_{u \in children(v)} |tr(u)|$ (rather than $|tr(v)| = \sum_{u \in children(v)} |tr(u)|$). Consequently, we propose a two-phase procedure to obtain a *consistent* estimate with respect to Definition 6.6 for each node (except the virtual root) in the noisy prefix tree $\mathcal{PT}$.

We first generate an intermediate estimate for the noisy count of each node v (except the virtual root) in $\mathcal{PT}$. Consider a root-to-leaf path p of $\mathcal{PT}$. Let us organize the noisy counts of nodes $v_i \in p$ into a sequence $\mathcal{S} = \langle c(v_1), c(v_2), \cdots, c(v_{|p|}) \rangle$, where v_i is a child of v_{i+1}. Let $mean[i, j]$ denote the mean of a subsequence of $\mathcal{S}$, $\langle c(v_i), c(v_{i+1}), ..., c(v_j) \rangle$, that is, $mean[i, j] = \frac{\sum_{m=i}^{j} c(v_m)}{j-i+1}$. We compute the intermediate estimates $\widetilde{\mathcal{S}}$ by Theorem 6.3 [61].

Theorem 6.3. *Let* $L_m = min_{j \in [m,|p|]} max_{i \in [1,j]} mean[i, j]$ *and* $U_m = max_{i \in [1,m]} min_{j \in [i,|p|]} mean[i, j]$. *The minimum* L_2 *solution* $\widetilde{\mathcal{S}} = \langle L_1, L_2, ..., L_{|p|} \rangle = \langle U_1, U_2, ..., U_{|p|} \rangle$. ■

The result of Theorem 6.3 satisfies the first type of constraints in Definition 6.6. However, a node v in $\mathcal{PT}$ appears in $|leaves(v)|$ root-to-leaf paths, where $leaves(v)$ denotes the leaves of the subtree of $\mathcal{PT}$ rooted at v, and therefore, has $|leaves(v)|$ intermediate estimates, each being an independent observation of the true count $|tr(v)|$. We compute the *consolidated* intermediate estimate of v as the mean of the estimates, normally the best estimate for $|tr(v)|$. We denote the consolidated intermediate estimate of v by $\widetilde{c}(v)$.

After obtaining $\widetilde{c}(v)$ for each node v, we compute its consistent estimate $\bar{c}(v)$ in a top-down fashion as follows:

$$\bar{c}(v) = \begin{cases} \widetilde{c}(v) & if v \in level(1, \mathcal{PT}) \\ \widetilde{c}(v) + min(0, \frac{\bar{c}(w) - \sum_{u \in children(w)} \widetilde{c}(u)}{|children(w)|}) & otherwise \end{cases}$$

where node w is the parent of node v. It follows the intuition that the observation $\sum_{u \in children(w)} \widetilde{c}(u) > \bar{c}(w)$ is strong evidence that excessive noise is added to the children. Since the variance of noise in $\bar{c}(w)$ is approximately $|children(w)|$ times smaller than $\sum_{u \in children(w)} \widetilde{c}(u)$, it is reasonable to decrease the children's counts according to $\bar{c}(w)$. However, we never increase the children's counts based on $\bar{c}(w)$ because a large $\bar{c}(w)$ simply indicates that many sequences actually terminate at w. It is easy to see that the consistency constraints in Definition 6.6 are respected among consistent estimates, and therefore the proof is omitted here.

Analysis

Privacy Analysis. Kifer and Machanavajjhala [69] point out that differential privacy must be applied with caution. The privacy protection provided by differential privacy relates to the data generating mechanism and deterministic aggregate-level background knowledge. In the STM case, transit data is independent of each other and no deterministic statistics of the raw database will ever be released. Hence differential privacy is appropriate for our problem. We now show that Algorithm 6.1 satisfies ϵ-differential privacy.

Theorem 6.4. *Given the total privacy budget ϵ, Algorithm 6.1 ensures ϵ-differential privacy.* ■

Proof. Algorithm 6.1 consists of two steps, namely *BuildNoisyPrefixTree* and *GeneratePrivateRelease*. In the procedure *BuildNoisyPrefixTree*, our approach appeals to the well-understood query model to construct the noisy prefix tree $\mathcal{PT}$. Consider a level of $\mathcal{PT}$, which is composed of two sub-levels. Since all nodes on the same sub-level contain *disjoint* sets of sequences. According to the *parallel composition* (Theorem 2.2 [87]), the entire privacy budget needed for a sub-level is bounded by the worst case, that is, $\frac{2\bar{\epsilon}}{f}$ for the first sub-level and $\frac{(f-2)\bar{\epsilon}}{f}$ for the second sub-level. The use of privacy budget on different sub-levels follows *sequential composition* (Theorem 2.1 [87]). Since there are at most h levels, the total privacy budget needed to

build the noisy prefix tree $\leq h \times (\frac{2\bar{\epsilon}}{f} + \frac{(f-2)\bar{\epsilon}}{f}) = \epsilon$.

For *GeneratePrivateRelease*, we make use of the inherent constraints of a prefix tree to boost utility. The procedure only accesses a differentially private noisy prefix tree, not the underlying database. As proven by Hay et al. [61], a post-processing of differentially private results remains differentially private. Therefore, Algorithm 6.1 as a whole maintains ϵ-differential privacy. □

Complexity Analysis. Algorithm 6.1 is of runtime complexity $O(|\mathcal{D}| \cdot |\mathcal{L}|)$, where $|\mathcal{D}|$ is the size of the input database $\mathcal{D}$ and $|\mathcal{L}|$ is the size of the location universe. For *BuildNoisyPrefixTree*, the major computational cost is node generation and sequence distribution. For each level of the noisy prefix tree, the number of nodes to generate approximates $k(1 + \frac{1}{f})|\mathcal{D}|$, where $k \ll |\mathcal{L}|$ is a number depending on $|\mathcal{L}|$. For each level, we need to distribute *at most* $2|\mathcal{D}|$ sequences to the newly generated nodes. Hence, the complexity of constructing a single level is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Therefore, the total runtime complexity of *BuildNoisyPrefixTree* for constructing a noisy prefix tree of height h is $O(h|\mathcal{D}| \cdot |\mathcal{L}|)$. In *GeneratePrivateRelease*, the complexity of calculating the intermediate estimates for a single root-to-leaf path is $O(h)$. Since there can be at most $|\mathcal{D}|$ distinct root-to-leaf paths, the complexity of computing all intermediate estimates is $O(h|\mathcal{D}|)$. To compute consistent estimates, we need to visit every node exactly twice, which is of complexity $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Similarly, the computational cost of generating the private release by traversing the noisy prefix tree once in postorder is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$. Since h is a very small constant compared to $|\mathcal{D}|$ and $|\mathcal{L}|$, the total complexity of Algorithm 6.1 is $O(|\mathcal{D}| \cdot |\mathcal{L}|)$.

Extensions

Our solution can be seamlessly applied to trajectory data. A trajectory is composed of a sequence of *location-timestamp* pairs in the form of $loc_1t_1 \rightarrow loc_2t_2 \rightarrow \cdots \rightarrow loc_nt_n$, where $t_1 \leq t_2 \leq \cdots \leq t_n$. The time factor is often discretized into intervals at

Table 6.5: Experimental dataset statistics.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{L}\|$	$max\|S\|$	$avg\|S\|$
Metro	847,668	68	90	4.21
Bus	778,724	944	121	5.67

different levels of granularity, e.g., hour, which is typically determined by the data publisher. All timestamps of a trajectory database form a timestamp universe.

In this case, we can label each node in the prefix tree by both a location and a timestamp. Therefore, two trajectories with the same sequence of locations but different timestamps are considered different. For example, $L_1T_1 \rightarrow L_2T_2$ is different from $L_1T_2 \rightarrow L_2T_3$, and the corresponding prefix tree will have two non-overlapping root-to-leaf paths. Consequently, when constructing the noisy prefix tree, in order to expand a node loc_it_i, we have to consider the combinations of all locations and the timestamps in the time universe that are greater than t_i (because the timestamps in a trajectory are non-decreasing), resulting in a larger candidate set. Due to the efficient implementation we propose, the computational cost will remain moderate.

6.3.2 Experimental Evaluation

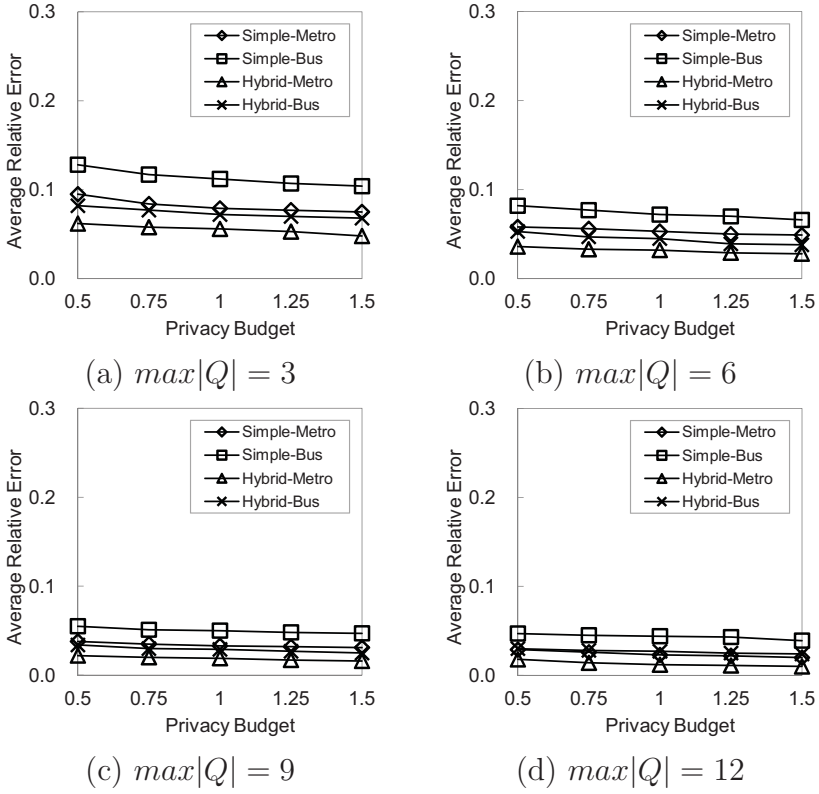
In this section, we examine the utility of sanitized data in terms of count queries and frequent sequential pattern mining, and evaluate the scalability of our approach for processing large-scale real-life data. In particular, we compare the utility improvements of the method using a hybrid-granularity prefix tree (referred to as *Hybrid*) over the method using a simple prefix tree (referred to as *Simple*). Our implementation was done in C++, and all experiments were performed on an Intel Core 2 Duo 2.26GHz PC with 2GB RAM. Extensive experiments were conducted on two real-life STM transit datasets, *Metro* and *Bus*, which record the transit history of passengers in the STM metro and bus networks, respectively. The characteristics of the datasets are summarized in Table 6.5, where $max|S|$ is the maximum sequence length and $avg|S|$ the average length.

Utility

Count Query. In our first set of experiments, we examine relative errors of count queries with respect to two different parameters, namely the privacy budget ϵ and the noisy prefix tree height h. We follow the evaluation scheme from previous works [125], [25]. For each privacy budget, we generate 40,000 random count queries with varying numbers of locations. We call the number of locations in a query the *length* of the query. We divide the query set into 4 subsets such that the query length of the i-th subset is uniformly distributed in $[1, \frac{ih}{4}]$ and each location is randomly drawn from the location universe $\mathcal{L}$. The sanity bound s is set to 0.1% of the dataset size, the same as [125], [25].

Figure 6.3 examines average relative errors under varying privacy budgets from 0.5 to 1.5 with $h = 12$. The X-axes represent the different query subsets by their maximum length $max|Q|$. As expected, the average relative errors decrease when ϵ increases because less noise is added and the construction process is more precise. In general, our approach maintains high utility for count queries. Even in the worst case ($\epsilon = 0.5$ and $max|Q| = 3$), the average relative error of *Hybrid* is still less than 8.2% on both datasets. Such level of relative errors is acceptable for data analysis at the STM. We can also observe that a hybrid-granularity structure can substantially reduce average relative errors (with 33%-48% improvement), especially on *Bus*, which is more sparse.

Figure 6.4 studies how average relative errors vary under different h values with query length fixed to 6. We can observe that with the increase of h, the relative errors do not decrease monotonically. Initially, the relative errors decrease when h increases because the increment of h allows to retain more information from the underlying database. However, after a certain threshold, the relative errors become larger with the increase of h, because when h gets larger, the noise added to each level grows quickly. It is interesting to see that the hybrid-granularity structure indeed better eliminates noise, and makes relative errors less sensitive to varying h



(a) $max|Q| = 3$ (b) $max|Q| = 6$

(c) $max|Q| = 9$ (d) $max|Q| = 12$

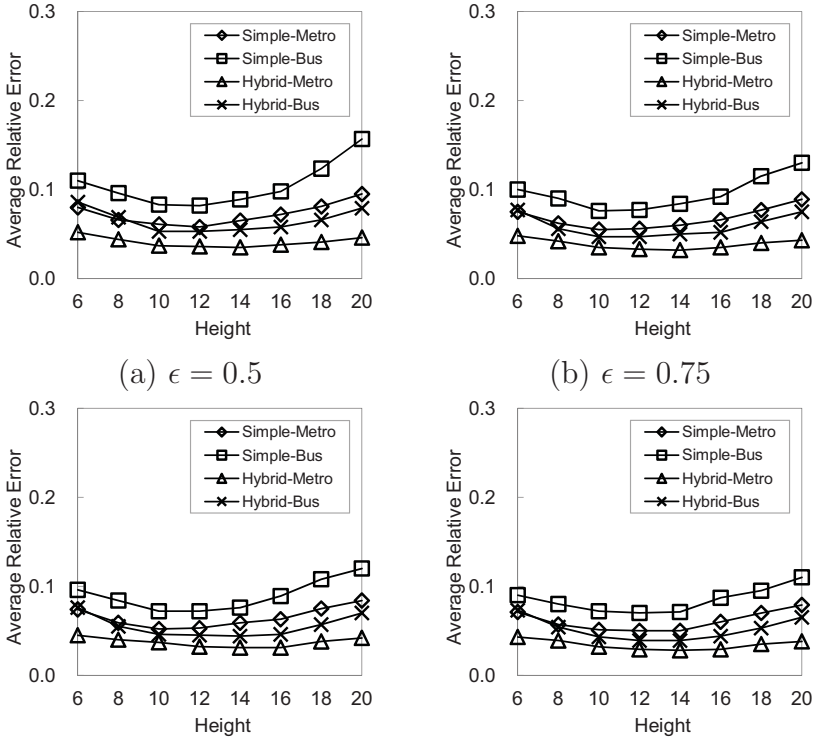
Figure 6.3: Average relative error vs. privacy budget.

values. This allows a wider range of h values (e.g., 10-16) to be used in order to obtain desirable relative errors.

Frequent sequential pattern mining. In the second set of experiments, we demonstrate the utility of sanitized data by frequent sequential pattern mining. Specifically, we employ PrefixSpan to mine frequent sequential patterns [5].

Table 6.6 shows how the utility changes with different top k values while fixing $\epsilon = 1.0$ and $h = 12$. When $k = 100$, the sanitized data generated by *Hybrid* is able to give the exact top 100 most frequent patterns that are of size greater than 1. With the increase of k values, the *accuracy* (the ratio of true positive to k) decreases. However, even when $k = 300$, the accuracy of *Hybrid* is still as high as

[5]An implementation of the PrefixSpan algorithm proposed in [98], available at http://code.google.com/p/prefixspan/.



(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$

(c) $\epsilon = 1.0$ (d) $\epsilon = 1.25$

Figure 6.4: Average relative error vs. prefix tree height.

257/300 = 85.7% on *Metro* and 233/300 = 77.7% on *Bus*. Again we can observe that *Hybrid* always outperforms *Simple* on both datasets under all k values. When $k = 300$, the improvement due to the hybrid-granularity structure is 6.6% on *Metro* and 9.9% on *Bus*.

Table 6.7 presents the utility for frequent sequential pattern mining under different ϵ values while fixing $h = 12$ and $k = 300$. Generally, larger privacy budgets lead to more true positives and fewer false positives (false drops). This conforms to the theoretical analysis that a larger privacy budget results in less noise and therefore a more accurate result. Since the most frequent sequential patterns are of small length, they have large supports from the underlying database. As a result, the utility is relatively insensitive to varying privacy budgets, and the accuracy is high even when the privacy budget is small.

Table 6.6: Utility for frequent sequential pattern mining vs. k

k	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
100	99/97	1/3	100/100	0/0
150	143/139	7/11	149/144	1/6
200	178/168	22/32	185/177	15/23
250	209/195	41/55	220/209	30/41
300	241/212	59/88	257/233	43/67

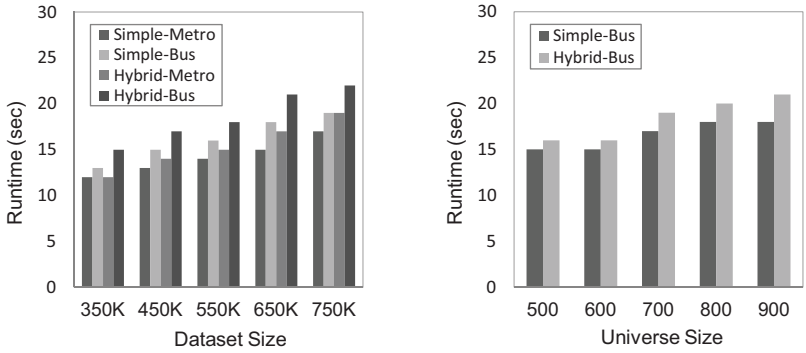
Table 6.7: Utility for frequent sequential pattern mining vs. ϵ

ϵ	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
0.5	227/194	73/106	244/215	56/85
0.75	239/206	61/94	253/224	47/76
1.0	241/212	59/88	257/233	43/67
1.25	243/216	57/84	259/238	41/62
1.5	248/224	52/76	261/242	39/58

Table 6.8: Utility for frequent sequential pattern mining vs. h

h	TP (M/B) *Simple*	FP(FD) (M/B) *Simple*	TP (M/B) *Hybrid*	FP(FD) (M/B) *Hybrid*
6	234/212	66/88	241/221	59/79
8	240/217	60/83	254/232	46/68
10	241/215	58/85	255/236	45/64
12	241/212	59/88	257/233	43/67
14	241/212	59/88	258/233	42/67
16	240/210	60/90	258/231	42/69
18	240/209	60/91	255/230	45/70
20	238/206	62/94	254/228	46/72

Table 6.8 studies how the utility varies under different h values with $\epsilon = 1.0$ and $k = 300$. It is interesting to see that in general frequent sequential pattern mining is also insensitive to varying h values. This can be similarly explained by the large supports of frequent sequential patterns, which make them more resistant to noise. In addition, we can observe that good performance for frequent sequential pattern mining can be obtained by a h value between 10 and 16, the same range as



(a) Runtime vs. $|\mathcal{D}|$ (b) Runtime vs. $|\mathcal{L}|$

Figure 6.5: Runtime vs. different parameters.

that for count queries.

Scalability

In the last set of experiments, we examine the scalability of our approach. Recall that the runtime complexity of our approach is dominated by the database size $|\mathcal{D}|$ and the location universe size $|\mathcal{L}|$. Therefore, we study the runtime under different database sizes and different location universe sizes. Figure 6.5.a presents the runtime under different database sizes with $\epsilon = 1.0$ and $h = 20$ for both *Simple* and *Hybrid*. The test sets are generated by randomly extracting records from *Metro* and *Bus*. We can observe that the runtime is linear to the database size. Moreover, it can be seen that the computational cost of a hybrid-granularity prefix tree structure is negligible. This further confirms the benefit of employing a hybrid-granularity prefix tree.

Figure 6.5.b shows how runtime varies under different location universe sizes. Since *Metro* is of a small universe size, we only study the effect of universe sizes on *Bus*. For each universe size, we remove all locations falling out of the universe from *Bus*. This results in a smaller database size. Consequently, we fix the database size for all test sets to 600,000. Again, it can be observed that the runtime scales linearly with the location universe size and that the computational cost of *Hybrid* is

comparable to that of *Simple* under different location universe sizes. As a summary, our approach is scalable to large sequential datasets. It takes less than 22 seconds to sanitize both datasets in all previous experiments.

6.3.3 Summary

All existing techniques for privacy-preserving sequential data publishing are derived using partition-based privacy models, which have been shown failing to provide sufficient privacy protection. In this section, motivated by the STM case, we study the problem of publishing sequential data in the framework of differential privacy. For the first time, we present a practical data-dependent solution for sanitizing large-scale real-life sequential data, which can also be seamlessly applied to trajectory data. In addition, we develop a constrained inference technique in order to better the resulting utility. Our solution has been tested on real-life STM transit data for two fundamental data analysis tasks performed at the STM and exhibits satisfactory effectiveness and efficiency. We believe that our solution could benefit many other sectors that are facing the dilemma between the demands of sequential data publishing and privacy protection.

6.4 Publishing Sequential Data via N-Grams

6.4.1 Sanitization Algorithm

The main idea of our scheme is simple: we add properly calibrated Laplace noise to the counts of high-quality grams and release them. Our goal is two-fold: (1) to release grams whose real counts are large enough to increase utility [6]; (2) to maximize the sizes of released grams (i.e., the n value) to preserve as much sequentiality

[6]The added noise is calibrated to the global sensitivity and is independent of the count values. As a result, larger counts provide better utility.

information from the underlying dataset as possible. There is a fundamental trade-off between the utility of noisy n-grams and their sizes: shorter grams enjoy smaller relative error due to Laplace noise but carry less sequentiality information; longer grams contain more sequentiality information but have smaller counts (and thus larger relative error). In this section, we address this trade-off by releasing *variable-length* n-grams of counts larger than a threshold [7] and of sizes less than a maximal size $n_{\max}$ [8]. For most practical datasets, setting $n_{\max}$ to a small value (e.g., 3-5) has been sufficient to capture most of the sequentiality information. Since short grams are typically of large counts, this property, which is also experimentally justified in Section 6.4.3, explains why the n-gram model is so powerful and why it provides an excellent basis for differentially private sequential data publishing.

To identify the set of high-quality (i.e., having low relative errors) n-grams with possibly varying n values ($1 \leq n \leq n_{\max}$) from an input sequential dataset, we propose a well-designed tree structure, called *exploration tree*. It groups grams with the same prefix into the same branch so that all possible n-grams with size $1 \leq n \leq n_{\max}$ can be explored efficiently. The exploration starts with unigrams and then proceeds to longer grams until $n_{\max}$ is reached. Intuitively, if the noisy count of a gram g is small (i.e., close to the standard deviation of the added noise), its true count also tends to be small and thus the relative error is large. Since all grams having the prefix g (i.e., all nodes in the subtree rooted at g) have smaller true counts than g's true count, they can be omitted from further computation. This observation makes our approach significantly faster than naively processing every single gram regardless of its size. It also explains why we do not adopt the approach that generates all possible n-grams and then prunes the tree.

[7] This threshold is set to limit the magnitude of noise in released data.

[8] Beyond $n_{\max}$, the utility gain of longer grams is usually smaller than the utility loss due to noise.

Terminology

Before discussing the details, we first give some notations used in our algorithm. The exploration tree is denoted by $\mathcal{T}$. Each node $v \in \mathcal{T}$ is labeled by a location $L \in \mathcal{L} \cup \{\&\}$, where & is a special symbol representing the termination of a sequence. The function $lb(v)$ returns v's location label. Each node v is associated with an n-gram defined by the sequence of locations from the root of $\mathcal{T}$ to v, denoted by $g(v)$. We slightly abuse the term *count* to mean the number of occurrences of $g(v)$ in the input dataset, which is denoted by $|g(v)|$. Note that an n-gram may occur multiple times in a sequence. For example, the count of $L_2 \rightarrow L_3$ in the sample database in Table 6.2 is 6, *not* 5. Each node v also keeps a noisy version of $|g(v)|$, denoted by $c(v)$. In addition, each node v conveys a conditional probability, denoted by $p(v)$, which predicts the probability of the transition from v's parent to v. $p(v)$ can be obtained by normalizing the noisy counts of v's siblings and v. For example, in Figure 6.6, $p(v_4) = P(L_1|L_2 \rightarrow L3) = 4/(4+0+1+2) = 4/7$. The set of all nodes in level i of $\mathcal{T}$ is denoted by $level(i, \mathcal{T})$ and these nodes represent all i-grams in the dataset. The level number of node v in $\mathcal{T}$ is denoted by $levelNum(v, \mathcal{T})$. The root of $\mathcal{T}$ is in level zero.

Detailed Descriptions

Private sequential database construction. Algorithm 6.3 provides an overview of our approach. It takes as inputs a sequential database $\mathcal{D}$, the total privacy budget ϵ, the maximal sequence length $\ell_{\max}$ and the maximal n-gram size $n_{\max}$, and returns a sanitized sequential database $\widetilde{\mathcal{D}}$ satisfying ϵ-differential privacy. $\ell_{\max}$ is a parameter specified by the data holder to limit the influence of a single sequence in computation. The algorithm considers only the first $\ell_{\max}$ locations in each input sequence. A larger $\ell_{\max}$ allows more information to be retained from $\mathcal{D}$, but requires more noise to be injected in later computation; a smaller $\ell_{\max}$ does the opposite. We discuss and report the effect of different $\ell_{\max}$ values in Section 6.4.3, and provide

Algorithm 6.3: N-gram Sequential Data Sanitization

Input: Raw sequential database $\mathcal{D}$
Input: Privacy budget ϵ
Input: Maximal sequence length $\ell_{\max}$
Input: Maximal n-gram size $n_{\max}$
Output: Private sequential database $\widetilde{\mathcal{D}}$

Truncate each $S \in \mathcal{D}$ by keeping the first $\ell_{\max}$ items;
$i = 0$;
Create an exploration tree $\mathcal{T}$ with a virtual root;
while $i < n_{\max}$ **do**
  for each non-leaf node $v_{ij} \in level(i, \mathcal{T})$ and $lb(v_{ij}) \neq \&$ **do**
    Calculate ϵ_{ij};
    $U_c \leftarrow$ all possible children of v_{ij} with labels $\mathcal{L} \cup \{\&\}$
    $\mathbf{Q} = \{|g(u_1)|, |g(u_2)|, \cdots, |g(u_{|\mathcal{I}|+1})|\}$, where $u_k \in U_c$;
    $\widetilde{\mathbf{Q}} = Laplace(\mathbf{Q}, \ell_{\max}/\epsilon_{ij})$;
    for each node $u_k \in U_c$ **do**
      Add u_k to $\mathcal{T}$;
      if $c(u_k) < \theta$ **then**
        Mark u_k as leaf;
      end if
    end for
  end for
  i++;
end while
Enforce consistency constraints on $\mathcal{T}$;
Construct $\widetilde{\mathcal{D}}$ from $\mathcal{T}$;
return $\widetilde{\mathcal{D}}$;

insights for a data holder to select a good $\ell_{\max}$ value in practice. $n_{\max}$ bounds the height of the exploration tree $\mathcal{T}$ and thus the maximal size of released grams. The choice of $n_{\max}$ affects the privacy parameter assigned to each level of $\mathcal{T}$, and, therefore, is also related to the magnitude of noise. In practice, $n_{\max}$ could be set to 5, which is the maximal n value that is widely used for n-gram applications in the literature. Similarly, we present more details on the selection of a reasonable $n_{\max}$ value in Section 6.4.3. We emphasize that this does not mean that all released grams have a size of $n_{\max}$ but rather their sizes can vary between 1 and $n_{\max}$.

In Algorithm 6.3, we first preprocess $\mathcal{D}$ by keeping only the first $\ell_{\max}$ locations

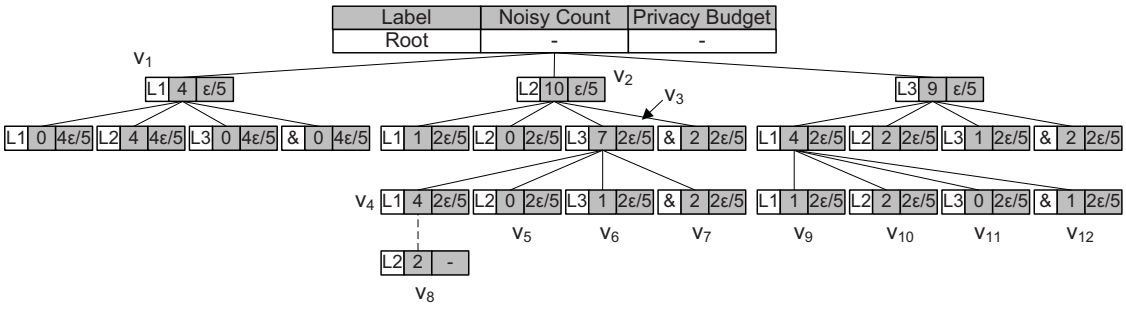


Figure 6.6: The exploration tree of the sample data in Table 6.2

of each sequence in order to bound the influence of a single sequence by $\ell_{\max}$ (Line 1). The construction of $\mathcal{T}$ starts by creating an empty tree with a virtual root (Line 3). In Lines 4-18, the algorithm iteratively constructs each level of $\mathcal{T}$. For level i of $\mathcal{T}$, we decide whether to expand a node $v_{ij} \in level(i, \mathcal{T})$ by comparing its noisy count $c(v_{ij})$ with a threshold θ. If $c(v_{ij}) \geq \theta$, we expand v_{ij} by explicitly considering every possible location in $\mathcal{L} \cup \{\&\}$ as a child of v_{ij} in order to satisfy differential privacy. By definition, nodes labeled by & cannot be expanded because it means the termination of a sequence. The entire exploration process ends when either the depth of the tree reaches $n_{\max}$ (i.e., all subsequently released grams would be longer than $n_{\max}$) or no node can be further expanded (since their noisy counts do not pass θ). Example 6.4.1 illustrates the construction of an exploration tree.

Example 6.4.1. Given $n_{\max} = 5$, $\ell_{\max} = 5$ and $\theta = 3$, the construction of a possible exploration tree over the sample dataset in Table 6.2 is illustrated in Figure 6.6 (ignore the privacy budget information and the node (v_8) connected by a dash line for now). ■

In the following, we detail the key components of Algorithm 6.3: how to compute the privacy budget ϵ_{ij} for each node in $\mathcal{T}$, how to compute the threshold θ for each node, how to approximate the counts of nodes in $\mathcal{T}$, and how to reconstruct a synthetic version of the input database $\mathcal{D}$ from $\mathcal{T}$.

Adaptive privacy budget allocation. Given the maximal n-gram size $n_{\max}$, a

simple privacy budget allocation scheme is to expect the height of $\mathcal{T}$ to be $n_{\max}$ and uniformly assign $\frac{\epsilon}{n_{\max}}$ to each level of $\mathcal{T}$ in order to calculate the noisy counts of all nodes in this level. However, in reality, many (or even all) root-to-leaf paths have a length much shorter than $n_{\max}$ for the reason of their counts not being able to pass θ. Hence assigning privacy budget solely based on $n_{\max}$ is clearly not optimal. For example, in Example 6.4.1, since the height of the exploration tree is 3 and $n_{\max} = 5$, at least $\frac{2\epsilon}{5}$ privacy budget would be wasted in all paths.

To address this drawback, we propose an adaptive privacy budget allocation scheme that allows private operations to make better use of the total privacy budget ϵ. Intuitively, a desirable privacy budget allocation scheme should take into consideration the length of a root-to-leaf path: for a shorter path, each node in the path should receive more privacy budget; for a longer path, each node should use less privacy budget. Therefore, we adaptively estimate the length of a path based on known *noisy* counts and then distribute the remaining privacy budget as per the estimated length.

At the beginning of the construction of $\mathcal{T}$, in the absence of information from the underlying dataset, we can only assume that each root-to-leaf path is of the same length $n_{\max}$ so that our algorithm would not exceptionally halt due to running out of privacy budget. Therefore, $\frac{\epsilon}{n_{\max}}$ is used to calculate the noisy counts of nodes in level 1. Once we obtain some information from the underlying dataset (e.g., nodes' noisy counts), we can make more accurate predictions on the length of a path.

For a node v in level $i \geq 2$ with noisy count $c(v)$, we predict the height h_v of the subtree rooted at v, denoted by $\mathcal{T}_v$, as follows. Let $p_{\max}$ be the estimation of the probability of transiting from v to the *mode* of its children (i.e., v's child with the largest noisy count). Assume that the probability of the mode at *each* level of $\mathcal{T}_v$ is also $p_{\max}$ [9]. Under this assumption, we can calculate the largest noisy count of the nodes in level h_v of $\mathcal{T}_v$ by $c(v) \cdot (p_{\max})^{h_v}$. Recall the fact that $\mathcal{T}_v$ will not be

[9] A more precise estimation could be obtained by applying the Markov assumption to each level of $\mathcal{T}_v$ at the cost of efficiency.

further expanded if none of the nodes in level h_v can pass the threshold θ. We get $c(v) \cdot (p_{\max})^{h_v} = \theta$, that is, $h_v = \log_{p_{\max}} \frac{\theta}{c(v)}$. Since the height of $\mathcal{T}_v$ is bounded by $n_{\max} - i$, we have

$$h_v = \min(\log_{p_{\max}} \frac{\theta}{c(v)}, n_{\max} - i).$$

Next we discuss how to calculate p_{max} for v. Let the i-gram associated with v be $L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_i$ ($L_j \in \mathcal{L}$ for $1 \leq j \leq i$). Then we need to estimate the probability distribution of v's children from noisy counts known by far. We resort to the Markov assumption for this task. Recall that the order $i-1$ Markov assumption states $P(L_{i+1}|L_i^1) :\approx P(L_{i+1}|L_i^2)$. Since $P(L_{i+1}|L_i^2)$ may *not* be known in $\mathcal{T}$ (because we expand a node only when it passes the threshold θ), we consider a chain of Markov assumptions (of different orders)

$$P(L_{i+1}|L_i^1) :\approx P(L_{i+1}|L_i^2) :\approx P(L_{i+1}|L_i^3) :\approx \cdots :\approx P(L_{i+1})$$

to find the best estimation of $P(L_{i+1}|L_i^1)$, which is the conditional probability with the longest condition (i.e., the leftmost conditional probability in the chain) that is known in $\mathcal{T}$. Since $\mathcal{T}$ contains all unigrams, there is always an estimation of $P(L_{i+1}|L_i^1)$, denoted by $\widetilde{P}(L_{i+1}|L_i^1)$. $p_{\max}$ is then defined to be

$$\max_{L_{i+1} \in \mathcal{L} \cup \{\&\}} \widetilde{P}(L_{i+1}|L_i^1).$$

If $P(L_{i+1}|L_i^1)$ and $\widetilde{P}(L_{i+1}|L_i^1)$ are represented by nodes v and v' in $\mathcal{T}$, respectively, then v' is the *Markov parent* of v in $\mathcal{T}$, and any pair of corresponding nodes in the above chain are *Markov neighbors*.

Once $p_{\max}$ is calculated, we can calculate the privacy parameter ϵ_v that is used for calculating the noisy counts of v's children as follows:

$$\epsilon_v = \frac{\bar{\epsilon}}{\min(\log_{p_{\max}} \frac{\theta}{c(v)}, n_{\max} - i)},$$

where $\bar{\epsilon}$ is the remaining privacy budget (i.e., the total privacy budget ϵ minus the sum of privacy parameters consumed by v and v's ancestors). It can be observed that this scheme ensures that the privacy budget used in a root-to-leaf path is always $\leq \epsilon$.

Example 6.4.2. Continue from Example 6.4.1. For all nodes in level 1, $\frac{\epsilon}{5}$ is used to calculate their noisy counts. For the expansion of the node labeled by v_1 in Figure 6.6, we have $p_{\max} = \frac{10}{4+10+9} = 0.43$ and $h_{v_1} = 1$. Therefore, the noisy counts of v_1's children are calculated with privacy parameter $\epsilon - \frac{\epsilon}{5} = \frac{4\epsilon}{5}$. For the expansion of the node v_2, we get $p_{\max} = \frac{10}{4+10+9} = 0.43$ and $h_{v_2} = 2$. Hence its children's noisy counts will be calculated with privacy parameter $\frac{\epsilon - \frac{\epsilon}{5}}{2} = \frac{2\epsilon}{5}$. For the expansion of the node labeled by v_3, we have $p_{\max} = \frac{4}{9}$ and $h_{v_3} = 1$. Hence, $\frac{2\epsilon}{5}$ is used to compute the noisy counts of v_3's children. ■

The sensitivities of $\mathbf{Q}$ (Lines 8-9) in different levels are different. For $\mathbf{Q}$ in level i, a single record of length $\leq \ell_{\max}$ can change $\mathbf{Q}$ by at most $\ell_{\max} - i + 1$. However, under the adaptive privacy budget allocation scheme, we have to use the largest sensitivity among all levels, that is $\ell_{\max}$, in all Laplace mechanisms; otherwise, ϵ-differential privacy may be violated.

Computing threshold θ. A node in $\mathcal{T}$ is not further expanded if its noisy count is less than the threshold θ. The main source of error in $\mathcal{T}$ comes from the nodes that are of a true count of zero but of a noisy count greater than θ (referred to as *false nodes*). For this reason, we design a threshold to limit the total number of false nodes in $\mathcal{T}$ with the goal of lowering the magnitude of noise in $\mathcal{T}$.

For each expansion, a false node v will generate $|\mathcal{L}| \cdot p_\theta$ (expected value) false nodes, where p_θ is the probability of Laplace noise passing θ. This is because a descendant of v must have a true count of zero. With the expansion of $\mathcal{T}$, the number of false nodes accumulates *exponentially* with the factor of $|\mathcal{L}| \cdot p_\theta$, resulting in excessive noise. To limit the exponential growth of false nodes, we should demand

$|\mathcal{L}| \cdot p_\theta \leq 1$, that is, $p_\theta \leq \frac{1}{|\mathcal{L}|}$. Since, under Laplace mechanism, given the threshold θ and the privacy parameter ϵ',

$$p_\theta = \int_\theta^\infty \frac{\epsilon'}{2\ell_{\max}} \exp\left(-\frac{x\epsilon'}{\ell_{\max}}\right) dx = \frac{1}{2} \exp\left(-\frac{\epsilon'\theta}{\ell_{\max}}\right),$$

we get the threshold $\theta = \frac{\ell_{\max} \cdot \ln \frac{|\mathcal{L}|}{2}}{\epsilon'}$. We show in Section 6.4.3 that this threshold is effective in eliminating false nodes while having very limited influence on nodes with large counts (i.e., the high-quality grams we want to identify).

Enforcing consistency constraints. The generated exploration tree $\mathcal{T}$ may contain some inconsistencies for the reason that: (1) the sum of children's noisy counts is very unlikely to equal their parent's noisy count, and (2) there are some leaf nodes whose noisy counts are missing (since their counts cannot pass the threshold θ). Consequently, we propose a method to resolve such inconsistencies with the goal of improving data utility. In Section 6.4.3, we experimentally show that this method helps achieve better performance.

The general idea is to approximate the missing counts by making use of the Markov assumption and then normalize children's counts based on their parent's count. More specifically, our method works as follows. If none of the children of a node v in $\mathcal{T}$ exceed the threshold θ, it is strong evidence that v should not be further expanded, and therefore all children of v (leaf nodes in $\mathcal{T}$) are assigned noisy counts 0. If all children pass θ, we first calculate the conditional probability of each child based on the sum of all children's noisy counts, and then obtain a consistent approximation by multiplying this probability with their parent's noisy count. If some children (but *not* all) of v pass θ, we approximate the noisy counts of the other children by the Markov assumption. Let v_c and $\mathcal{C}(v_c)$ denote a child of v whose noisy count cannot pass θ (called a *missing node*) and its Markov parent in $\mathcal{T}$, respectively. Let V denote the set of v's children. We partition V into V^+ and V^-, where V^+ contains all nodes passing the threshold, whereas V^- contains the

rest.

1. Define the following ratio for each $v_i \in V^-$:

$$r_{v_i} = \frac{p(\mathcal{C}(v_i))}{\sum_{v_j \in V^+} p(\mathcal{C}(v_j))}$$

   For each $v_j \in V^+$, let $A(v_j)$ denote the noisy count resulted by the Laplace mechanism in Line 9 of Algorithm 6.3.

2. If $levelNum(\mathcal{C}(v_c), \mathcal{T}) \geq 2$, $\forall v_i \in V^-$, $A(v_i) = r_{v_i} \cdot \sum_{v_j \in V^+} A(v_j)$

3. Otherwise,

   (a) If $\sum_{v_j \in V^+} A(v_j) \leq c(v)$, $\forall v_i \in V^-$, $A(v_i) = \frac{c(v) - \sum_{v_j \in V^+} A(v_j)}{|V^-|}$

   (b) Otherwise, $\forall v_i \in V^-$, $A(v_i) = 0$

4. Renormalization: $\forall v_i \in V$, $c(v_i) = c(v) \cdot \frac{A(v_i)}{\sum_{v_j \in V} A(v_j)}$

If v_c can find a high-quality Markov parent in $\mathcal{T}$ (i.e., one representing an n-gram with $n \geq 2$ [10]), we estimate its counts from its high-quality siblings based on the ratio defined in Step 1. The idea behind this definition is that the *ratio* of any node insignificantly changes between Markov neighbors, and hence, it can be well approximated from the Markov parents. Otherwise, we approximate the noisy counts by assuming a uniform distribution, that is, equally distribute the count left among the missing nodes (Step 3). In Step 4, these estimated counts are normalized by the parent's count in order to obtain consistent approximations.

Example 6.4.3. Continue from Example 6.4.1. Suppose that $A(v_9) = 2.1$, $A(v_{10}) = 4$, $A(v_{11})$ do not pass θ, and $A(v_{12}) = 1.9$. Since $r_{v_{11}} = 0/(4+0+0) = 0$, $A(v_{11}) :\approx (1.9 + 4 + 2.1) \cdot 0 = 0$. Finally, renormalizing the result, we obtain $c(v_9) = 4 \cdot$

[10] A unigram conveys an unconditional probability and therefore cannot provide a very accurate estimation.

$2.1/(2.1+4+1.9+0) \approx 1, c(v_{10}) = 4 \cdot 4/(2.1+4+1.9+0) = 2, c(v_{11}) = 0, c(v_{12}) = 4 \cdot 1.9/(2.1+4+1.9+0) \approx 1$. ■

Synthetic sequential database construction. The released n-grams are useful for many data analysis tasks. However, it is often necessary to generate a synthetic database for different types of queries (and some other tasks). Hence, we propose an efficient solution to construct a synthetic version of the input sequential database from the exploration tree $\mathcal{T}$ (Line 20). The general idea is to iteratively generate longer grams (up to size $\ell_{\max}$) based on the Markov assumption and then make use of the theorem below for synthetic sequential database construction.

Theorem 6.5. *Given the set of n-grams with size $1 \leq n \leq \ell_{\max}$, the (truncated) input database (with maximal sequence length $\ell_{\max}$) can be uniquely reconstructed.* ■

Proof. Since $\ell_{\max}$-grams can only be supported by sequences of length $\ell_{\max}$, all sequences of length $\ell_{\max}$ can be uniquely reconstructed by $\ell_{\max}$-grams. Once all sequences of length $\ell_{\max}$ have been identified, we can eliminate their influences on the given set of n-grams by updating the counts of all grams supported by them. The resulting set of n-grams ($1 \leq n \leq \ell_{\max} - 1$) can be considered as if they were generated from an input database of sequences with maximal length $\ell_{\max} - 1$. Therefore, sequences of length $\ell_{\max} - 1$ can be uniquely reconstructed as well. Following this iterative process, all sequences can be uniquely identified. This proof explains the way we generate the synthetic database based on noisy n-gram. □

Intuitively, longer grams can be generated by "joining" shorter grams. Formally, we define a *join* operation over two n-grams. Let $g_1 = L_{11} \rightarrow L_{12} \rightarrow \cdots \rightarrow L_{1n}$ and $g_2 = L_{21} \rightarrow L_{22} \rightarrow \cdots \rightarrow L_{2n}$. Then g_1 can join with g_2 if $\forall 2 \leq i \leq n$, $L_{1i} = L_{2(i-1)}$, denoted by $g_1 \bowtie g_2$, and $g_1 \bowtie g_2 = L_{11} \rightarrow L_{12} \rightarrow \cdots \rightarrow L_{1n} \rightarrow L_{2n}$. Note that the join operation is *not* symmetric: it is possible that g_1 can join with g_2, but not vice versa.

Let the height of $\mathcal{T}$ be h. We iteratively extend $\mathcal{T}$ by generating n-grams with $h < n \leq \ell_{\max}$, starting from level h of $\mathcal{T}$. We extend $\mathcal{T}$ level by level, where level $n+1$, representing all $(n+1)$-grams, can be generated by joining all possible n-grams in level n. Let g_1 and g_2 be two n-grams that can be joined. We estimate the count of the joined $(n+1)$-gram as follows:

$$\begin{aligned}
|g_1 \bowtie g_2| &= c(g_1) \times P(L_{2n}|g_1) \\
&= c(g_1) \times P(L_{2n}|L_{11} \to L_{12} \to \cdots \to L_{1n}) \\
&\approx c(g_1) \times P(L_{2n}|L_{12} \to L_{13} \to \cdots \to L_{1n}) \\
&= c(g_1) \times P(L_{2n}|L_{21} \to L_{22} \cdots \to L_{2(n-1)}) \\
&\approx c(g_1) \times \frac{c(L^{21}_{2n})}{c(L^{21}_{2(n-1)})}
\end{aligned}$$

Note that all counts in the above equation are noisy ones for the reason of privacy (see more details in Section 6.4.2). Since $c(L^{21}_{2n})$ and $c(L^{21}_{2(n-1)})$ must have been known in the extended $\mathcal{T}$, $|g_1 \bowtie g_2|$ can be computed. We keep extending $\mathcal{T}$ until: 1) $\ell_{\max}$ has been reached, or; 2) no grams in a level can be joined.

Example 6.4.4. Continue from Example 6.4.1. $L_2 \to L_3 \to L_1$ and $L_3 \to L_1 \to L_2$ can be joined to generate $L_2 \to L_3 \to L_1 \to L_2$. Its count can be calculated by $c(L_2 \to L_3 \to L_1) \times \frac{c(L_3 \to L_1 \to L_2)}{c(L_3 \to L_1)} = 4 \times \frac{2}{4} = 2$. This 4-gram can be represented as a new node in $\mathcal{T}$, as illustrated by v_8 in Figure 6.6. Similarly, $L_2 \to L_3 \to L_1$ can join with $L_3 \to L_1 \to L_1$, resulting in $L_2 \to L_3 \to L_1 \to L_1$. Since these two 4-grams cannot be joined, the extension of $\mathcal{T}$ ends at level 4. ■

After extending $\mathcal{T}$, we can generate the synthetic database in the following way. Let the height of the extended $\mathcal{T}$ be h_e. We start from level h_e. For each $v \in level(h_e, \mathcal{T})$, we publish $c(v)$ copies of $g(v)$, and update the count of each node v' supported by $g(v)$ by subtracting $c(v) \cdot c_i$ (i.e., all nodes representing a gram that can be generated from $g(v)$), where c_i is the number of occurrence of $g(v')$ in $g(v)$.

An n-gram supports *at most* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ nodes and therefore requires *at most* $\frac{n(n+1)}{2}$ updates. With a hash map structure, each update can be done in constant time.

Example 6.4.5. Continue from Example 6.4.4. For node v_8 in level 4 of $\mathcal{T}$, we publish 2 copies of $L_2 \rightarrow L_3 \rightarrow L_1 \rightarrow L_2$ and update the counts of all nodes supported by $g(v_8)$, that is, the nodes representing L_1, L_2, L_3, $L_2 \rightarrow L_3$, $L_3 \rightarrow L_1$, $L_1 \rightarrow L_2$, $L_2 \rightarrow L_3 \rightarrow L_1$ and $L_3 \rightarrow L_1 \rightarrow L_2$. ■

6.4.2 Privacy Analysis

We give the formal privacy guarantee of our approach below.

Theorem 6.6. *Algorithm 6.3 satisfies ϵ-differential privacy.* ■

Proof. (Sketch) Due to the subtle correlation of the counts in the same level of $\mathcal{T}$ (i.e., a single sequence can affect multiple counts in a level), the sequential composition and parallel composition properties [87] must be applied with caution. Hence, we prove the theorem by the definition of ϵ-differential privacy. Consider two neighboring databases $\mathcal{D}$ and $\mathcal{D}'$. We first consider Lines $1-18$ of Algorithm 6.3, that is, the construction of $\mathcal{T}$. Let this part be denoted by $\mathcal{A}$. Then we need to prove $\frac{Pr[\mathcal{A}(\mathcal{D})=\mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}')=\mathcal{T}]} \leq e^{\epsilon}$. In essence, $\mathcal{T}$ is built on the noisy answers to a set of count queries (via Laplace mechanism). Let each root-to-leaf path be indexed by j. We denote a node in level i and path j by v_{ij}, its privacy parameter by ϵ_{ij}, and its true count in $\mathcal{D}$ and $\mathcal{D}'$ by $Q(\mathcal{D})_{ij}$ and $Q(\mathcal{D}')_{ij}$, respectively. Then we have

$$\frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} = \prod_{i=1}^{n_{\max}} \prod_{j=1}^{|\mathcal{L}|^{n_{\max}}} \frac{\exp(-\epsilon_{ij} \frac{|c(v_{ij}) - Q(\mathcal{D})_{ij})|}{\ell_{\max}})}{\exp(-\epsilon_{ij} \frac{|c(v_{ij}) - Q(\mathcal{D}')_{ij})|}{\ell_{\max}})} \tag{6.1}$$

We first claim that a single record can only affect *at most* $\ell_{\max}$ root-to-leaf paths. This is due to two facts: 1) all ancestors of a node that is influenced by the

additional record must also be influenced; second, $\sum_j |Q(\mathcal{D})_{ij} - Q(\mathcal{D}')_{ij}| \leq \ell_{\max}$. Therefore, Equation 5 could be rewritten as

$$\frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} = \prod_{i=1}^{n_{\max}} \prod_{j=1}^{\ell_{\max}} \frac{\exp(-\epsilon_{ij} \frac{|c(v_{ij}) - Q(\mathcal{D})_{ij})|}{\ell_{\max}})}{\exp(-\epsilon_{ij} \frac{|c(v_{ij}) - Q(\mathcal{D}')_{ij})|}{\ell_{\max}})}$$

Since $\sum_i \epsilon_{ij} = \epsilon$, we have

$$\begin{aligned}
\frac{Pr[\mathcal{A}(\mathcal{D}) = \mathcal{T}]}{Pr[\mathcal{A}(\mathcal{D}') = \mathcal{T}]} &\leq \exp\left(\frac{\sum_{i=1}^{n_{\max}} \sum_{j=1}^{\ell_{\max}} \epsilon_{ij} |Q(\mathcal{D})_{ij} - Q(\mathcal{D}')_{ij}|}{\ell_{\max}}\right) \\
&\leq \exp\left(\frac{1}{\ell_{\max}} \sum_{j=1}^{\ell_{\max}} \sum_{i=1}^{n_{\max}} \epsilon_{ij}\right) \\
&= \exp\left(\frac{1}{\ell_{\max}} \sum_{j=1}^{\ell_{\max}} \epsilon\right) \\
&= e^{\epsilon}
\end{aligned}$$

Note that ϵ_{ij} is calculated based on *noisy* counts. Hence, the construction of $\mathcal{T}$ satisfies ϵ-differential privacy. In addition to the construction of $\mathcal{T}$, we approximate the nodes' counts and generate the synthetic sequential database in Lines 19 and 20. Since these two steps are conducted on noisy data and do not require access to the original database, they satisfy 0-differential privacy. Therefore, our solution as a whole gives ϵ-differential privacy. □

6.4.3 Performance Analysis

Error Analysis

The error of the sanitized data comes from three major sources: first, using the n-grams with $1 < n \leq h$ to estimate longer n-grams with $h < n \leq \ell_{\max}$ (recall the h is the height of the unextended exploration tree); second, the truncation conducted to limit the effect of a single record; third, the noise added to the n-grams with

Table 6.9: Experimental dataset characteristics.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{L}\|$	$max\|D\|$	$avg\|D\|$
MSNBC	989,818	17	14,795	5.7
STM	1,210,096	342	121	6.7

$1 < n \leq h$ to satisfy differential privacy. We call the first two types of error *approximation error* and the last type of error *Laplace error*. Given a specific ϵ value, the total error of our approach is determined by $\ell_{\max}$ and $n_{\max}$. Intuitively, a smaller $\ell_{\max}$ value incurs larger approximation error, but meanwhile it introduces less Laplace error because of a smaller sensitivity. Analogously, a smaller $n_{\max}$ value causes larger approximation error, but results in more accurate counts. Therefore our goal is to identify good values for $\ell_{\max}$ and $n_{\max}$ that minimize the sum of approximation error and Laplace error. In next section, we experimentally study the effect of varying $\ell_{\max}$ and $n_{\max}$ values on the performance of our solution and provide our insights on selecting good $\ell_{\max}$ and $n_{\max}$ values. In general, our solution is designed to perform stably well under a relatively wide range of $\ell_{\max}$ and $n_{\max}$ values.

Experimental Evaluation

We experimentally evaluate the performance of our solution (referred to as *N-gram*) in terms of two data analysis tasks, namely *count query* and *frequent sequential pattern mining*. As a reference point, for count query, we compare the utility of our solution with the approach proposed in [21], which relies on a prefix tree structure (referred to as *Prefix*); for frequent sequential pattern mining, we compare our approach with both *Prefix* and the method designed in [88] for finding frequent (sub)strings (referred to as *FFS*). Two real-life sequential datasets are used in our experiments. The *MSNBC* dataset describes sequences of URL categories browsed by users in time order on *msnbc.com*. It is publicly available at the UCI machine

learning repository [11]. The *STM* dataset records sequences of stations visited by passengers in time order in the Montreal transportation system. It is provided by the *Société de transport de Montréal* [12]. The detailed characteristics of the datasets are summarized in Table 6.9, where $|\mathcal{D}|$ is the number of records (sequences) in $\mathcal{D}$, $|\mathcal{L}|$ is the universe size, $\max|D|$ is the maximum length of sequences in $\mathcal{D}$, and $avg|D|$ the average length of sequences.

Count query. To evaluate the performance of our approach for count queries, we follow the evaluation scheme that has been widely used in previous works [125], [123], [25] (also used in Section 4.4 and Section 6.3.2). The utility of a count query Q is measured by the *relative error* of its answer on the sanitized sequential database $Q(\widetilde{\mathcal{D}})$ with respect to the true answer on the original database $Q(\mathcal{D})$. Unlike the definition given in Section 6.3.2. We consider a more stringent instantiation of count query: the answer to Q is defined to be the number of occurrences of Q in the database (a single record may contain more than one occurrence of Q). For example, given $Q = L_2 \rightarrow L_3$, its answer over the database in Table 6.2 is 6.

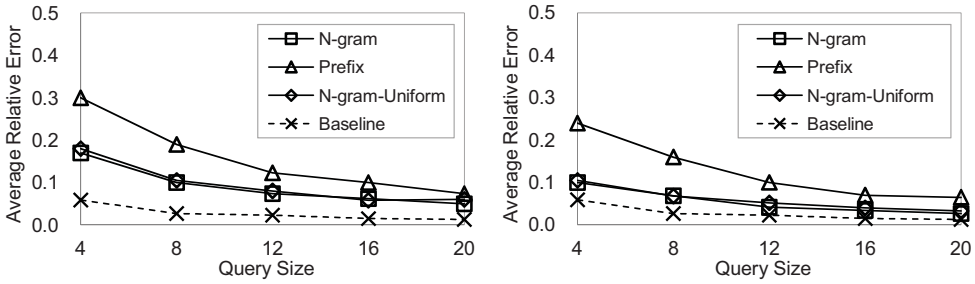
In the first set of experiments, we examine the average relative errors of count queries under different query sizes (i.e., the number of locations in a query) and different privacy budgets. We divide all queries into five subsets with different maximal query sizes (4, 8, 12, 16 and 20). For each subset, we generate 10,000 random queries of sizes that are uniformly distributed at random between 1 and its maximal size. Each location in a query is uniformly selected at random from the location universe.

Figures 6.7 and 6.8 report the average relative errors of *N-gram* and *Prefix* under different query sizes over two typical ϵ values [13] while fixing $\ell_{\max} = 20$ and $n_{\max} = 5$. It can be observed that the average relative errors of *N-gram* are

[11] `http://archive.ics.uci.edu/ml/`

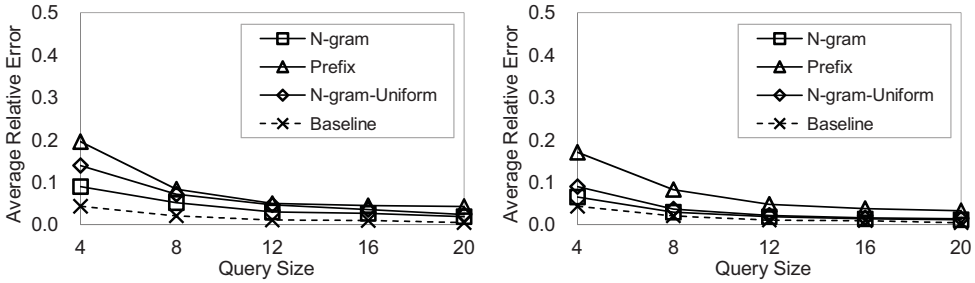
[12] `http://www.stm.info`

[13] According to [88], $\epsilon = 0.1$ and $\epsilon = 1.0$ correspond to *high* and *medium* privacy guarantees, respectively.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Figure 6.7: Average relative error vs. ϵ on *MSNBC*



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Figure 6.8: Average relative error vs. ϵ on *STM*

consistently lower than those of *Prefix* under all settings. The improvements are substantial, ranging from 32% to 63%. The relative errors of *N-gram* are relatively small even under a strong privacy requirement (i.e., $\epsilon = 0.1$).

To demonstrate the effectiveness of the n-gram model, we apply the synthetic sequential database construction technique on *non-noisy* 5-grams of both *MSNBC* and *STM*, and issue count queries on the two synthetic databases (referred to as *Baseline*). The average relative errors of *Baseline* give the approximation error due to the employment of the n-gram model, while the differences between *Baseline* and *N-gram* ascribe to Laplace error. As one can observe, the approximation errors are relatively small on both datasets, demonstrating that the n-gram model is effective in capturing the essential sequentiality information of a database. For Laplace error, we stress that the n-gram model provides a general and flexible framework that accommodates other more advanced noise injection mechanisms, such as the *matrix*

mechanism [76] and the *MWEM mechanism* [57]. Hence it may require less noise added than Laplace mechanism, resulting in smaller Laplace error. It may even allow a larger $n_{\max}$ value to be used and therefore further reduce approximation error. Thus, we deem that the variable-length n-gram model bears great promise for differentially private sequential data release.

To prove the benefit of our adaptive privacy budget allocation scheme, we report the average relative errors of a variant of *N-gram* (referred to as *N-gram-Uniform*), in which the adaptive allocation scheme is replaced by the uniform allocation scheme described before. The improvement is less obvious on $MSNBC$ because many paths are actually of length n_{max}, whereas the improvement on STM is noticeable, especially when $\epsilon = 0.1$.

Due to the truncation operation conducted in Algorithm 6.3, any count query with a size greater than $\ell_{\max}$ receives an answer 0 on the sanitized dataset. However, we point out that in reality it is *not* a problem because the true answer of such a query is typically very small (if not 0). For many real-life analyses (e.g., ridership analysis), the difference between such a small value and 0 is negligible. In addition, this limitation also exists in *Prefix* and is inherent in any differentially private mechanism because Laplace mechanism cannot generate reliable answers on extremely small values.

Next we examine the impact of $\ell_{\max}$ and $n_{\max}$ on average relative error of count queries. In Figure 6.9, we study how relative error changes under different $\ell_{\max}$ values with $\epsilon = 1.0$, $n_{max} = 5$ and query size equal to 8. In theory, a larger $\ell_{\max}$ value allows more information of the underlying database to be retained at the cost of higher sensitivity (and hence larger Laplace noise). Therefore, the selection of $\ell_{\max}$ needs to take into consideration the trade-off between approximation error and Laplace error. However, in reality, a reasonable $\ell_{\max}$ value could be chosen more easily because the average sequence length of many real-life datasets is relatively small. Consequently, Laplace error is the major concern in this case. This is confirmed

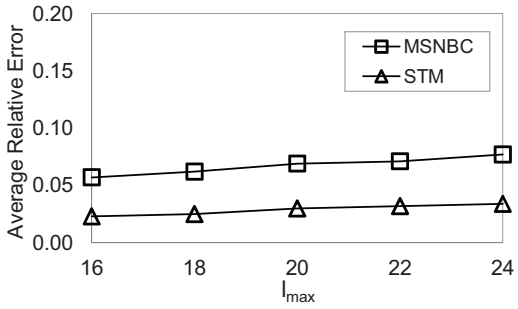


Figure 6.9: Average relative error vs. $\ell_{\max}$ ($\epsilon = 1.0$)

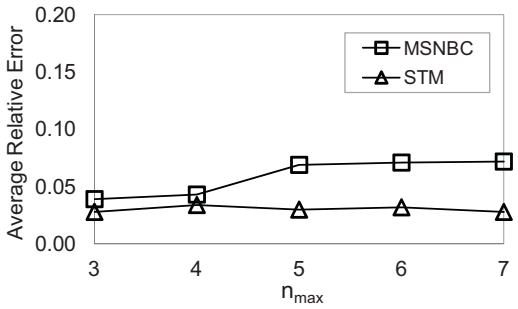
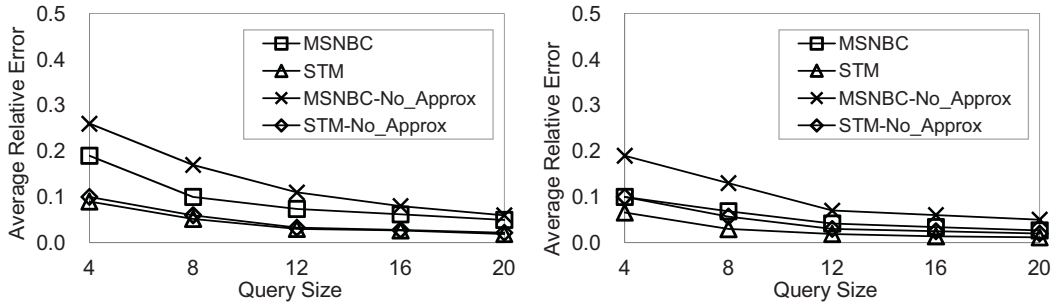


Figure 6.10: Average relative error vs. $n_{\max}$ ($\epsilon = 1.0$)

by Figure 6.9. Since most sequences in $MSNBC$ and STM are of a small length, when $\ell_{\max}$ is sufficiently large (i.e., 16), increasing $\ell_{\max}$ does not significantly lower approximation error, but simply increases Laplace noise. Moreover, we can observe that our approach performs relatively stable under varying $\ell_{\max}$ values. This can be explained by the large counts of short grams, which are more resistant to Laplace noise.

Figure 6.10 examines the performance of *N-gram* with respect to varying $n_{\max}$ values, where $\epsilon = 1.0$, $\ell_{\max} = 20$ and query size is 8. Similar to the selection of $\ell_{\max}$, the selection of $n_{\max}$ also involves the trade-off between approximation error and Laplace error. A larger $n_{\max}$ reduces approximation error while increasing Laplace error. To obtain a reasonable trade-off, we develop the adaptive privacy budget allocation scheme and the formal choice of the threshold value, which automatically select the best gram sizes on the fly. Even a data holder specifies a unreasonably large $n_{\max}$, our approach may end up with shorter grams. Therefore, it can be



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Figure 6.11: Effect of node count approximation.

observed that the performance of our solution is insensitive to varying $n_{\max}$ values. From our experimental results, we believe that, in most cases, $n_{\max} = 5$ is a good choice. In addition, we point out that a good $n_{\max}$ value is related to $|\mathcal{D}|$ and $|\mathcal{L}|$, a larger $|\mathcal{D}|$ or a smaller $|\mathcal{L}|$ suggests a larger $n_{\max}$ value.

One key technique that we develop to improve accuracy of count queries is to enforce consistency constraints by approximating the counts of the nodes that cannot pass the threshold. In the next set of experiments, we demonstrate that this technique indeed improves the accuracy of count queries compared to the case when we naively set the noisy counts of all nodes that do not pass the threshold to 0. In Figure 6.11, $\ell_{\max} = 20$ and $n_{\max} = 5$. *MSNBC_No-Approx* and *STM_No-Approx* give the relative errors of *N-gram* without the approximation technique. As we can observe, this technique improves the relative error for all query sizes under different ϵ values, up to 47%.

Frequent sequential pattern mining. The second data analysis task we consider is frequent sequential pattern mining, a more specific data mining task. Given a positive number K, we are interested in the top K most frequent sequential patterns (i.e., most frequent sub-sequences) in the dataset. This data analysis task helps, for example, a transportation agency better understand passengers' transit patterns and consequently optimize its network geometry.

Table 6.10: True positive ratio vs. K value on $MSNBC$

(a) $\epsilon = 0.1$

K value	20	40	60	80	100
N-gram	100%	90%	93%	96%	94%
Prefix	85%	78%	80%	84%	86%
FFS	70%	63%	57%	58%	55%

(b) $\epsilon = 1.0$

K value	20	40	60	80	100
N-gram	100%	93%	97%	99%	97%
Prefix	90%	82.5%	85%	90%	89%
FFS	70%	63%	57%	58%	55%

We compare the performance of *N-gram* with *Prefix* and *FFS*. All size-1 frequent patterns are excluded from the results since they are of less interest and trivial in frequent sequential pattern mining. We would like to clarify that *FFS* actually has two assumptions: 1) all frequent patterns are of the same length; 2) the lengths of frequent patterns are identical to the lengths of input sequences. Since generally these two assumptions cannot be satisfied in a frequent sequential pattern mining task, it is not fair to directly compare *FFS* with *N-gram* and *Prefix*. However, there are very few approaches that support frequent sequential pattern mining under differential privacy. Hence we still report the performance of *FFS* and provide insights on the key factor that guarantees high utility on frequent sequential pattern mining. For both *FFS* and *Prefix*, we have tested various parameter settings and report the best results we have obtained.

To give an intuitive impression on the performance of these three approaches, we first report their *true positive ratios* under different K and ϵ values in Table 6.10 and Table 6.11. Given K, we generate the top K most frequent sequential patterns on both the original dataset $\mathcal{D}$ and the sanitized dataset $\widetilde{\mathcal{D}}$, which are denoted by $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, respectively. The true positive ratio is then defined to be the percentage of frequent patterns that are correctly identified, that is, $\frac{|\mathcal{F}_K(\mathcal{D}) \cap \mathcal{F}_K(\widetilde{\mathcal{D}})|}{K}$. The results indicate that *N-gram* can reliably identify the most frequent patterns in

Table 6.11: True positive ratio vs. K value on STM

(a) $\epsilon = 0.1$

K value	20	40	60	80	100
N-gram	95%	93%	93%	94%	91%
Prefix	65%	68%	75%	83%	82%
FFS	35%	33%	35%	36%	43%

(b) $\epsilon = 1.0$

K value	20	40	60	80	100
N-gram	100%	100%	98%	100%	98%
Prefix	70%	68%	80%	86%	85%
FFS	35%	33%	35%	36%	43%

a given database with strong privacy guarantee.

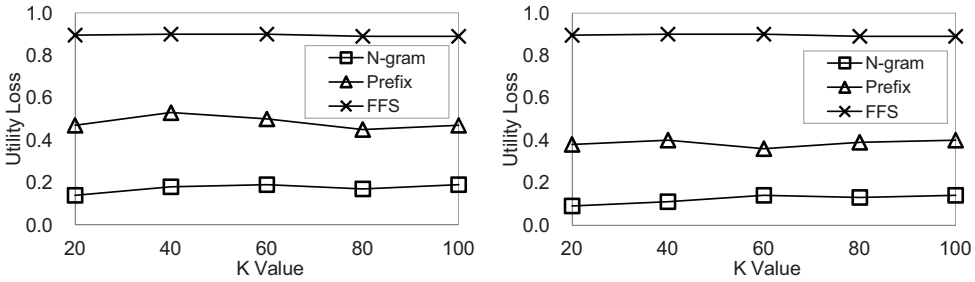
To measure the utility of sanitized data more precisely, we adopt the metric proposed in [25], which further takes into consideration the accuracy of the supports of patterns in $F_K(\widetilde{\mathcal{D}})$ [14]. The utility loss on the sanitized dataset is defined to be the difference between $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, that is,

$$\frac{\sum_{F_i \in F_K(\mathcal{D})} \frac{|sup(F_i, F_K(\mathcal{D})) - sup(F_i, F_K(\widetilde{\mathcal{D}}))|}{sup(F_i, F_K(\mathcal{D}))}}{K},$$

where $sup(F_i, F_K(\mathcal{D}))$ and $sup(F_i, F_K(\widetilde{\mathcal{D}}))$ denote the supports of F_i in $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$, respectively. If $F_i \notin F_K(\widetilde{\mathcal{D}})$, $sup(F_i, F_K(\widetilde{\mathcal{D}})) = 0$. Therefore, if the metric equals 0, it means that $F_K(\mathcal{D})$ is identical to $F_K(\widetilde{\mathcal{D}})$ (even the support of every frequent pattern); if the metric equals 1, it implies that $F_K(\mathcal{D})$ and $F_K(\widetilde{\mathcal{D}})$ are totally different.

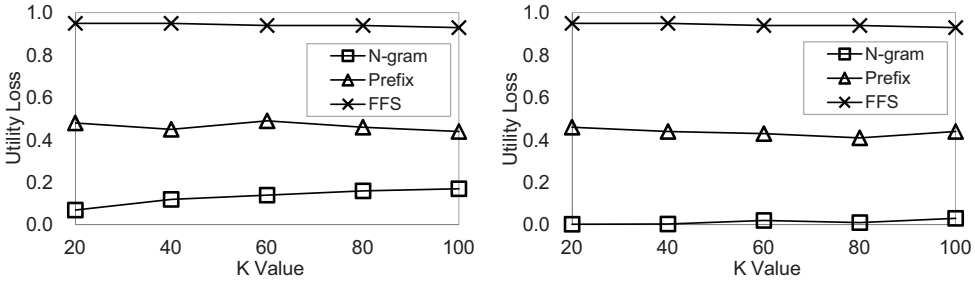
In Figure 6.12 and Figure 6.13, where $\ell_{\max} = 20$ and $n_{\max} = 5$, we can observe that our proposal significantly outperforms the other two approaches. In addition, for the frequent patterns that are correctly identified, the relative error of their supports is typically very small even when $\epsilon = 0.1$. The main reason is that *N-gram*

[14] The support of a pattern is the number of its occurrences in a database.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Figure 6.12: Utility loss vs. K on *MSNBC*



(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Figure 6.13: Utility loss vs. K on *STM*

extracts the essential information of a database in terms of a set of n-grams, which are actually the most frequent patterns in the database. This fact allows *N-gram* to perform well even under a small ϵ value. In contrast, in *Prefix*, the noise of a frequent pattern's count accumulates quickly in proportion to the number of longer sequences which contain the frequent pattern. The major limitation of *FFS* is its prefix data structure, which generates frequent patterns based on very short prefixes.

In the last set of experiments, we study the impact of $\ell_{\max}$ and $n_{\max}$ on frequent sequential pattern mining. Figure 6.14 reports the utility loss of *N-gram* under different $\ell_{\max}$ with $\epsilon = 1.0$ and $n_{\max} = 5$. The aforementioned trade-off in the selection of $\ell_{\max}$ still applies to frequent sequential pattern mining. This time, we can clearly observe such a trade-off in Figure 6.14: when $\ell_{\max}$ is small, the approximate error is the main source of error; when $\ell_{\max}$ becomes larger, the total error is dominated by Laplace error. Nevertheless, *N-gram* can provide good utility

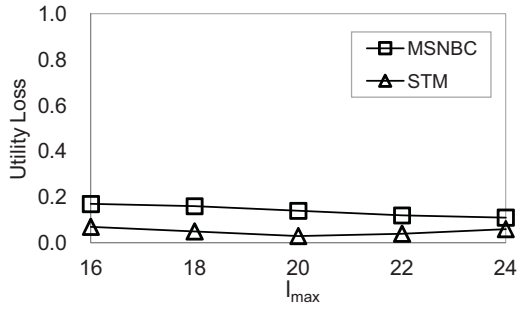


Figure 6.14: Utility loss vs. $\ell_{\max}$ ($\epsilon = 1.0$)

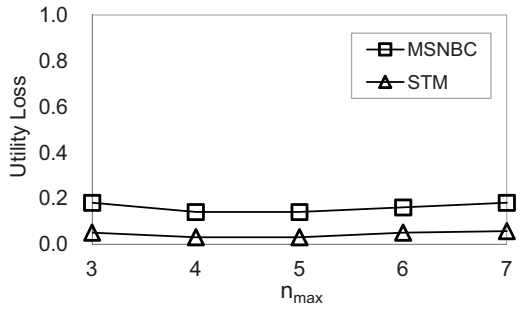


Figure 6.15: Utility loss vs. $n_{\max}$ ($\epsilon = 1.0$)

for a wide range of $\ell_{\max}$ values. This property makes it easier for a data holder to select a good $\ell_{\max}$ value.

Similar trade-off due to $n_{\max}$ can be observed in Figure 6.15, where $\ell_{\max}$ is fixed to 20. There exists a $n_{\max}$ value that minimizes the sum of approximation error and Laplace error. Due to the series of techniques we propose, the utility lost under different $n_{\max}$ values is comparable.

6.4.4 Summary

In this section, we proposed a novel approach of differentially private sequential data publication based on a variable-length n-gram model. This model extracts the essential information of a sequential database in terms of a set of variable-length n-grams whose counts are relatively large and therefore subject to lower Laplace error. We developed a set of key techniques that are vital to the success

of the n-gram model. Furthermore, we designed a synthetic sequential database construction method, which allows published n-grams to be used for a wider range of data analysis tasks. Extensive experiments on real-life datasets proved that our solution substantially outperforms state-of-the-art techniques [88], [21] in terms of count query and frequent sequential pattern mining.

Chapter 7

Network Data Sanitization

7.1 Introduction

In the last few years, information networks in various application domains, such as social networks, communication networks and transportation networks, have experienced vigorous developments. In particular, social networks, such as Facebook, LinkedIn and Myspace, have become very prevalent. With the growth of information networks, a large volume of network data has been generated, which enables a wide spectrum of data analysis tasks. Network data is typically represented as graphs, where nodes represent a set of individuals with their attributes, and edges represent connections between them. Therefore, in this chapter we use the term *network data* and *graph* interchangeably.

It has been shown that with naively sanitized network data (e.g., merely replacing explicit identifiers by pseudo-identifiers), an adversary is able to launch different types of privacy attacks that re-identify nodes, reveal edges between nodes, or expose node attributes [59]. Therefore, network data needs to be sanitized with formal, provable privacy guarantees before it can be released to the public.

In addressing privacy issues in network data publication, there has been a series of research [133], [60], [82], [131], [30], [134], [83], [58], [122], [26], [132], [54] based on

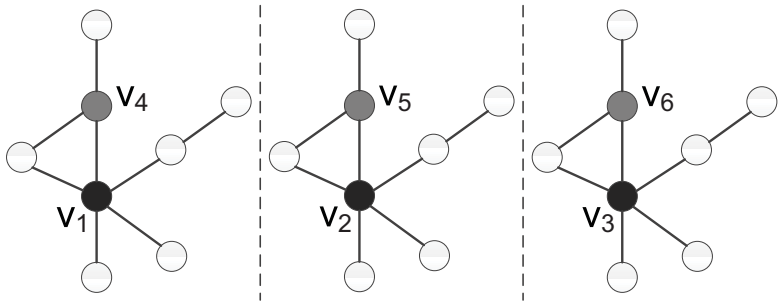


Figure 7.1: k-isomorphism is insufficient for preventing edge disclosure.

different privacy models. Most of these works [133], [60], [82], [30], [134], [26], [132] focus on preventing node re-identification and/or associated attribute disclosure against adversaries with structural background knowledge. In contrast, only a few papers [131], [83], [122], [26], [54] consider privacy threats due to *edge disclosure*, which lets an adversary learn the sensitive relationships between individuals. Moreover, the papers [131], [83], [122] lack a formal privacy definition for edge anonymity. Among all privacy models for network data, k-isomorphism [26] provides relatively strong privacy protection for edge disclosure (i.e., an adversary cannot determine if two individuals are connected via a path with a probability $\geq \frac{1}{k}$). However, we show that, with moderate background knowledge, an adversary is able to ascertain a direct link between two individuals on a k-isomorphic graph, as illustrated in Example 7.1.1.

Example 7.1.1. Consider the 3-isomorphic graph in Figure 7.1. Suppose that the adversary has successfully identified Bob as one of $\{v_1, v_2, v_3\}$ and Ann as one of $\{v_4, v_5, v_6\}$ and seeks to learn if there is a direct link between Bob and Ann, which is considered sensitive. With the background knowledge that Bob and Ann are connected via a common friend, the adversary can ascertain that both Bob and Ann are in the same subgraph, and therefore learn that there is a direct link between Bob and Ann. ■

The vulnerability of k-isomorphism is largely due to its deterministic nature.

This fact motivates our use of *differential privacy* [37], which requires *inherent randomness* of a sanitization algorithm. The traditional ϵ-differential privacy provides rigorous privacy guarantees on a database whose records are generated *independently*; however, its application to network data is hindered by the fact that network data may be inherently *correlated*. In the context of network data, the *evidence of participation* [69] of a record (e.g., an edge) may be reflected by several other records. For example, the presence of an edge in a network database can be inferred by the existence of several other edges. The deletion of a single edge will not be able to fully mask its presence in the database. Consequently, ϵ-differential privacy fails to provide the claimed privacy guarantee in the correlated setting (e.g., an adversary can obtain a probability change greater than e^{ϵ}).

In addressing this issue, we propose a stronger variant of differential privacy, called (ϵ, k)-*differential privacy*, which provides provable privacy protection even when a record is correlated to *at most* $k-1$ other records. We then quantify the relationship between *sensitivity* and data correlation, and derive the concept of *correlated sensitivity*, which enables the standard mechanisms, Laplace mechanism [37] and exponential mechanism [89], to be used for achieving (ϵ, k)-differential privacy.

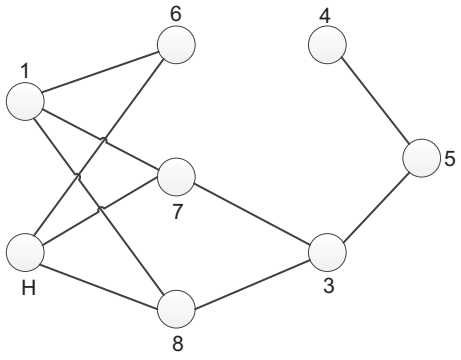
In addition to correlation, another major challenge of applying (ϵ, k)-differential privacy (or ϵ-differential privacy) to network data is scalability and utility. This is confirmed by the recent paper [54], to the best of our knowledge, the only existing work that studies network data publication in the *non-interactive* setting under differential privacy. It requires the input graph to be *dense*, which is very unlikely to be satisfiable on real-life datasets, and leaves finding an efficient algorithm as an open problem. In this chapter, we follow the line of *data-dependent* solutions [25], [90], which adaptively make use of noisy information from the underlying database to improve efficiency and effectiveness. We first explore dense regions of the adjacency matrix of an input graph using an adapted quadtree, which avoids the high complexity of graph operations, and then propose an efficient use of exponential mechanism

to reconstruct the leaf nodes of the quadtree while satisfying (ϵ, k)-differential privacy.

Contributions. Our contributions in network data sanitization are summarized as follows.

- First, we propose (ϵ, k)-differential privacy, a stronger variant of ϵ-differential privacy [37], which guarantees provable privacy protection when the underlying data is correlated (Section 7.3). (ϵ, k)-differential privacy is a general, practical version of ϵ-differential privacy and free-lunch privacy [69]. It applies to not only network data but also any type of data that is correlated. We introduce the notion of *correlated sensitivity*, which allows Laplace mechanism and exponential mechanism to be used for achieving (ϵ, k)-differential privacy, and show that (ϵ, k)-differential privacy inherits the composition properties of ϵ-differential privacy.

- Second, based on (ϵ, k)-differential privacy, we propose an efficient *non-interactive* solution for network data publication, which prevents an adversary from learning the existence of a direct link between any two individuals (Section 7.4). This is the first *efficient* non-interactive solution in the framework of differential privacy. Compared with the only previous work [54] on non-interactively releasing a private graph under differential privacy, our improvements are significant: (1) we achieve the stronger (ϵ, k)-differential privacy, whereas Gupta et al. [54] achieve (ϵ, δ)-differential privacy, a weaker privacy notion of ϵ-differential privacy; (2) we lift the impractical assumption that the input graph has to be dense. We show that theoretically our approach obtains high utility as long as sufficient edge information is contained in some dense subgraphs and that experimentally our approach performs very well on many different types of real-life network datasets; (3) most importantly, our approach is efficient to handle large real-life datasets.



(a) A sample graph G

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	1	1	1
2	0	0	0	0	0	1	1	1
3	0	0	0	0	1	0	1	1
4	0	0	0	0	1	0	0	0
5	0	0	1	1	0	0	0	0
6	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0
8	1	1	1	0	0	0	0	0

(b) The adjacency matrix $A(G)$

Figure 7.2: A sample graph and its adjacency matrix.

- Third, we conduct an extensive experimental study over various types of real-life network datasets, including social network, collaboration network and transportation network (Section 7.5). We examine the utility of sanitized data for different data analysis tasks, namely *degree distribution* and *cut queries*. We demonstrate that our approach maintains high utility and scales to large real-life network data.

The results of this chapter are currently under review in [24].

7.2 Preliminaries

7.2.1 Network Data and Adjacency Matrix

In this thesis, we follow the convention of modeling an input network dataset as a *simple graph* (i.e., an undirected graph with no loops or multiple edges), $G = (V, E)$, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges. For a graph G, we use $V(G)$ and $E(G)$ to respectively denote the vertex set and the edge set of G. When the graph is clear from the context, we omit G from the notation. We assume that a vertex labeling has been given in a way that is independent of the underlying edge set.

The *adjacency matrix* of a vertex-labeled simple graph $G = (V, E)$, denoted by $A(G)$, is a square $|V| \times |V|$ matrix satisfying:

$$A(G)_{ij} = \begin{cases} 1 & if (v_i, v_j) \in E(G) \\ 0 & otherwise \end{cases}$$

It is evident that, for any simple graph G, $A(G)$ is a symmetric matrix with a zero diagonal. Figure 7.2 presents the adjacency matrix of a given simple graph (ignore the bold lines for now). A $(0, 1)$-matrix is called a *graphic* matrix if it is an adjacency matrix of some simple graph. Apparently, a $(0, 1)$-matrix is graphic if and only if it is a symmetric matrix with a zero diagonal.

We define the density of a region $R \subseteq A$ with size $|R| = m \times l$ to be $den(R) = \sum_{i=1}^{m} \sum_{j=1}^{l} R_{ij}/ml$. It gives the fraction of elements in R which are equal to 1. The region in A formed by rows $i, \cdots, j$ and columns $m, \cdots, n$ is denoted by $A[i, j; m, n]$. For example, the densities of A and $A[1, 3; 6, 8]$ are $20/64 = 31.25\%$ and $8/9 = 88.89\%$, respectively.

7.2.2 Utility Requirement

Our general goal is to generate a sanitized graph $\widetilde{G}$, whose adjacency matrix $\widetilde{A}$ minimizes $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}|$ (i.e., $\widetilde{A}$ is as close to A as possible). When $\widetilde{A}$ is identical to A, $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}| = 0$; when $\widetilde{A}$ is totally different from A, $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}| = |V|^2 - |V|$. Minimizing $\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} |A_{ij} - \widetilde{A}_{ij}|$ naturally makes the published network data useful for many analysis tasks, including *degree distribution* and *cut queries*.

Degree distribution. Given a graph G, we use a vector $F(G)$ of size $|V(G)|$ to denote the *degree frequency sequence* of G. The i-th value in $F(G)$ is $\frac{|\{v \in V : deg(v) = i\}|}{|V|}$, where $deg(v)$ is the degree of v. For example, the degree frequency sequence of the graph in Figure 7.2 is {0, 0.125, 0.25, 0.625, 0, 0, 0, 0}. Given the degree

frequency sequences $F(G)$ and $F(\widetilde{G})$, we measure their difference by *Kullback-Leibler divergence* (KL-divergence) [68]:

$$D_{KL}(F(G)||F(\widetilde{G})) = \sum_{i=0}^{|V|-1} F(G)[i] \log \frac{F(G)[i]}{F(\widetilde{G})[i]}. \tag{7.1}$$

If $F(G) = F(\widetilde{G})$, $D_{KL}(F(G)||F(\widetilde{G})) = 0$. We follow the standard convention that $0 \log 0 = 0$.

Cut query. In this chapter, a *cut* of a graph G is defined by any two subsets of vertices $S, T \subseteq V(G)$ [54]. A *cut query* returns the number of edges in the *cut-set* of a cut, that is, $Q_{S,T}(G) = |\{(u, v) \in E(G) : u \in S, v \in T\}|$. For example, given the graph in Figure 7.2, $S = \{v_1, v_2\}$ and $T = \{v_6, v_7, v_8\}$, we have $Q_{S,T} = 6$.

Similarly, we measure the utility loss of a cut query over a sanitized graph $\widetilde{G}$ by its *relative error*, with respect to the true count over the original graph G, which is computed as:

$$error(Q_{S,T}(\widetilde{G})) = \frac{|Q_{S,T}(\widetilde{G}) - Q_{S,T}(G)|}{\max\{Q_{S,T}(G), s\}},$$

where s is a *sanity bound* that mitigates the effect of the queries with extremely small *selectivities* [125], [123], [25].

7.3 (ϵ, k)-Differential Privacy

ϵ-differential privacy is built on the assumption that all underlying records are independent of each other. In the context of network data, this assumption does not always hold. Kifer and Machanavajjhala [69] indicates that in the correlated setting, ϵ-differential privacy cannot provide the claimed privacy protection because the removal of a single record cannot hide its *evidence of participation* (e.g., its participation could still be inferred by the existence of some other records to which it is correlated). In this section, we propose a stronger variant of ϵ-differential privacy,

known as (ϵ, k)-differential privacy, which provides guaranteed privacy even when the underlying records are correlated.

Definition 7.1 ((ϵ, k)-differential privacy)**.** A privacy mechanism $\mathcal{A}$ satisfies (ϵ, k)-differential privacy if for any two databases D_1 and D_2 that differ on at most one record, and for any possible output $O \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) = O] \leq e^{\frac{\epsilon}{k}} \times Pr[\mathcal{A}(D_2) = O] \tag{7.2}$$

where the probability is taken over the randomness of $\mathcal{A}$, and k is a measure of the extent of correlation. ■

Before we formalize the privacy guarantee provided by (ϵ, k)-differential privacy with respect to correlation, we first define a k-correlated database.

Definition 7.2 (k-correlated database)**.** A database D is k-correlated if the existence of any record in D can be inferred by at most $k-1$ other records. ■

A k'-correlated database is called *k-correlation bounded* if $1 \leq k' \leq k$. The fundamental observation under (ϵ, k)-differential privacy is that *in the correlated setting an adversary's probability change is not bounded by the probability change of a privacy mechanism, but may be magnified by the extent of correlation due to inference.*

Theorem 7.1. *If neighboring databases are k-correlation bounded, then (ϵ, k)-differential privacy bounds an adversary's probability change by e^{ϵ}.* ■

Proof. For k-correlation bounded databases, it is sufficient to cancel out the effect of data correlation on any computation by considering all correlated records (*at most* k records) as if they were removed from the underlying database. Therefore, bounding an adversary's probability change by e^{ϵ} on two databases differing on one record in the correlated setting is equivalent to bound the probability change of

a privacy mechanism by e^{ϵ} over two databases differing on k records in the *non-correlated* setting. A privacy mechanism $\mathcal{A}$ that achieves ϵ-differential privacy on two databases differing on k records gives $\frac{\epsilon}{k}$-differential privacy on two databases differing on one record [37]. □

We will see how (ϵ, k)-differential privacy helps thwart the privacy attack described in [69] after giving the definition of *correlated sensitivity*. (ϵ, k)-differential privacy satisfies the monotonic property below.

Theorem 7.2. *A (ϵ, k)-differentially private mechanism $\mathcal{A}$ gives (ϵ, k')-differential privacy for all $1 \leq k' \leq k$.* ■

This is true because $\frac{Pr[\mathcal{A}(D_1)=O]}{Pr[\mathcal{A}(D_2)=O]} \leq e^{\frac{\epsilon}{k}} \leq e^{\frac{\epsilon}{k'}}$. Theorem 7.2 indicates that if k is specified as the upper bound of correlation, (ϵ, k)-differential privacy can always bound an adversary's probability change by e^{ϵ}. (ϵ, k)-differential privacy is a generalized, practical version of ϵ-differential privacy [37] with $k = 1$ and free-lunch privacy [69] with $k = n$, where n is the database size. The definition of (ϵ, k)-differential privacy is realistic and practical in many applications. In any case, data with extremely strong correlation (e.g., $k = O(n)$) cannot be published with useful information under *any* privacy model. The k value may vary from application to application. We leave how to determine a reasonable k value in a specific application as an open problem (meanwhile, we experimentally show that our solution can preserve useful information even when k is relatively large).

The traditional ϵ-differential privacy can be achieved by Laplace mechanism and exponential mechanism by properly defining global sensitivity. Similarly, (ϵ, k)-differential privacy can be achieved by Laplace mechanism and exponential mechanism based on the concept of *correlated sensitivity*.

Definition 7.3 (Correlated Sensitivity)**.** For any function $f : D \to \mathbb{R}^d$, the correlated sensitivity of f is

$$CS(f) = k \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1 \tag{7.3}$$

for all k-correlated databases D_1, D_2 s. t. D_1 and D_2 differ on at most one record. ■

Correlated sensitivity is defined to be k times of global sensitivity, which implies that extra noise is needed in order to hide the effect of the k correlated records. We now show that applying correlated sensitivity to Laplace mechanism and exponential mechanism gives (ϵ, k)-differential privacy.

Theorem 7.3. *The substitution of correlated sensitivity for global sensitivity in Laplace mechanism and exponential mechanism achieves (ϵ, k)-differential privacy.* ■

Proof. By definition, (ϵ, k)-differential privacy requires to reduce the probability change of a private mechanism to $e^{\frac{\epsilon}{k}}$ in order to cancel out the effect of data correlation. For either Laplace mechanism and exponential mechanism, this can be achieved by increasing global sensitivity k times, which is correlated sensitivity. □

We revisit the example provided in [69] under correlated sensitivity to see how it thwarts the privacy attack described below.

Example 7.3.1. [1] Bob or one of his 9 immediate family members may have contracted a highly infectious disease, in which case the entire family would have been infected. An attacker asks the query "how many in Bob's family have this disease?" to infer if Bob has been infected. The true answer is of high probability to be either 0 or 10. Suppose the noisy answer returned is 12. If this answer is obtained by adding Laplace noise based on global sensitivity, the attacker learns that the probability of

[1] The strong attacker mentioned in [69] cannot be prevented as his prior knowledge (without accessing any database) has allowed him to succeed in a privacy attack.

10 being the true answer is $e^{10\epsilon}$ times larger than the probability of 0 being the true answer. In contrast, if the answer is obtained by adding Laplace noise based on correlated sensitivity (in this case, $k \geq 10$), we have $\frac{Pr[c=10|\widetilde{c}=12]}{Pr[c=0|\widetilde{c}=12]} \leq \frac{exp(-\frac{\epsilon|10-12|}{10})}{exp(-\frac{\epsilon|0-12|}{10})} \leq exp(\epsilon)$. It ensures that an attacker's probability estimate can change by a factor of at most e^{ϵ}. ■

In the rest of the chapter, we consider a specific instantiation of (ϵ, k)-differential privacy in the context of network data. Two neighboring databases are defined as two databases that differ on at most one *edge*. This instantiation prevents an adversary from learning the presence of any single edge (i.e., if two individuals are directly connected) even when its existence can be inferred by $k-1$ other edges.

7.4 Sanitization Algorithm

We provide an overview of our solution, called *density-based exploration and reconstruction* (DER), in Algorithm 7.1, which takes as inputs a graph G, a privacy "budget" ϵ, and a correlation parameter k, and returns a sanitized graph $\widetilde{G}$ satisfying (ϵ, k)-differential privacy. Our solution consists of two steps, and therefore ϵ is divided into two portions, ϵ' and ϵ'', each being used in a step.

First, we design a differentially private and data-dependent partitioning process by adapting a standard quadtree to explore dense regions of the adjacency matrix A of G, which can be reconstructed later with high accuracy. This process results in a noisy quadtree $\mathcal{QT}$ whose nodes represent a region of A and are associated with a noisy count. The major technical challenges in this step include the design of stop conditions based on an estimation of the height of $\mathcal{QT}$, the selection of splitting points based on exponential mechanism, an adaptive privacy budget allocation scheme and an efficient implementation, which are key to the success of the entire algorithm.

Second, we propose an efficient edge arrangement algorithm to reconstruct a

Algorithm 7.1: DER Algorithm

Input: Raw graph G
Input: Privacy budget ϵ
Input: Correlation parameter k
Output: Sanitized graph $\widetilde{G}$

1: $\epsilon = \epsilon' + \epsilon''$;
2: Generate the adjacency matrix A from G;
3: Noisy quadtree $\mathcal{QT} \leftarrow ExploreDenseRegion(A, \epsilon', k)$;
4: Sanitized graphic matrix $\widetilde{A} \leftarrow ArrangeEdge(\mathcal{QT}, A, \epsilon'', k)$;
5: Generate $\widetilde{G}$ from $\widetilde{A}$;
6: **return** $\widetilde{G}$;

noisy, graphic matrix $\widetilde{A}$ that minimizes $\sum_{i=1}^{|V(G)|} \sum_{j=1}^{|V(G)|} |A_{ij} - \widetilde{A}_{ij}|$. Our method is based on a novel use of exponential mechanism, which is of independent interest. It successfully reduces the run-time complexity to $O(|V|^2)$, in contrast to the *factorial complexity* of a naive implementation.

7.4.1 Dense Region Exploration

Based on the adjacency matrix A, we perform a recursively partitioning process guided by *density* in order to identify dense regions (and, implicitly, sparse regions) of A, which can be reconstructed accurately. This process could be done based on many popular space-partitioning data structures, such as *kd-tree* [10], *quadtree* [42], and *Hilbert R-tree* [65]. In this section, we employ *quadtree* as the basic data structure for the exploration because it achieves the best trade-off between utility and efficiency.

A standard quadtree decomposes a given two dimensional region into four *equal* quadrants, subquadrants, and so on with each leaf node meeting certain *stop condition*. The splitting point of a standard quadtree is independent of the input data (e.g., it always selects the midpoint of each dimension). Each node in a quadtree represents a region of A. For our task, we adapt a quadtree in a data-dependent, differentially private manner. Each node (except the root) in a quadtree records not

Algorithm 7.2: *ExploreDenseRegion*

Input: Raw adjacency matrix A
Input: Privacy budget ϵ'
Input: Correlation parameter k
Output: Noisy quadtree $\mathcal{QT}$

```
1: i = 0;
2: QT ← ∅;
3: Calculate the height h of QT;
4: while i < h do
5:    if i = 0 then
6:       Insert a node representing A to QT;
7:    end if
8:    for each non-leaf node u ∈ level(i, QT) do
9:       Calculate privacy budget portion ε_u^c and ε_u^p;
10:      Subregions R ← partition(u, ε_u^p, k);
11:      for each R ∈ R do
12:         c̃ = NoisyCount(R, ε_u^c);
13:         Insert a node v representing R to QT;
14:         if v meets stop condition then
15:            Mark v as leaf;
16:         end if
17:      end for
18:   end for
19:   i++;
20: end while
21: return QT;
```

only the region it represents, but also a noisy count of the number of 1's in its region. We abuse the term *count* to mean the number of 1's in a region. Algorithm 7.2 presents the details of *ExploreDenseRegion*.

Stop condition. One key problem in the partitioning process is to determine the height of the quadtree. Previous works [90], [28] normally require a user to specify the height. In our work, we calculate a good estimate of the height based on other inputs (Line 3). It is very difficult to calculate a very precise height of a *data-dependent* quadtree with our *adaptive* privacy budget allocation scheme. Instead, we use a standard quadtree with the *geometric budget scheme* [28] to derive a reasonably good estimate. A geometric budget scheme assigns all nodes on the same level i the

same privacy budget ϵ_i, and increases the budget by a factor of $2^{1/3}$ with the increase of nodes' depth.

Theorem 7.4. *[28] Given the privacy budget ϵ_{count}, the geometric budget scheme, assigning $\frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$ to nodes with depth i in a quadtree of height h, achieves maximum accuracy for range queries.* ■

According to Theorem 7.4, the privacy budget allocated to leaf regions is $\epsilon_h = \frac{2^{h/3}(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$. Observing that the size of a leaf region in a standard quadtree is $\frac{|V|^2}{4^h}$, we calculate h by requiring the size of a leaf region to be greater than twice of the noise's standard deviation because we cannot get useful noisy counts on overly small regions, that is:

$$\begin{aligned} \frac{|V|^2}{4^h} &\geq \frac{2\sqrt{2}CS(f)}{\epsilon_h} \\ &= \frac{2\sqrt{2}(2^{(h+1)/3}-1)CS(f)}{2^{h/3}(\sqrt[3]{2}-1)\epsilon_{count}} \end{aligned}$$

where f is a count query. Apparently, in the k-correlated setting, for a count query over a region R of A, *in the worst case*, the correlated sensitivity $CS(f) = \min\{2k, |R|\}$ [2]. From the above equation, we get:

$$\sqrt[3]{2}(2^h)^2 - (2^h)^{5/3} \leq \frac{(\sqrt[3]{2}-1)|V|^2\epsilon_{count}}{2\sqrt{2}CS(f)} \tag{7.4}$$

Theorem 7.5. *$f(h) = \sqrt[3]{2}(2^h)^2 - (2^h)^{5/3}$ monotonically increases on $[0, +\infty)$.* ■

Proof. Let $t = 2^h$. $\forall h \geq 0$, $t \geq 1$. Plugging t to the equation, we get:

$$f'(t) = 2\sqrt[3]{2}t - \frac{5}{3}t^{\frac{2}{3}} \geq \frac{5}{3}t^{\frac{2}{3}}(t^{\frac{1}{3}} - 1) > 0$$

on $(1, +\infty)$. This completes the proof. □

[2] For a region R not containing elements on the diagonal, $CS(f) = \min\{k, |R|\}$ because a single edge difference cannot change two elements in this region.

Because the left hand side (LHS) of Equation 8 increases monotonically, we can always find a maximal h value satisfying Equation 8. This maximal h value is then used to be the height of the quadtree. In Section 7.5, we demonstrate that this estimate performs very well for different types of real-life datasets.

In addition, we propose another two heuristics to improve the efficiency and utility of our approach. First, if a region is dense enough (the density is calculated based on noisy counts), then there is no need to further partition it because we can reconstruct its noisy version with high accuracy. In practice, we consider a region R with $den(R) \geq 80\%$ to be dense (experiments show that there is no significant utility difference among the density thresholds in the range $[75\%, 90\%]$). Second, we can stop partitioning a region R if the number of elements with value 1 in R is small enough. Note that the determination of a sparse region is based on its noisy count, *not* its density. Specifically, we set the threshold to be $80\% \times \frac{|V|^2}{4^h}$, the number of 1's needed to form *at least* one dense region. Any region with number of 1's $< 80\% \times \frac{|V|^2}{4^h}$ is not worth further partitioning as it will only lead to excessive noise.

Partitioning. For a non-leaf region R, we employ exponential mechanism to find the splitting point that partitions R into subregions with the maximal density difference among all possible splitting points (Line 10). Intuitively, such a split best distinguishes the dense and sparse subregions. For a non-leaf region R of size $m \times l$, there could be *at most* $(m-1)(l-1)$ possible splitting points. We denote the set of all possible splitting points by $\mathcal{P}$. The utility function of selecting a splitting point $p \in \mathcal{P}$ over a region R is designed to be

$$q(R, p) = \max_{\forall R' \in \mathcal{R}} (den(R')) - \min_{\forall R' \in \mathcal{R}} (den(R')),$$

where $\mathcal{R}$ is the set of subregions of R resulted by p.

In order to obtain a reasonably low sensitivity, we constrain the minimum size

of a region in level i (except the root and leaves) of $\mathcal{QT}$ to be $\frac{|V|^2}{4^{i+1}}$. This guarantees, depending on the location of the region R, $CS(q) = \frac{2k4^{i+1}}{|V|^2}$ (if R contains elements on the diagonal) or $CS(q) = \frac{k4^{i+1}}{|V|^2}$ (otherwise). Due to this constraint, we can apply exponential mechanism on a smaller set of possible splitting points $\mathcal{P}$ (no need to consider the splitting points on level i resulting in a subregion with size less than $\frac{|V|^2}{4^{i+1}}$), which selects a splitting point p_i on R with the following probability,

$$\frac{exp(\frac{\epsilon_{partition}}{2hCS(q)}q(R,p_i))}{\sum_{p_j\in\mathcal{P}} exp(\frac{\epsilon_{partition}}{2hCS(q)}q(R,p_j))}, \tag{7.5}$$

where $\frac{\epsilon_{partition}}{h}$ is the privacy budget assigned to the exponential mechanism, as explained below.

Example 7.4.1. Consider the adjacency matrix in Figure 7.2. Suppose the height of $\mathcal{QT}$ is 2. The first possible partition operation is illustrated by the boldest lines, resulting in four subregions: $R_1 = A[1,3;1,5]$, $R_2 = A[1,3;6,8]$, $R_3 = A[4,8;1,5]$ and $R_4 = A[4,8;6,8]$. Assume that the noisy counts of R_1 and R_4 indicate that they are sparse and the noisy count of R_2 indicates that it is dense. DER only needs to further partition R_3. After that, the height has been reached and $\mathcal{QT}$ ends with seven leaf nodes. ■

Privacy budget allocation. Generally, the total privacy budget ϵ is divided into two portions: ϵ' and ϵ'', each being used in a step. More specifically, ϵ is divided for three tasks: ϵ_{count} for calculating noisy counts of all (sub)regions, $\epsilon_{partition}$ for selecting splitting points on all internal nodes of $\mathcal{QT}$, and $\epsilon_{reconstruct}$ for reconstructing all leaf regions, where $\epsilon' = \epsilon_{count} + \epsilon_{partition}$ and $\epsilon'' = \epsilon_{reconstruct}$.

The first problem is to determine the values of ϵ_{count}, $\epsilon_{partition}$ and $\epsilon_{reconstruct}$. In general, we assign larger budgets to ϵ_{count} and $\epsilon_{reconstruct}$ because, as shown later in Theorem 7.11, as long as we can obtain relatively accurate noisy counts, we can always find denser (or sparser) subregions, which can be reconstructed with higher

accuracy. Between ϵ_{count} and $\epsilon_{reconstruct}$, more budget is given to ϵ_{count} because a sufficiently dense (or sparse) leaf region can be recovered with reasonable accuracy regardless of the privacy budget (see Theorem 7.10). Since it is difficult to theoretically quantify the values, we experimentally choose proper portions for each of them, complying with the analysis above.

Once ϵ_{count}, $\epsilon_{partition}$ and $\epsilon_{reconstruct}$ are fixed, we employ the following allocation scheme to distribute them to each node of $\mathcal{QT}$ (Line 9). For noisy counts, we employ an *adaptive* privacy budget allocation scheme based on the geometric budget scheme [28]. Initially, we assume that each root-to-leaf path in $\mathcal{QT}$ will be of the same length h (e.g., $\mathcal{QT}$ is perfect) and assign $\frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$ to each node with depth $1 \leq i < h$. Since an input dataset is always non-empty, there is no need to assign any budget to the root level. Hence we add the portion of the root level, $\frac{(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$, to the leaves, that is, a node with depth h receives $\frac{(2^{h/3}+1)(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$. Then, we adaptively adjust privacy budgets during the partitioning process.

Due to the stop conditions, $\mathcal{QT}$ may not be perfect (i.e., a path may stop before it reaches level h), and, therefore, we want to reallocate the privacy budget left to fully make use of the total budget. For a leaf node v whose depth $i < h$, let $\widetilde{c_1}$ be the noisy count obtained by privacy parameter $\epsilon_1 = \frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$. We can calculate another noisy count $\widetilde{c_2}$ of v with $\epsilon_2 = \frac{(2^{h/3}+1)(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1}$. Obviously, $\widetilde{c_2}$ has a better accuracy than $\widetilde{c_1}$ because $Var(\widetilde{c_2}) < Var(\widetilde{c_1})$. We can replace $\widetilde{c_1}$ by $\widetilde{c_2}$ as a more precise estimate of the true count, but this simple strategy essentially wastes the privacy parameter used for generating $\widetilde{c_1}$. Here we propose a strategy that combines both $\widetilde{c_1}$ and $\widetilde{c_2}$ to calculate a more accurate estimate than *both* $\widetilde{c_1}$ and $\widetilde{c_2}$.

Theorem 7.6. *Let* $\widetilde{c} = \frac{\epsilon_1^2}{\epsilon_1^2+(\gamma\epsilon_2)^2}\widetilde{c_1} + \frac{(\gamma\epsilon_2)^2}{\epsilon_1^2+(\gamma\epsilon_2)^2}\widetilde{c_2}$, *where* $\gamma = \frac{\epsilon_2}{\epsilon_1} = 2^{\frac{h-i}{3}} > 1$. *Then* $Var(\widetilde{c}) < Var(\widetilde{c_2}) < Var(\widetilde{c_1})$. ■

Proof. Since $Var(\widetilde{c_1}) = \frac{2}{{\epsilon_1}^2}$ and $Var(\widetilde{c_2}) = \frac{2}{{\epsilon_2}^2}$, we have:

$$\begin{aligned} Var(\widetilde{c}) &= (\frac{\epsilon_1^2}{\epsilon_1^2 + (\gamma\epsilon_2)^2})^2 Var(\widetilde{c_1}) + (\frac{(\gamma\epsilon_2)^2}{\epsilon_1^2 + (\gamma\epsilon_2)^2})^2 Var(\widetilde{c_2}) \\ &= \frac{2}{\frac{(\epsilon_1^2+(\gamma\epsilon_2)^2)^2}{{\epsilon_1}^2+\gamma^4{\epsilon_2}^2}} \end{aligned}$$

Hence, we need to prove that $\frac{(\epsilon_1^2 + (\gamma\epsilon_2)^2)^2}{{\epsilon_1}^2 + \gamma^4{\epsilon_2}^2} > {\epsilon_2}^2 > {\epsilon_1}^2$. This is equivalent to prove that $\frac{\gamma^4{\epsilon_2}^2 + 2\gamma^2{\epsilon_1}^2 + \frac{{\epsilon_1}^4}{{\epsilon_2}^2}}{\gamma^4{\epsilon_2}^2 + {\epsilon_1}^2} > 1$. Since $\gamma > 1$, $2\gamma^2{\epsilon_2}^2 + \frac{{\epsilon_1}^4}{{\epsilon_2}^2} > {\epsilon_1}^2$. Therefore, $Var(\widetilde{c}) < \frac{2}{{\epsilon_2}^2} = Var(\widetilde{c_2})$. Since $\epsilon_1 < \epsilon_2$, we get $Var(\widetilde{c}) < Var(\widetilde{c_2}) < Var(\widetilde{c_1})$. □

For a leaf node v whose depth $i < h - 1$, the portion of privacy budget left from partitioning, $\frac{(2^{h/3}-2^{(i+1)/3})\epsilon_{count}}{2^{(h+1)/3}-1}$, is added to $\epsilon_{reconstruct}$ so that we can make full use of the privacy budget.

For selecting splitting points by exponential mechanism, we use a uniform budget scheme that equally distributes $\frac{\epsilon_{partition}}{h}$ to each internal node in $\mathcal{QT}$. For reconstructing leaf regions, each leaf node in $\mathcal{QT}$ receives $\epsilon_{reconstruct}$ plus the privacy parameter left from partitioning.

Efficient implementation. In order to apply exponential mechanism, for *every* internal node of $\mathcal{QT}$, we need to compute the densities of the four subregions resulted from *every* possible splitting position. A naive implementation takes run-time $O(|V|^4)$ to calculate the densities for all possible splitting points for all nodes on the same level of $\mathcal{QT}$. We propose a data structure, called *count summary matrix* C of size $|V| \times |V|$, which improves the run-time complexity of calculating all densities for a level of $\mathcal{QT}$ from $O(|V|^4)$ to $O(|V|^2)$. $\forall 1 \leq i, j \leq |V|$, $C[i, j]$ gives the number of 1's in the region $A[1, i; 1, j]$, that is, $C[i, j] = \sum_{m=1}^{i} \sum_{l=1}^{j} A_{ij}$. C can be constructed with run-time complexity $O(|V|^2)$ using the following observation:

$$C[i, j] = C[i - 1, j] + C[i, j - 1] - C[i - 1, j - 1] + A_{ij},$$

where $C[i,j] = 0$ when $i < 1$ or $j < 1$. Note that C just needs to be computed *once* for the entire sanitization process. Once C is constructed, the density of any region $A[k,l;m,n]$ can be computed in $O(1)$ by

$$\frac{C[l,n] - C[l,m-1] - C[k-1,n] + C[k-1,m-1]}{(n-m+1)(l-k+1)}.$$

In addition, when the input dataset is extremely large, sampling (i.e., checking the splitting points with a step larger than 1) could be used at the cost of slightly worse utility.

Run-time complexity. The run-time complexity of Algorithm 7.2 is given in Theorem 7.7.

Theorem 7.7. *The run-time complexity of Algorithm 7.2 is* $O(|V|^2)$. ■

Proof. The complexity of Algorithm 7.2 is dominated by the application of exponential mechanism to select the splitting points. Suppose the size of a node v_i in level j of $\mathcal{QT}$ is $m_i \times l_i$. A single application of exponential mechanism needs to consider at most $(m_i - 1)(l_i - 1)$ possible splitting positions. Due to the count summary matrix, each position can be checked in constant time. Since $m_i l_i > (m_i - 1)(l_i - 1)$ is the area of the region represented by v_i, we have $\sum_{v_i \in level(j,\mathcal{QT})} m_i l_i \leq |V|^2$ because the sum of the areas represented by all nodes on level j cannot be greater than the total area $|V|^2$. So the complexity of building level j is $O(|V|^2)$. Therefore, the total complexity of building $\mathcal{QT}$ of height h must be bounded by $O(h|V|^2)$. Since $h \ll |V|^2$, the run-time complexity of Algorithm 7.2 can be further considered as $O(|V|^2)$. □

Finally, in the exploration process, we can conduct a simple post-processing step by rounding the noisy count $\widetilde{c}$ of a region R with size $m \times l$ into the range of $[0, ml]$.

7.4.2 Edge Arrangement

In this section, we denote an original region in A by R and its reconstructed counterpart in $\widetilde{A}$ by $\widetilde{R}$. Since our utility requirement is to build a differentially private $\widetilde{A}$ such that $\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}|A_{ij}-\widetilde{A}_{ij}|$ is minimized, it naturally requires to reconstruct each leaf region $\widetilde{R}$ of size $m\times l$ with $\sum_{i=1}^{m}\sum_{j=1}^{l}|R_{ij}-\widetilde{R}_{ij}|$ minimized.

Given a leaf region $\widetilde{R}$ of size $m\times l$ with a noisy count $\widetilde{c}\leq ml$, we design an exponential mechanism to select an edge arrangement r by the following utility function:

$$q(\widetilde{R},r)=ml-\sum_{i=1}^{m}\sum_{j=1}^{l}|R_{ij}-\widetilde{R}_{ij}| \tag{7.6}$$

Intuitively, the utility function measures how many elements of $\widetilde{R}$ are correctly assigned with respect to R. The correlated sensitivity of $q(R,r)$ is $CS(q)=\min\{2k,ml\}$ or $CS(q)=\min\{k,ml\}$, depending on the location of R in A.

A naive implementation of exponential mechanism needs to explicitly consider a total of $\binom{ml}{\widetilde{c}}$ possible arrangements, which is of *factorial complexity*. Instead, we propose an efficient implementation, which takes run-time complexity of only $O(ml)$ for assigning edges in a single leaf region. We first implicitly group all arrangements with the same score into a group. At first glance, for any region with size $m\times l$, there can be *at most* $ml+1$ groups because there are at most $ml+1$ possible score values (from 0 to ml as defined in Equation 10). Now we show that the actual number of groups to consider is $\leq\left\lceil\frac{ml+1}{2}\right\rceil$ by giving the sufficient and necessary condition of a possible score.

Theorem 7.8. *For a leaf region $\widetilde{R}$ of size $m\times l$ with a noisy count $\widetilde{c}$ and the true count c, a score s is possible if and only if $s\in[\max\{\widetilde{c}+c-ml,ml-c-\widetilde{c}\},\min\{ml+c-\widetilde{c},ml+\widetilde{c}-c\}]$ and $\frac{s+c+\widetilde{c}-ml}{2}$ is an integer. The total number of possible scores is*

$$\leq\left\lceil\frac{\min\{ml+c-\widetilde{c},ml+\widetilde{c}-c\}-\max\{\widetilde{c}+c-ml,ml-c-\widetilde{c}\}}{2}\right\rceil\leq\frac{ml+1}{2}.\ \blacksquare$$

Proof. We first calculate the lower and upper bounds of a possible score by considering all possible cases. 1) $\widetilde{c} \geq c$ and $\widetilde{c} \leq ml - c$: the maximum score is achieved when c 1's are assigned to the elements where $R_{ij} = 1$ and the rest $\widetilde{c} - c$ 1's are assigned to the elements with $R_{ij} = 0$, which gives the score $ml + c - \widetilde{c}$; the minimal score is achieved when $\widetilde{c}$ 1's are assigned to the elements with $R_{ij} = 0$, which gives the score $ml - c - \widetilde{c}$. 2) $\widetilde{c} \geq c$ and $\widetilde{c} > ml - c$: similarly, the maximum score is $ml + c - \widetilde{c}$ while the minimum is $\widetilde{c} + c - ml$. 3) $\widetilde{c} < c$ and $\widetilde{c} \leq ml - c$: the maximum is $ml + \widetilde{c} - c$ while the minimum is $ml - c - \widetilde{c}$. 4) $\widetilde{c} < c$ and $\widetilde{c} > ml - c$: the maximum is $ml + \widetilde{c} - c$ and the minimum is $\widetilde{c} - (ml - c)$. Combining these four cases, we get the bounds of s.

Next, we prove that $\frac{s+c+\widetilde{c}-ml}{2}$ must be an integer in order to make s possible. Consider the allocation of $\widetilde{c}$ 1's in $\widetilde{R}$. Suppose the numbers of elements where $R_{ij} = 0 \wedge \widetilde{R}_{ij} = 0$, $R_{ij} = 0 \wedge \widetilde{R}_{ij} = 1$, $R_{ij} = 1 \wedge \widetilde{R}_{ij} = 0$, $R_{ij} = 1 \wedge \widetilde{R}_{ij} = 1$, are respectively x, y, z and w. For an arrangement with a score s, we have:

$$\begin{aligned} x + y + z + w &= ml \\ x + w &= s \\ y + w &= \widetilde{c} \\ z + w &= c \end{aligned}$$

Solving these equations, we get $w = \frac{s+c+\widetilde{c}-ml}{2}$. Apparently, only if x, y, z and w are non-negative integers, s is possible. Since $s \in [\max\{\widetilde{c} + c - ml, ml - c - \widetilde{c}\}, \min\{ml + c - \widetilde{c}, ml + \widetilde{c} - c\}]$, x, y, z and w must be non-negative. So we just need to require $\frac{s+c+\widetilde{c}-ml}{2}$ to be an integer, which consequently guarantees that x, y, z are also integers. Finally, since $\frac{s+c+\widetilde{c}-ml}{2}$ has to be an integer, all possible scores have to be either all even or all odd. We complete the proof. □

We call a group of arrangements with a possible score a *valid* group. We can calculate the size of each valid group by Theorem 7.9.

Theorem 7.9. *Given a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c}$ and the true count c, the size of a valid group G_s with score s is*

$$|G_s| = \binom{c}{\frac{s+c+\widetilde{c}-ml}{2}} \binom{ml-c}{\frac{ml+\widetilde{c}-s-c}{2}}, \tag{7.7}$$

where $\binom{0}{0}$ is defined to be 1. ■

Proof. Following the proof of Theorem 7.8, we have $w = \frac{s+c+\widetilde{c}-ml}{2}$, which means that we need to assign $\frac{s+c+\widetilde{c}-ml}{2}$ 1's to the elements where $R_{ij} = 1$ and $\widetilde{c} - \frac{s+c+\widetilde{c}-ml}{2}$ 1's to the elements where $R_{ij} = 0$. For the former case, there are a total of $\binom{c}{\frac{s+c+\widetilde{c}-ml}{2}}$ possible combinations; for the latter case, there are a total of $\binom{ml-c}{\frac{ml+\widetilde{c}-s-c}{2}}$ possible combinations. Therefore, $|G_s| = \binom{c}{\frac{s+c+\widetilde{c}-ml}{2}}\binom{ml-c}{\frac{ml+\widetilde{c}-s-c}{2}}$. □

Then exponential mechanism can be used to select a group G_i with the following probability,

$$\frac{exp(\frac{i\bar{\epsilon}}{2CS(q)}) \times |G_i|}{\sum_{j=0}^{ml}(exp(\frac{j\bar{\epsilon}}{2CS(q)}) \times |G_j|)}, \tag{7.8}$$

where $\bar{\epsilon}$ equals $\epsilon_{reconstruct}$ plus the privacy budget left from the exploration process, $CS(q) = \min\{2k, ml\}$ or $CS(q) = \min\{k, ml\}$, and the size of an *invalid* group is 0.

Finally, conditional on that the group G_i is selected, we can uniformly generate a random arrangement within G_i by randomly assigning $\frac{i+c+\widetilde{c}-ml}{2}$ 1's to the elements R_{ij} with $R_{ij} = 1$ and $\frac{ml+\widetilde{c}-i-c}{2}$ 1's to the elements R_{ij} with $R_{ij} = 0$. Obviously, generating such an arrangement could be done with run-time complexity $O(1)$. In particular, if a generated arrangement makes $\widetilde{A}_{ii} = 1$ for any $1 \leq i \leq |V|$, an alternative arrangement could be generated because a graphic matrix contains a zero diagonal.

We give the utility guarantee of our edge arrangement method below.

Theorem 7.10. *Given a leaf region $\widetilde{R}$ of size $m \times l$ with a noisy count $\widetilde{c}$ and the*

true count c, with probability $1-\beta$,

$$\begin{aligned}\forall \widetilde{c} < c,\ q(\widetilde{R}, r^*) \ &\geq\ \max\{\widetilde{c} + c - ml, ml - c - \widetilde{c},\\ &\quad ml - c + \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \binom{ml}{\widetilde{c}} - \log \binom{c}{\widetilde{c}} - ln\beta)\}\end{aligned}$$

and

$$\begin{aligned}\forall \widetilde{c} \geq c,\ q(\widetilde{R}, r^*) \ &\geq\ \max\{\widetilde{c} + c - ml, ml - c - \widetilde{c},\\ &\quad ml + c - \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \binom{ml}{c} - \log \binom{\widetilde{c}}{c} - ln\beta)\}\end{aligned}$$

where r^* *is the arrangement selected by our approach.* ■

Proof. Let us define $OPT_q(\widetilde{R}) = \max_{r\in\mathcal{R}} q(\widetilde{R}, r)$, $\mathcal{R}_{OPT} = \{r \in \mathcal{R} : q(\widetilde{R}, r) = OPT_q(\widetilde{R})\}$ and $r^* = Exponential(\widetilde{R}, \mathcal{R}, q, \bar{\epsilon})$. In [89], [53], it has been proven that

$$Pr[q(\widetilde{R}, r^*) \leq OPT_q(\widetilde{R}) - \frac{2GS(q)}{\bar{\epsilon}}(\log \frac{|\mathcal{R}|}{|\mathcal{R}_{OPT}|} + t)] \leq e^{-t} \tag{7.9}$$

When $\widetilde{c} < c$, $OPT_q(\widetilde{R})$ is achieved when all $\widetilde{c}$ 1's are assigned to the elements with $R_{ij} = 1$, and $OPT_q(\widetilde{R}) = ml - c + \widetilde{c}$. The total number of possible arrangements is $\binom{ml}{\widetilde{c}}$ and the number of arrangements achieving $OPT_q(\widetilde{R})$ is $\binom{c}{\widetilde{c}}$. Therefore, setting $t = ln(1/\beta)$, we obtain

$$q(\widetilde{R}, r^*) \geq ml - c + \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \binom{ml}{\widetilde{c}} - \log \binom{c}{\widetilde{c}} - ln\beta).$$

Combining the lower bound of a score given in Theorem 7.8, we get the lower bound of $q(\widetilde{R}, r^*)$.

When $\widetilde{c} \geq c$, $OPT_q(\widetilde{R})$ is achieved when all the elements with $R_{ij} = 1$ are assigned 1's and the rest $\widetilde{c} - c$ 1's are assigned to the elements with $R_{ij} = 0$. We get $OPT_q(\widetilde{R}) = c + (ml - c) - (\widetilde{c} - c) = ml + c - \widetilde{c}$. The number of arrangements

achieving $OPT_q(\widetilde{R})$ is $\binom{ml-c}{\widetilde{c}-c}$. Hence we have, with probability $1-\beta$,

$$\begin{aligned} q(\widetilde{R}, r^*) &\geq ml + c - \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \frac{\binom{ml}{\widetilde{c}}}{\binom{ml-c}{\widetilde{c}-c}} - ln\beta) \\ &= ml + c - \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \frac{\binom{ml}{\widetilde{c}}}{\frac{\binom{ml}{\widetilde{c}}\binom{\widetilde{c}}{c}}{\binom{ml}{c}}} - ln\beta) \\ &= ml + c - \widetilde{c} - \frac{2CS(q)}{\bar{\epsilon}}(\log \binom{ml}{c} - \log \binom{\widetilde{c}}{c} - ln\beta) \end{aligned}$$

Similarly, the lower bound in Theorem 7.8 also applies. This completes the proof. □

Specifically, when $\widetilde{c} = c$, we have

$$q(\widetilde{R}, r^*) \geq \max\{2c - ml, ml - 2c, ml - \frac{2CS(q)}{\bar{\epsilon}}(\log \binom{ml}{c} - ln\beta)\}.$$

We can observe that when c is either relatively large or relatively small with respect to ml (that is, either $den(R)$ is large enough or small enough), the reconstructed $\widetilde{R}$ could be very close to R. The worst utility occurs when $c = \frac{ml}{2}$. However, this case can always be avoided by further partitioning.

Theorem 7.11. *Given a region R, any partitioning of R results in sub-regions R' satisfying either $den(R') \leq den(R)$ or $den(R') \geq den(R)$, with equality attained if and only if 1's are uniformly distributed in R.* ■

The proof is obvious and is therefore omitted here. According to the *power law distribution* [41], R is very unlikely to have a uniform distribution. Therefore, Theorem 7.11 suggests that keeping partitioning a region leads to a more precise reconstruction. This observation is based on the assumption that $\widetilde{c}$ is accurate. However, when subregions become smaller, the accuracy of $\widetilde{c}$ decreases, which causes extra utility lost in edge assignment. Therefore, it justifies our design of the stop

condition that takes into consideration both the accuracy of noisy counts and the size of a leaf region.

After reconstructing each leaf region, we perform an extra step to make $\widetilde{A}$ *graphic*: $\forall 1 \leq i, j \leq |V|$, if $\widetilde{A}_{ij} \neq \widetilde{A}_{ji}$, set both $\widetilde{A}_{ij}$ and $\widetilde{A}_{ji}$ to either 0 or 1 with probability 50%.

We can see that the complexity of reconstructing all leaf regions is $O(|V|^2)$ because $\sum_i \frac{m_i l_i + 1}{2} < |V|^2$. Therefore, the total complexity of DER is $O(|V|^2)$.

7.4.3 Privacy Analysis

In this section, we prove that Algorithm 7.1 satisfies (ϵ, k)-differential privacy.

Theorem 7.12. *DER is (ϵ, k)-differentially private.* ■

Proof. Recall that the given total privacy budget ϵ is divided into three portions: ϵ_{count} for calculating noisy counts of all regions, $\epsilon_{partition}$ for selecting the splitting points and $\epsilon_{reconstruct}$ for reconstructing leaf regions.

We first show that the *sequential composition* property [87] and the *parallel composition* property [87] also apply to (ϵ, k)-differential privacy.

Theorem 7.13. *Let $\mathcal{A}_i$ each provide (ϵ_i, k_i)-differential privacy. A sequence of $\mathcal{A}_i(D)$ over the database D provides $(\sum_i \epsilon_i, \min(k_i))$-differential privacy.* ■

Theorem 7.14. *Let $\mathcal{A}_i$ each provide (ϵ_i, k_i)-differential privacy. A sequence of $\mathcal{A}_i(D_i)$ over a set of disjoint databases D_i provides $(\max(\epsilon_i), \min(k_i))$-differential privacy.* ■

The proof of Theorem 7.13 and Theorem 7.14 is analogous to the proof under ϵ-differential privacy in [87]. Specifically, by definition, any sub-database of a k-correlated database can be *at most* k-correlated. According to Theorem 7.2, an (ϵ, k)-differentially private mechanism is able to bound an adversary's probability

Table 7.1: Experimental dataset statistics

Datasets	$\|V\|$	$\|E\|$	Edge density
ca-GrQc	5,242	14,496	0.00106
ca-HepTh	5,000	17,138	0.00137
wiki-Vote	7,115	100,762	0.00398
STM	1,012	7,860	0.01536

change over all sub-databases by e^{ϵ}. Actually, if the correlation parameter of a sub-database can be precisely calculated, then less noise could be added.

Because of the parallel composition property, the privacy budget used in each root-to-leaf path of $\mathcal{QT}$ is independent of each other, while the privacy budget within a path follows the sequential composition property. Under our adaptive privacy budget allocation scheme, the privacy budget used in a single path is *at most*

$$\sum_{i=0}^{h} \frac{2^{i/3}(\sqrt[3]{2}-1)\epsilon_{count}}{2^{(h+1)/3}-1} + \sum_{i=0}^{h-1} \frac{\epsilon_{partition}}{h} + \epsilon_{reconstruct} = \epsilon.$$

Since the correlated sensitivity is used, our approach satisfies (ϵ, k)-differential privacy and bounds an adversary's probability change by e^{ϵ} on any k-correlation bounded input dataset. □

7.5 Experimental Evaluation

In this section, we experimentally evaluate the performance of our sanitization algorithm (DER). As a reference point, we compare the utility of DER with a random graph of the same numbers of nodes and edges [60], [26] (referred to as *Random*) and a sanitized graph generated by a simple Laplace mechanism based approach proposed in [54] (referred to as *Laplace*). In all figures, the results reported are the average of 10 runs. Our implementation was done in C++, and all experiments were performed on an Intel Core 2 Duo 2.80GHz PC with 8GB RAM.

Four real-life datasets from three different types of networks are used in our

Table 7.2: Average relative error of large query size

Datasets	$0.2 \cdot \|V\|$	$0.4 \cdot \|V\|$	$0.6 \cdot \|V\|$	$0.8 \cdot \|V\|$	$\|V\|$
ca-GrQc	0.040	0.033	0.036	0.050	0.047
ca-HepTh	0.035	0.037	0.036	0.045	0.042
wiki-Vote	0.041	0.046	0.045	0.042	0.047
STM	0.076	0.043	0.024	0.014	0.009

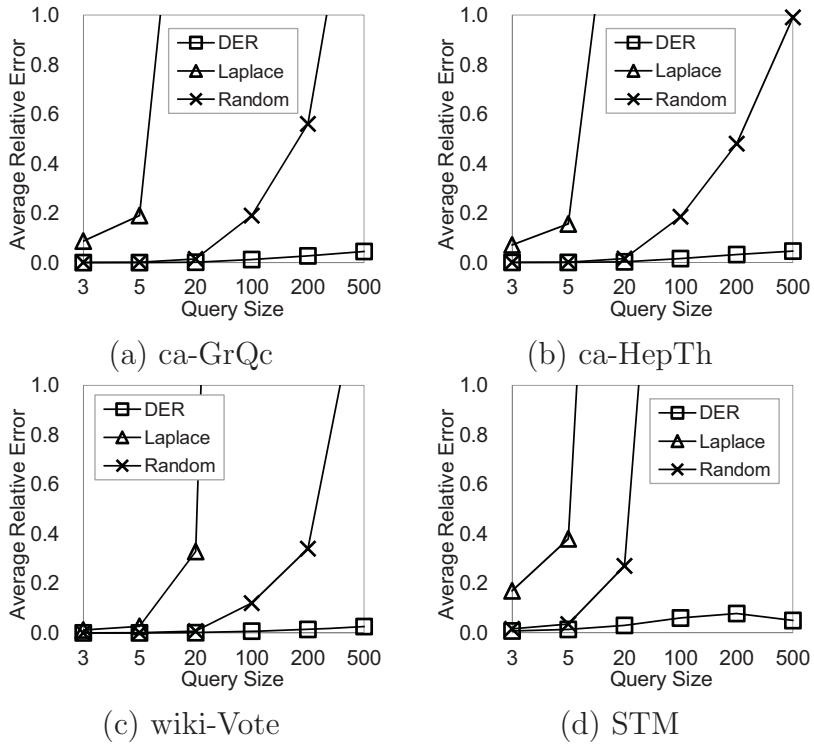
experiments. [3] *ca-GrQc* is a subset of the collaboration network of Arxiv general relativity category. Two authors are connected if they coauthored at least one paper. *ca-HepTh* is extracted from the collaboration network of Arxiv high energy physics theory category. Similarly, there is an edge if two authors coauthored at least one paper. The *wiki-Vote* dataset contains social network information about Wikipedia voting on promotion to administratorship. An edge is created between two persons if one voted on or was voted by the other. *STM* provides the transportation network information of the Montreal transportation system. Two stations are considered connected if there are more than 500 passengers commuting between them within one week. The detailed characteristics of the datasets are summarized in Table 7.1.

7.5.1 Data Utility

Cut query. In the first set of experiments, we examine the utility for cut queries in terms of average relative error. We examine all possible query sizes (i.e., the number of vertices in a cut query), and report the results of 11 representative query sets with sizes spanning over the full spectrum in Figure 7.3 and Table 7.2. Each query set consists of queries with sizes that are uniformly randomly distributed between 1 and the specified maximal size. For each query set, we randomly generate 20,000 queries. The sanity bound s is set to 0.1% of $|E|$, the same as [125], [25].

Figure 7.3 presents the average relative errors of cut queries of relatively small

[3] *ca-GrQc*, *ca-HepTh* and *wiki-Vote* are publicly available in the Stanford large network dataset collection (http://snap.stanford.edu/data/index.html). *STM* is provided by the Société de transport de Montréal (http://www.stm.info).



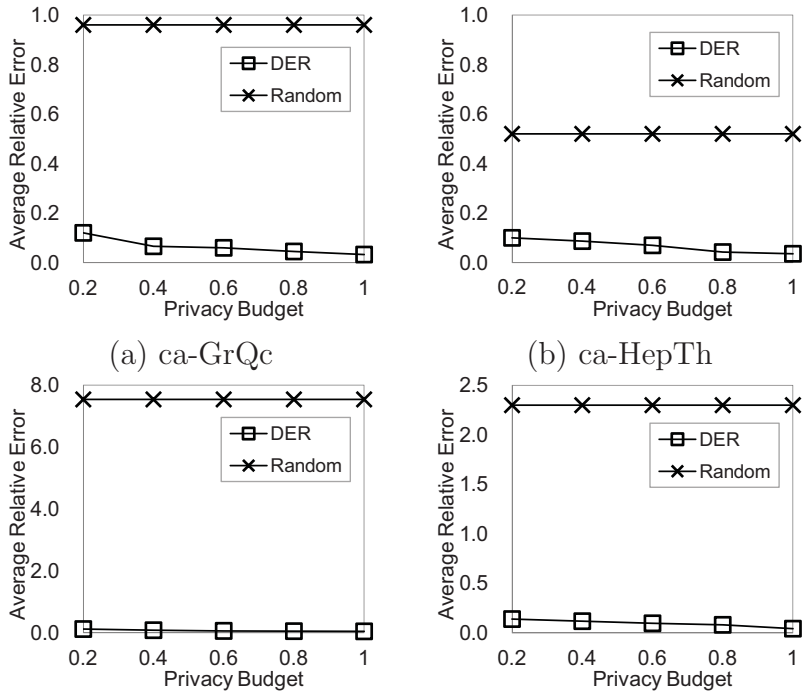
(a) ca-GrQc (b) ca-HepTh

(c) wiki-Vote (d) STM

Figure 7.3: Average relative error vs. query size.

query sizes while fixing $\epsilon = 1.0$ and $k = 1$. Six query sets (with maximal query sizes 3, 5, 20, 100, 200 and 500, respectively) are used to represent the general trends of the three approaches. As one can observe, the average relative errors of *DER* are consistently small under all query sizes. It is worth mentioning that the relative errors of *DER* do not monotonically increase with the increase of query sizes. It is surprising to see that *Laplace* performs much worse than *Random*. This is because Laplace noise generated under a small privacy budget can easily make the original value of an element (either 0 or 1) indistinguishable. With the increase of query sizes, both *Laplace* and *Random* provide very poor utility. Another interesting observation is that the utility of *Random* is subtly related to the edge density: its performance deteriorates quickly with the increase of edge density.

Table 7.2 inspects the performance of DER under large query sizes with $\epsilon = 1.0$ and $k = 1$, where the query sets have the maximal sizes $0.2 \cdot |V|$, $0.4 \cdot |V|$, $0.6 \cdot |V|$,



(a) ca-GrQc (b) ca-HepTh

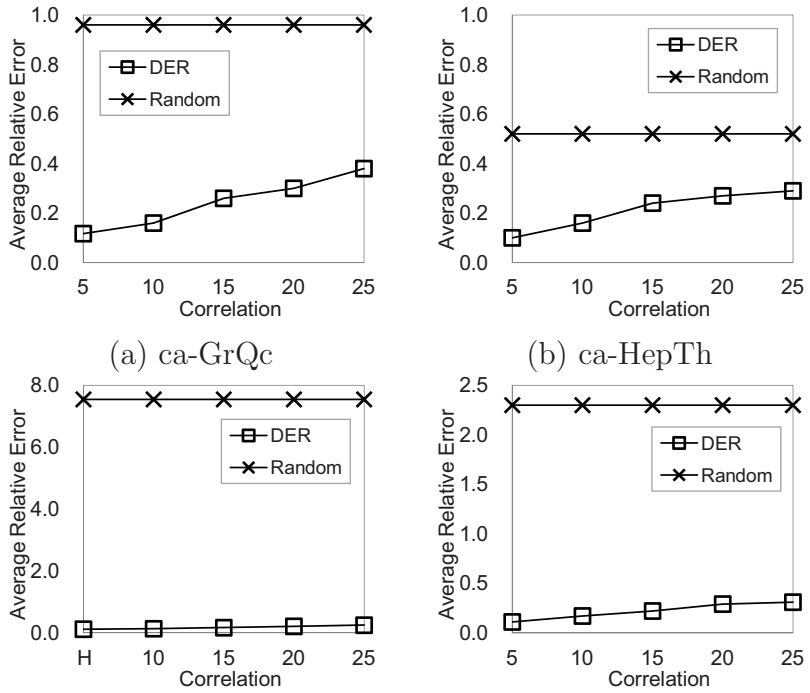
(c) wiki-Vote (d) STM

Figure 7.4: Average relative error vs. ϵ.

$0.8 \cdot |V|$ and $|V|$, respectively. We can observe that *DER* also performs stably well under all large query sizes.

In Figure 7.4, we present relative errors of *DER* and *Random* under varying privacy budgets from 0.2 to 1.0 while fixing the maximal query size to be $0.4 \cdot |V|$ and $k = 1$ (*Laplace*'s relative errors are too large to fit into the figures). As expected, the relative error increases when the privacy budget decreases. Nevertheless, *DER* achieves relative errors less than 14% on all datasets even when $\epsilon = 0.2$.

We finally study how average relative errors vary under different correlation parameters while fixing $\epsilon = 1.0$ and the maximal query size to $0.4 \cdot |V|$ in Figure 7.5. In general, the relative error increases with the increment of k because larger noise has to be injected to hide stronger correlation. Roughly, increasing k is equivalent to decreasing ϵ (as the magnitude of noise is determined by $\frac{e}{k}$). However, on a dataset with a small size, k has a weaker influence than ϵ because for a specific region the



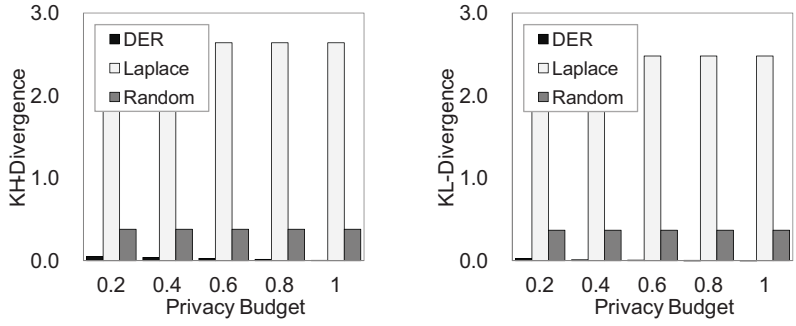
(a) ca-GrQc (b) ca-HepTh

(c) wiki-Vote (d) STM

Figure 7.5: Average relative error vs. k.

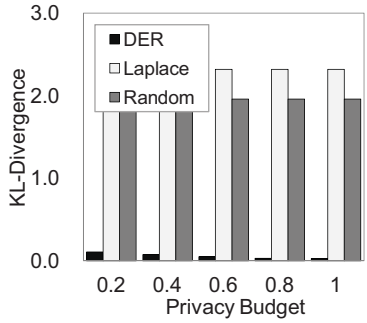
actual noise to add is also bounded by the region's size. This is confirmed by relative errors of STM. When $k = 5$ and $\epsilon = 1.0$, the error is 10% while when $k = 1$ and $\epsilon = 0.2$, the error is 13%. Moreover, we can observe that *DER* can still provide some useful information even when k is relatively large. In practice, many types of networks (e.g., transportation networks) have relatively small correlation, and, thus, our approach can provide meaningful data utility without sacrificing privacy.

Degree distribution. In the second set of experiments, we demonstrate the utility of sanitized data in terms of degree distribution, measured by KL-divergence. Figure 7.6 presents the KL-divergences for all datasets under different privacy budgets with $k = 1$. We can observe that our approach is extremely suitable for preserving degree distributions. Similarly, *Laplace* can barely provide any useful information in terms of degree distribution because its KL-divergence is almost the same as an

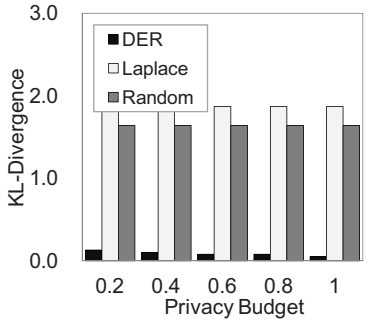


(a) ca-GrQc

(b) ca-HepTh



(c) wiki-Vote



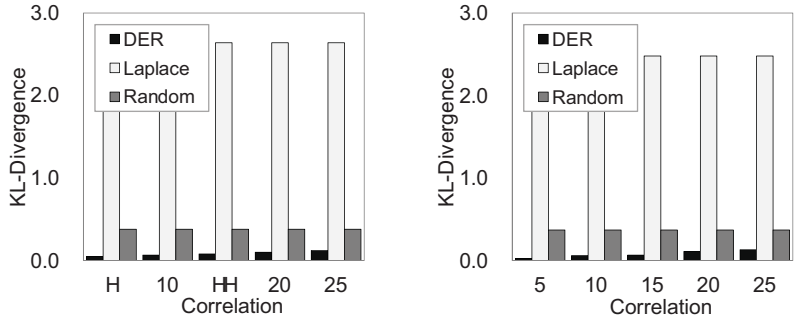
(d) STM

Figure 7.6: Degree distribution vs. ϵ.

empty graph (i.e., $|E| = 0$). Figure 7.7 examines the KL-divergence for varying correlation parameters with $\epsilon = 1.0$. It demonstrates that *DER* performs very stable with increasing correlation. Even when $k = 25$, it is still able to preserve the general degree distributions of all datasets.

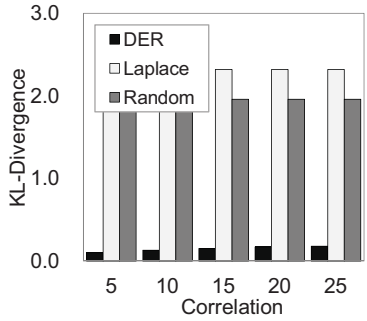
7.5.2 Efficiency

According to the complexity analysis of *DER*, its run-time is dominated by $|V|$. Therefore, we present the run-time of *DER* under different $|V|$ values in Figure 7.8. The test sets are generated by randomly extracting a subset from the original datasets. The X-axis represents the percentage of the test sets' $|V|$ values with respect to the original datasets. It can be observed that roughly the run-time grows quadratically with $|V|$, which confirms our theoretical analysis. Since our approach

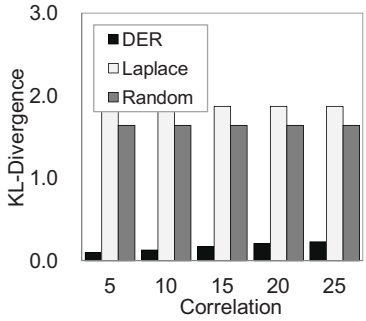


(a) ca-GrQc

(b) ca-HepTh



(c) wiki-Vote



(d) STM

Figure 7.7: Degree distribution vs. k.

is used in the *non-interactive* setting, it meets the scalability requirement of most real-life applications. In the few extreme cases, we note that *DER* can substantially speed up by sampling (see Section 7.4.1) at the cost of slight utility degradation (using the step of 10 on *ca-GrQc*, *ca-HepTh* and *wiki-Vote* makes *DER* roughly 8 times faster with 10% utility deterioration).

7.6 Summary

In this chapter, we propose the (ϵ, k)-differential privacy model, a stronger variant of ϵ-differential privacy, which provides provable privacy guarantees over correlated databases. Based on this privacy model, we present an efficient non-interactive approach for publishing network data. This is the first work that gives a practical solution for network data publication in the spirit of differential privacy. Extensive

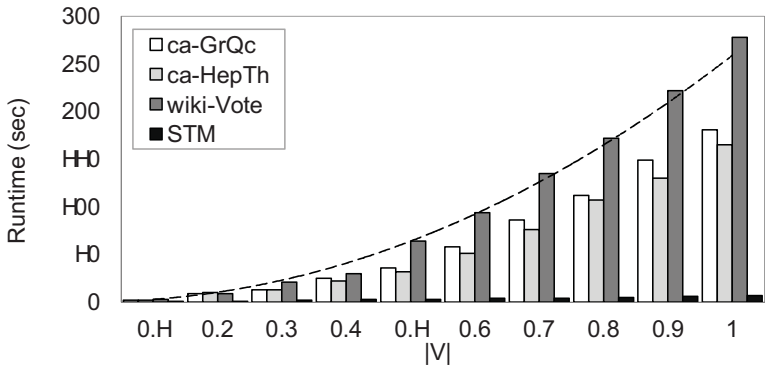


Figure 7.8: Run-time vs. $|V|$.

experiments demonstrate that our solution performs very well on different types of real-life network datasets.

Chapter 8

Conclusions

With the general trend of digitalization, information sharing has become part of the routine activity of many individuals, companies, organizations, and government agencies. Privacy-preserving data publishing techniques have been playing an increasingly important role in privacy protection in information sharing. Early research of PPDP focuses on protecting private and sensitive information in relational and statistical data. However, with the deployment of privacy-threatening technology, such as smart card automated fare collection systems and social networks, the privacy concerns in sharing high-dimensional data, including set-valued data, trajectory data, sequential data and network data, have been substantially raised. In this thesis, we response to these privacy concerns by developing efficient and effective non-interactive data publishing solutions for various utility requirements. We recap the major contributions of this thesis as follows:

- In Chapter 4, we presented the first study of set-valued data publication in the framework of differential privacy. We proposed a probabilistic partitioning-based algorithm that provides guaranteed utility. Our work also contributes to the research of differential privacy by initiating the line of *data-dependent non-interactive* solutions, which are more effective and efficient than existing data-independent solutions.

- In Chapter 5, we proposed the $(K, C)_L$-privacy model to acknowledge the emergence of heterogeneous trajectory data publishing scenarios. We developed a generic anonymization framework, which for the first time introduces local suppression to trajectory data anonymization. This framework also distinguishes itself by accommodating diverse data utility metrics and therefore supporting various data analysis tasks.

- In Chapter 6, motivated by the real-life demand of the STM, we developed two alternative solutions for sequential data publication. These solutions are the pioneers in the use of differential privacy for publishing sequential data. The first solution makes use of a hybrid-granularity prefix tree structure and performs constrained inferences to boost utility, while the second solution introduces the use of a variable-length n-gram model for achieving differential privacy, along with a set of novel noise reduction techniques. Both of them have exhibited desirable performances on different real-life datasets.

- In Chapter 7, we proposed a variant of differential privacy, known as (ϵ, k)-differential privacy, to address its deficiency over correlated data. Based on this stronger privacy notion, we provided a holistic solution for publishing large-volume real-life network data that is useful for cut queries and degree distribution. To the best of our knowledge, this is the first practical solution for publishing network data in the spirit of differential privacy.

In conclusion, as a preliminary effort toward privacy in high-dimensional data publishing, this thesis has reported encouraging results, which demonstrate great promise for releasing useful high-dimensional data while preserving individual privacy.

Bibliography

[1] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proc. of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 376–385, 2008.

[2] N. R. Adam and J. C. Wortmann. Security control methods for statistical databases. *ACM Computing Surveys*, 21(4):515–556, 1989.

[3] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 901–909, 2005.

[4] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *Proc. of the 9th International Conference on Extending Database Technology (EDBT)*, pages 183–199, 2004.

[5] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 247–255, 2001.

[6] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th International Conference on Data Engineering (ICDE)*, pages 3–14, 1995.

[7] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography.

In *Proc. of the 16th International Conference on World Wide Web (WWW)*, pages 181–190, 2007.

[8] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proc. of the 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 273–282, 2007.

[9] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 217–228, 2005.

[10] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[11] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 503–512, 2010.

[12] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proc. of the 40th ACM Symposium on Theory of Computing (STOC)*, pages 609–618, 2008.

[13] R. Brand. Microdata protection through noise addition. *Inference Control in Statistical Databases*, pages 97–116, 2002.

[14] A. Bruno, M. Catherine, and T. Martin. Mining public transport user behaviour from smart card data. In *Proc. of 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, 2006.

[15] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: a maximal frequent itemset algorithm for transactional databases. In *Proc. of the 17th IEEE International Conference on Data Engineering (ICDE)*, pages 443–452, 2001.

[16] L. Burnett, K. Barlow-Stewart, A. Pros, and H. Aizenberg. The gene trustee: A universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10(4):506–513, 2003.

[17] J. Cao, P. Karras, C. Raissi, and K.-L. Tan. ρ-uncertainty inference proof transaction anonymization. *Proceedings of the VLDB Endowment*, 3(1):1033–1044, 2010.

[18] M. Catherine, T. Martin, and A. Bruno. Measuring transit performance using smart card data. In *Proc. of the 11th World Conference on Transportation Research*, 2007.

[19] B.-C. Chen, K. LeFevre, and R. Ramakrishnan. Privacy skyline: privacy with multidimensional adversarial knowledge. In *Proc. of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 770–781, 2007.

[20] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proc. of the 19th ACM Conference on Computer and Communications Security (CCS)*, 2012.

[21] R. Chen, B. C. M. Fung, and B. C. Desai. Differential private trajectory data publication. *CoRR*, 2011.

[22] R. Chen, B. C. M. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: a case study on the montreal transportation system. In *Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2012.

[23] R. Chen, B. C. M. Fung, N. Mohammed, B. C. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences: Special Issue on Data Mining for Information Security*, in press.

[24] R. Chen, B. C. M. Fung, P. S. Yu, and B. C. Desai. Correlated network data publication via differential privacy. *Very Large Data Bases Journal (VLDBJ)*, under submission.

[25] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

[26] J. Cheng, A. W.-C. Fu, and J. Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *Proc. of the 36th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 459–470, 2010.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, third edition, 2009.

[28] G. Cormode, M. Procopiuc, E. Shen, D. Srivastava, and T. Yu. Differentially private spatial decompositions. In *Proc. of the 27th IEEE International Conference on Data Engineering (ICDE)*, pages 20–31, 2012.

[29] G. Cormode, M. Procopiuc, D. Srivastava, and T. Tran. Differentially private summaries for sparse data. In *Proc. of the 15th International Conference on Database Theory (ICDT)*, 2012.

[30] G. Cormode, D. Srivastava, S. Bhagat, and B. Krishnamurthy. Class-based graph anonymization for social network data. *Proceedings of the VLDB Endowment*, 2(1):766–777, 2009.

[31] L. H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75(370):377–385, 1980.

[32] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15:429–444, 1977.

[33] T. Dalenius. Finding a needle in a haystack or identifying anonymous census record. *Journal of Official Statistics*, 2(3):329–336, 1986.

[34] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 202–210, 2003.

[35] W. Du, Z. Teng, and Z. Zhu. Privacy-maxent: integrating background knowledge in privacy quantification. In *Proc. of the 34th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 459–472, 2008.

[36] C. Dwork. Differential privacy. In *Proc. of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12, 2006.

[37] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of the 3rd Theory of Cryptography Conference (TCC)*, pages 265–284, 2006.

[38] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *Proc. of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 85–94, 2007.

[39] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proc. of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 381–390, 2009.

[40] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[41] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. of the 23rd ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 251–262, 1999.

[42] R. A. Finkel and J. L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[43] A. Friedman and A. Schuster. Data ming with differential privacy. In *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–502, 2010.

[44] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, 9(2):383–406, 1993.

[45] B. C. M. Fung, M. Cao, B. C. Desai, and H. Xu. Privacy protection for RFID data. In *Proc. of the 24th ACM SIGAPP Symposium on Applied Computing (SAC)*, pages 1528–1535, 2009.

[46] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: a survey of recent developments. *ACM Computing Surveys*, 42(4):14:1–14:53, 2010.

[47] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE)*, pages 205–216, 2005.

[48] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(5):711–725, 2007.

[49] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 265–273, 2008.

[50] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *Proc. of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 715–724, 2008.

[51] A. Ghosh, T. Roughgarden, and M. Sunararajan. Universally utility-maximizing privacy mechanisms. In *Proc. of the 41st ACM Symposium on Theory of Computing (STOC)*, pages 351–360, 2009.

[52] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Trajectory pattern mining. In *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 330–339, 2007.

[53] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar. Differentially private combinatorial optimization. In *Proc. of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1106–1125, 2010.

[54] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *Proc. of the 9th Theory of Cryptography Conference (TCC)*, pages 339–356, 2012.

[55] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan Kaufmann, third edition, 2011.

[56] S. Hansell. AOL removes search data on vast group of web users. *New York Times*, August 8, 2006.

[57] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, 2012.

[58] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *Proc. of the 9th IEEE International Conference on Data Mining (ICDM)*, pages 169–178, 2009.

[59] M. Hay, K. Liu, G. Miklau, J. Pei, and E. Terzi. Privacy-aware data management in information networks. In *Proc. of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1201–1204, 2011.

[60] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, 2008.

[61] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1):1021–1032, 2010.

[62] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *Proceedings of the VLDB Endowment*, 2(1):934–945, 2009.

[63] H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng. Privacy-aware location data publishing. *ACM Transactions on Database Systems (TODS)*, 35(3):17, 2010.

[64] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 279–288, 2002.

[65] I. Kamel and C. Faloutsos. Hilbert r-tree: an improved r-tree using fractals. In *Proc. of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 500–509, 1994.

[66] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[67] D. Kifer. Attacks on privacy and deFinetti's theorem. In *Proc. of the 35th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 127–138, 2009.

[68] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *Proc. of the 32nd ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 217–228, 2006.

[69] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proc. of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 193–204, 2011.

[70] J. Kim and W. Winkler. Masking microdata files. In *Proc. of the Survey Research Methods Section, American Statistical Association*, pages 114–119, 1995.

[71] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *Proc. of the 18th International Conference on World Wide Web (WWW)*, pages 171–180, 2009.

[72] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[73] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proc. of the 33rd ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 593–604, 2007.

[74] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of the 31st ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 49–60, 2005.

[75] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE)*, page 25, 2006.

[76] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proc. of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 123–134, 2010.

[77] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: privacy beyond k-anonymity and ℓ-diversity. In *Proc. of the 23rd IEEE International Conference on Data Engineering (ICDE)*, pages 106–115, 2007.

[78] T. Li and N. Li. Injector: Mining background knowledge for data anonymization. In *Proc. of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 446–455, 2008.

[79] T. Li and N. Li. On the tradeoff between privacy and utility in data publishing. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 517–526, 2009.

[80] X. Li, J. Han, and S. Kim. Motion-alert: automatic anomaly detection in massive moving objects. In *Proc. of the 4th IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 166–177, 2006.

[81] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *Proc. of the 10th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 441–459, 2007.

[82] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proc. of the 34th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 93–106, 2008.

[83] L. Liu, J. Wang, J. Liu, and J. Zhang. Privacy preservation in social networks with sensitive edge weights. In *Proc. of the 9th SIAM International Conference on Data Mining (SDM)*, pages 954–965, 2009.

[84] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: theory meets practice on the map. In *Proc. of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 277–286, 2008.

[85] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[86] P. Marie-Pierre, T. M., and M. Catherine. Smart card data use in public transit: A literature review. *Transportation Research C: Emerging Technologies*, 19(4):557–568, 2011.

[87] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of the 35th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 19–30, 2009.

[88] F. McSherry and R. Mahajan. Differentially private network trace analysis. In *Proc. of the 34th ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 123–134, 2010.

[89] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proc. of the 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[90] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–501, 2011.

[91] N. Mohammed, B. C. M. Fung, and M. Debbabi. Preserving privacy and utility in RFID data publishing. Technical Report 6850, Concordia University, 2010.

[92] D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 210–219, 2004.

[93] A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

[94] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proc. of the 29th IEEE Symposium on Security and Privacy (S&P)*, pages 111–125, 2008.

[95] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. In *Proc. of the 23rd IEEE International Conference on Data Engineering (ICDE)*, pages 1417–1421, 2007.

[96] U.S. Department of Health & Human Services. *The HIPAA Privacy Rule*, 2000 (accessed on June 20, 2012).

[97] M. O'Halloran and M. Glavin. RFID patient tagging and database system. In *Proc. of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, page 162, 2006.

[98] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. PrefixSpan: mining sequential patterns by prefix-projected growth. In *Proc. of the 17th IEEE International Conference on Data Engineering (ICDE)*, pages 215–224, 2001.

[99] M.-P. Pelletier, M. Trepanier, and C. Morency. Smart card data in public transit planning: a review. Technical Report CIRRELT-2009-46, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, 2009.

[100] R. G. Pensa, A. Monreale, F. Pinelli, and D. Pedreschi. Pattern-preserving k-anonymization of sequences and its application to mobility data mining. In *Proc. of the 1st International Workshop on Privacy in Location-Based Applications*, 2008.

[101] S. P. Reiss. Practical data-swapping: the first steps. *ACM Transactions on Database Systems (TODS)*, 9(1):20–37, 1984.

[102] S. P. Reiss, M. J. Post, and T. Dalenius. Non-reversible privacy transformations. In *Proc. of the 1st ACM Symposium on Principles of Database Systems (PODS)*, pages 139–146, 1982.

[103] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 765–774, 2010.

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

[105] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, page 188, 1998.

[106] A. Skowron and C. Rauszer. *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, chapter The discernibility matrices and functions in information systems, pages 331–362. 1992.

[107] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[108] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[109] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proc. of the 9th International Conference on Mobile Data Management (MDM)*, pages 65–72, 2008.

[110] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *Proceedings of the VLDB Endowment*, 1(1):115–125, 2008.

[111] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *Very Large Data Bases Journal (VLDBJ)*, 20(1):83–106, 2011.

[112] M. Utsunomiya, J. Attanucci, and N. Wilson. Potential uses of transit smart card registration and transaction data to improve transit planning. *Transportation Research Record: Journal of the Transportation Research Board*, (1971):119–126, 2006.

[113] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *Proc. of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 466–473, 2005.

[114] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowledge and Information Systems (KAIS)*, 11(3):345–368, 2007.

[115] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In *Proc. of the 4th IEEE International Conference on Data Mining (ICDM)*, pages 249–256, 2004.

[116] T. Wang, S. Meng, B. Bamba, L. Liu, and C. Pu. A general proximity privacy principle. In *Proc. of the 25th IEEE International Conference on Data Engineering (ICDE)*, pages 1279 – 1282, 2009.

[117] D. Wegener, D. Hecker, C. Körner, M. May, and M. Mock. Parallelization of R-programs with GridR in a GPS-trajectory mining application. In *Proc. of the 1st Ubiquitous Knowledge Discovery Workshop*, 2008.

[118] R. C. W. Wong, A. Fu, K. Wang, P. S. Yu, and J. Pei. Can the utility of anonymized data be used for privacy breaches. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(3):16:1–16:24, 2011.

[119] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Anonymization-based attacks in privacy-preserving data publishing. *ACM Transactions on Database Systems (TODS)*, 34(2):8:1–8:46, 2009.

[120] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 754–759, 2006.

[121] R. C. W. Wong, J. Li, A. W. C. Fu, and K. Wang. (α,k)-anonymous data publishing. *Journal of Intelligent Information Systems*, pages 209–234, 2009.

[122] L. Wu, X. Ying, and X. Wu. Reconstruction from randomized graph via low rank approximation. In *Proc. of the 10th SIAM International Conference on Data Mining (SDM)*, pages 60–71, 2010.

[123] X. Xiao, G. Bender, M. Hay, and J. Gehrke. iReduct: differential privacy with reduced relative errors. In *Proc. of the 37th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 229–240, 2011.

[124] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of the 32nd ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 229–240, 2006.

[125] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *Proc. of the 26th IEEE International Conference on Data Engineering (ICDE)*, pages 225–236, 2010.

[126] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Proc. of the 7th VLDB Workshop on Secure Data Management*, pages 150–168, 2010.

[127] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W. C. Fu. Utility-based anonymization using local recoding. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 785–790, 2006.

[128] Y. Xu, B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Publishing sensitive transactions for itemset utility. In *Proc. of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 1109–1114, 2008.

[129] Y. Xu, K. Wang, A. W. C. Fu, and P. S. Yu. Anonymizing transaction databases for publication. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 767–775, 2008.

[130] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: How to hide a MOB in a crowd? In *Proc. of the 12th International Conference on Extending Database Technology (EDBT)*, pages 72–83, 2009.

[131] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *Proc. of the 8th SIAM International Conference on Data Mining (SDM)*, pages 739–750, 2008.

[132] M. Yuan, L. Chen, and P. S. Yu. Personalized privacy protection in social networks. *Proceedings of the VLDB Endowment*, 4(2):141–150, 2011.

[133] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proc. of the 24th IEEE International Conference on Data Engineering (ICDE)*, pages 506–515, 2008.

[134] L. Zou, L. Chen, and M. T. Ozsu. K-automorphism: a general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.