Multiple-sized Bucketization For Privacy Protection

by

Peng Wang
B.Sc., University of Science and Technology of China, 2010

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2014

APPROVAL

Name: Peng Wang

Degree: Master of Science

Title of Thesis: Multiple-sized Bucketization For Privacy Protection

Examining Committee:

Dr. Qianping Gu, Professor
Chair

Dr. Ke Wang, Professor
Senior Supervisor

Dr. Wo-shun Luk, Professor
Supervisor

Dr. Jiangchuan (JC) Liu, Associate Professor
Internal Examiner

Date Approved: May 5th, 2014

Partial Copyright Licence

Abstract

Publishing data without revealing the sensitive information about individuals is an important issue in the field of computer science. In recent years, there are several methods widely used to protect people's privacy: generalization, bucketization and randomization. In this thesis, we begin with giving definition of several well-known privacy protection notions: k-anonymity, ℓ-diversity and t-closeness, and discussing their three major drawbacks, namely, 1) the lack of flexibility for handling different types of variable sensitivity; 2) the large loss of information utility; 3) the vulnerability to auxiliary information. We then propose a new approach by generating the multiple-sized buckets to offer a better protection of individual privacy. This approach also has a higher information utility without violating personal privacy. We design two pruning algorithms for two-sized bucketing: lose-based pruning and privacy-based pruning. Both of them make the two-sized bucketing algorithm perform efficiently for the real data. We also implement a recursive algorithm to test our multiple size bucketing approach. Finally, we apply it to the empirical studies on the real data to demonstrate its effectiveness.

Acknowledgments

I would like to thank my senior supervisor Professor Ke Wang for his support, invaluable guidance, sharing of the knowledge and every opportunity he gave to me. He has always been there giving me advice and direction in research during last three years. At the writing-up stage, he carefully read every version of the draft and put his effort to improve my thesis. I really appreciate his help. His dedication and passion for research has always been exemplary to me. I am grateful to Dr. Wo-shun Luk for being my co-supervisor. I would also thank Dr. Qianping Gu, Dr. Jiangchuan Liu for their advice on my research and defence.

To my friends Bo Hu, Chao Han, Hongwei Liang, Jiahua Yu, Jian Peng, Jiahua Yu, Jing Xv, Judy Yeh, Jun Gan, Peng Peng, Ryan Shea, Weiguang Ding, Weipeng Lin, Xiaoqiang Ma, Yao Wu, Yi Zhong, Yuechen Feng, Zhihui Guo, Zhensong Qian. Thank you for your constant support, encouragement and the great time I have with you during my study in Simon Fraser University.

Finally, I would like to thank my family for their forever love.

Contents

List of Tables

List of Figures

List of Programs

Chapter 1

Introduction

Data mining is an important research area in computer science nowadays when computers have been widely used in all parts of business from the production to the management methods. Large-scale data is collected and accumulated in dramatic pace. There is a huge demand to turn such voluminous data into useful information. Data Mining is the process of extracting useful and previously unknown information from large data sets. It is also known as the analysis step of the "Knowledge Discovery in Databases" process [1]. The main motivation for this research is its real-world applications ranging from the policies making of the governments to marketing for business.

To succeed in data mining, we need high quality data and effective information sharing. The data set must be large enough to contain useful information while remaining concise enough to be mined within an acceptable time limit. Driven by mutual benefit and by regulations that require certain data to be published, the demand for sharing data among various parties is increasing. The ability to exchange data in different formats and the openness to share among different parties are required for effective information sharing. Since detailed data in its original form often contains sensitive information of individuals, sharing such data could potentially leak personal private information [2].

The current privacy protection practice primarily relies on rules and guidelines on the type of publishable data, the use and storage of sensitive data. Rules and guidelines can not prevent those adversaries who do not follow them in the first place. Contract and agreement can not keep a person away from leaking personal information because of carelessness. For example, in 2007, two computer discs holding the personal details of all families with a child under 16 went missing in the UK. The Child Benefit data on them includes name, address, date of birth, National Insurance Number and, where relevant, bank details of 25 million people [3].

It is very important to develop methods and tools for publishing data so that the published data can remain practically useful while individual privacy is well protected. This undertaking is called *privacy-preserving data publishing* (PPDP), which can be viewed as a technical response to complement privacy policies. PPDP has received considerable attention in the community of computer science. In recent papers [2] [4] [5] , the authors gave an excellent introduction on recent development related to PPDP. They systematically reviewed and compared different approaches, discussed the properties of PPDP and proposed future research directions. In what follows, we will first give a brief account on data collecting and data publishing. Then, we outline the arrangement of this thesis.

1.1 Assumptions of Data Collecting and Publishing

There are two phases involved in data collecting and publishing. In the data collection phase, the data holders collect data from record owners. For example, if the hospital collects the personal information from patients, then the data holder is the hospital, and the record owners are the patients. In the data publishing phase, the data holder releases the collected data to a data miner or the public, called the data recipient, who will conduct data mining on the published data. If the hospital publishes the collected data to the health research centre, then the health research centre is the data recipient. The health research centre could conduct data mining (analysis) on the data received.

Data holders can be classified as two types: the untrusted holders and the trusted holders [6]. Untrusted data holders may attempt to identify sensitive information from record owners. Various cryptographic solution, anonymous communication [7], and statistical methods are proposed to collect records anonymously from their owners. For the trusted data holders, record owners are willing to provide their personal information to them. Throughout the whole thesis, we assume that the data holders can be trusted and only consider the privacy issues in the data publishing phase.

Every data publishing situation in practice has its own assumptions and requirements on the data holders, the data recipients, and the data publishing purpose. We can make the following assumptions:

- *The data holder is non-expert.* The data holder does not have the ability or need to perform data mining. Any activities of data mining are performed only by the data recipient. In some cases, the data holder does not even know who will perform the data mining. Thus, the data holder could only release anonymized data set for publishing. In other cases, the data holder is likely interested in the data mining results and knows the data recipient in advance. To obtain the expected result, the data holder could release customized data set with certain specific patterns. In any case, the data recipient will get the data which is different from the original data.

- *The data recipient is possibly an adversary (attacker)* . For example, we give the data to a trusted company. However, it is impossible to guarantee every employee in the company is trustworthy. This makes the encryption approaches useless, in which only the trustworthy recipients have the key to encryption. The encryption does not work because encryption aims to prevent an unauthorized party from accessing the data, but allows the authorized party to have full access to the data. When we give the trustworthy company the access to the data, we will take the risk that some untrustworthy people have the full access to the data too.

1.2 Privacy-Preserving Data Publishing

In the most basic forms of privacy-preserving data publishing (PPDP), the data holder has a table of data in the form

$$\text{T}\ (Explicit\_Identifier, Quasi\_Identifier, Sensitive\_Attributes) \tag{1.1}$$

where the “Explicit Identifier” is a set of attributes, such as names and social security numbers (SSN), containing information that can be used to explicitly identify record owners. The “Quasi Identifier (QID)” [8] is a set of attributes that could potentially, but not sufficiently, identify record owners, such as sex, date of birth and zip codes. The combination of them is distinct ID for the record. The “Sensitive Attributes (SA)” consists of sensitive person-specific information such as disease, salary, disability status and so on [9]. These three sets of attributes are disjoint, that is, the intersection of any two sets is the empty set.

$Anonymization$ [10] is an approach to protect personal privacy by hiding the identity of records and/or the sensitive data of record owners, even after modifying the data, it still can be used for data analysis. Obviously, the “Explicit Identifiers” of record owners must be removed. Even after all the “Explicit Identifiers” have been removed, the personal privacy of data owners can still be leaked to adversaries. Sweeney showed an example in 2002 [11]. He stated that more than 87% of the population in the U.S (216 million of 248 million) are likely unique based on only three “Quasi-Identifiers”: 5-digit ZIP, gender and date of birth. Each of these attributes does not uniquely identify a record owner but their combination, called the QID, often singles out a unique or a small number of record owners. This approach is called linking attack.

To perform such linking attacks, the attacker needs two pieces of prior knowledge: the victim’s record in the released data and the QID of the victim. For example, the attacker notices the victim was hospitalized, and therefore knows that the victim’s medical record would appear in the released patient database. It is rather easy for the attacker to obtain the victim’s zip code, date of birth, and sex, which could serve as the “Quasi-Identifier” in linking attacks. Another famous example is the release of detailed search logs of a large number of users by AOL (American Online). One data mining researcher identified a lot of users based on the logs. AOL quickly acknowledged it was a mistake and removed the logs due to re-identification of the researcher [12].

To prevent linking attacks, the data publisher published an anonymized table

$$T^* \text{ (QID, Sensitive Attributes)},$$

where QID in the published data set T^* is an anonymized version of the original QID obtained by applying anonymization operations to the attributes of QID in the original table T, defined in (1.1). By anonymization operations, some detailed information is concealed so that several records become indistinguishable with respect to QID. Consequently, if a person is linked to a record through QID, this person is also linked to all other records that have the same value for QID. This makes the link between the record owner and QID ambiguous. Alternatively, anonymization operations could generate synthetic data table T^* based on the statistical properties of the original table T, or add noise to the original table T. Such methods usually reduce information represented in the records. This reduction may lead to some loss in data management or the effectiveness of mining algorithms. The anonymization problem is to produce an anonymized T^* that satisfies a given privacy requirement determined by the chosen privacy model and to retain as much data utility as possible. The utility of an anonymized table is measured by information metric.

1.3 Contribution

The main concern of this thesis is privacy-preserving data publishing, that is, publishing data allowing data analysis while preserving data privacy. Below we will discuss the main contributions and arrangement of this thesis.

In Chapter 2, we first discuss the difference between privacy-preserving data publishing and its related works such as privacy-preserving data mining. We then focus on two privacy protection models and introduce some concepts such as k-anonymous, ℓ-diversity, t-closeness and β-likeness. We also discuss some their major limitations, focusing on the poor utility and vulnerability to auxiliary knowledge. They use a uniform privacy requirement. Therefore, all of them cannot deal with the varied sensitivity and varied frequency distribution of different sensitive value very well.

In Chapter 3, we develop so-called multiple-sized bucketing approach to address the limitation of ℓ-diversity. We first discuss the limitations of ℓ-diversity. Then we propose the definition of F'-privacy to specify the different privacy requirement for each sensitive value. Thus the records can be put into buckets of different sizes. The flexibility of bucket sizes makes it easy to find a solution to satisfy the privacy protection requirements. Then we define the utility metric to quantify the information loss of bucketing. At last, we clearly state the problem we are going to solve in this thesis using the definitions of privacy F'-privacy and utility metric. In the next three chapter, we design the protocols to solve this defined problem.

In Chapter 4, we discuss the simplest case of bucketing: one-sized bucketing. We first introduce the Round-Robin bucketing method to assign the records as evenly as possible. Then we propose the validation condition for the one-sized bucketing. Using this condition, we are able to eliminate the unqualified bucket setting without trying to assign records to the buckets.

In Chapter 5, we propose the two-sized bucketing approach. First, we state the validation checking condition in this case and give a partitioning algorithm. To find the optimal bucket setting, we need to list all possible bucketing to do the validation checking. The valid bucket setting with smallest information loss is the optimal solution. There are many possible settings to check. In order to improve the speed of algorithm, we introduce two pruning algorithms for two-sized bucketing: one is loss-based pruning and the other is privacy-based pruning.

In Chapter 6, we first model the multiple-sized bucketing problem and state its validation condition. The multiple-sized bucketing can be solved by Integer Linear Programming (ILP) algorithm. However, as we know, the ILP algorithm is NP hard. Hence, we propose a top-down algorithm by recursively calling the two-sized bucketing algorithm to get an approximate solution for the multiple-sized bucketing.

In Chapter 7, we evaluate several bucketing algorithms, including loss-based pruning and privacy-based pruning algorithm and top-down multiple size algorithm on the real data set CENSUS. The figures illustrate that the high performance of top-down multiple size bucketing algorithm, and effectiveness of the two pruning strategies.

In Chapter 8, we sum up what we did in this thesis and discuss further work to be explored in this direction.

Chapter 2

Background Knowledge

In this chapter, we begin with discussing some related research such as privacy-preserving data mining, privacy-preserving data publishing and statistical disclosure control. We then introduce several well-known anonymization approaches of data publishing by going through two main privacy protection models. Finally, we outline some drawbacks of these approaches.

2.1 Related Research Areas

Motivated by the growing concern about the personal privacy information in published data, a concept called *privacy-preserving data mining* (PPDM) was introduced in 2000s [13] [14] [15]. The initial idea of PPDM was to extend traditional data mining techniques to work with the modified data after hiding sensitive personal information. One of the key issues was how to process the data mining to the modified data. Usually we need to take the data mining algorithms into consideration. In contrast, privacy-preserving data publishing (PPDP) does not necessarily consider specific data mining tasks, which is usually unknown in the data publishing phase.

The difference between PPDP and PPDM lies in the following several aspects.

- The main focus of PPDP is on the techniques for publishing data rather than the the techniques for data mining. It is expected that standard data mining techniques are applied to the published data. The data holder in PPDM needs to randomize the data in such way that data mining results can be recovered from the randomized data. In this case, the data holder must fully understand the data mining tasks and the algorithms involved. However, the data holder in PPDP usually is not an expert in data mining.

- Both randomization and encryption do not preserve the truthfulness of values at the record level. Therefore, the released data is basically meaningless at the record level to the recipients. In such case, the data holder in PPDM may consider releasing the data mining results rather than publishing the modified data.

- PPDP primarily "anonymizes" the data by hiding the identity of record owners, whereas PPDM seeks to directly hide the sensitive data. So PPDP can keep more information in the modified data set, people can use the data mining techniques to find the interesting information according their needs.

Another related research area is in the field of *statistical disclosure control (SDC)* [16], where the research focuses on privacy-preserving publishing methods for statistical tables. There are mainly three types of data disclosures: 1) identity disclosure, 2) attribute disclosure, 3) inferential disclosure [17]. Identity disclosure occurs when an individual is linked to a particular record in the released table. Attribute disclosure occurs when new information about some individuals is revealed. For example, the released data makes it possible to infer the characteristics of an individual more accurately than it would be possible before the data release. Identity disclosure often leads to attribute disclosure. Once there is identity disclosure, an individual is re-identified and the corresponding sensitive values are revealed. Attribute disclosure can occur with or without identity disclosure. Inferential disclosure occurs when information can be inferred with high confidence from statistical properties of the released data. For example, the data may show a high correlation between income and purchase price of a home. As the purchase price of a home is typically public information, a third party might use this information to infer the income of a data subject.

The work of SDC involves the non-interactive query model [18] and the interactive query model [19] [20]. In the study of the non-interactive query model, the data recipient can submit one query to the system. This type of non-interactive query model may not fully address the information needs of data recipients because, in some cases, it is very difficult for a data recipient to accurately construct a query for a data mining task in one shot. In the study of the interactive query model, the data recipients, including adversaries, can submit a sequence of queries based on previously received query results. The database server is responsible for keeping track of all queries of each user and determine whether or not the currently received query has violated the privacy requirement with respect to all previous queries.

There has been a series of results [18] [20] [21] [22] that suggests an adversary (or a group of corrupted data recipients) will be able to reconstruct all but $1 - o(1)$ fraction of the original data exactly, which is a serious violation of privacy. The interactive privacy-preserving query model may only answer a sub-linear number of queries in total. When the maximum number of queries is reached, the query service must be closed to avoid privacy leak. In the case of the non-interactive query model, the adversary can issue only one query and, therefore, the non-interactive query

Job	Sex	Age	Disease
Engineer	Male	35	Cancer
Lawyer	Male	38	HIV
Engineer	Male	38	Cancer
Singer	Female	30	Flu
Singer	Female	30	HIV
Dancer	Female	30	HIV
Dancer	Female	30	Flu

Table 2.1: Table of Original Patient Data in Example 1

model can not achieve the same degree of privacy defined by the interactive model. We can consider that privacy-preserving data publishing is a special case of the non-interactive query model.

The main focus of this thesis is PPDP. We will discuss two main privacy protection models in PPDP, namely the record linkage model and the attribute linkage model.

2.2 The Record Linkage Model

If a privacy threat occurs when an attacker is able to link a record owner to a record in a published data table, this threat is called record linkage [23] [24]. In the record linkage model, some value of QID identifies a small number of records in the published table T. If the victim's QID get identified, the victim is vulnerable to being linked to the small number of records in this group. Hence, there are only a small number of possibilities for the victim's record, the adversary has a good chance to uniquely identify the victim's record from this group under the help of additional knowledge.

Example 1. *Suppose that a hospital publishes patients' records in Table 2.1 to a research center. Suppose that the research center has access to the external Table 2.2 and knows that every person with a record in Table 2.2 has a record in Table 2.1. If two tables on the QID attribute (Job, Sex, Age) are joined, the adversary may link the identity of a person to his/her Disease. For example, Bob, a 35-year-old male engineer, is identified as an cancer patient by* $QID =< Engineer, Male, 35 >$ *after the join.*

In 1998, Samarati and Sweeney [25] [26] [11] proposed the notion of k-anonymity in order to prevent the record linkage attack through QID: if one record in the table have some value of QID, at least k-1 other records also have the same value of QID. This means that the minimum group size on QID is at least k. A data publishing policy satisfying this requirement is called k-anonymous. For the k-anonymous policy, each record in the data set is indistinguishable from at least k-1 other records with respect to same QID. Thus the probability of linking a victim to a specific record through QID is at most 1/k.

Name	Job	Sex	Age
Bob	Engineer	Male	35
John	Lawyer	Male	38
Jack	Engineer	Male	38
Alice	Singer	Female	30
Mary	Singer	Female	30
Gayze	Dancer	Female	30
Emily	Dancer	Female	30

Table 2.2: Table of External Data in Example 1

Definition 1. *Given a table, if its minimum group size on QID is at least k, we say this table satisfies k-anonymity.*

Job	Sex	Age	Disease
Professional	Male	$[35-40)$	Cancer
Professional	Male	$[35-40)$	Cancer
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

Table 2.3: Data Set of 3-anonymous Patient Data

Example 2. *Table 2.3 shows a 3-anonymous table generated from Table 2.1. It has only two distinct groups on QID, namely $\{Professional, Male, [35-40)\}$ and $\{Artist, Female, [30-35)\}$. Since each group contains at least 3 records, the table is 3-anonymous. If we try to link the records in Table 2.2 to the records in Table 2.3 through QID, each record is linked to either no record or at least 3 records. By this way the privacy of data owners is protected.*

Notice that k-anonymity does not take the sensitive information into account. So the records in one group may share one sensitive value, which causes the information leak. The sensitive attributes play an important role in the attribute linkage model we are going to discuss next section.

2.3 The Attribute Linkage Model

If a privacy threat occurs when an attacker is able to link a sensitive attribute in a published data table, this threat is called attribute linkage [2]. In the attribute linkage model, the adversary could infer his/her sensitive values from the published data T^* without being able to identify the record of the target victim exactly. That is done by associating the set of sensitive values associated to the group that the victim belongs to. Even if k-anonymity is satisfied, attribute linkage can still pose privacy into threat since an attacker can still find out the relation for some sensitive values in a group. This can be seen in the following example.

Example 3. *Consider the 3-anonymous data in the Table 2.3. Suppose the attacker knows that the target victim Mary is a female singer at age 30 and owns a record in the table. The adversary may infer that Mary has HIV with* 75% *confidence because 3 out of 4 female artists within the age of* $[30, 35)$ *have HIV. Regardless of the correctness of the inference, Mary's privacy has been disclosed.*

To eliminate attribute linkage threat, Clifton [27] suggested limiting the released data size. As we know that some sensitive data such as HIV patients' data should be hard to obtain. Limiting data size will make the data easy to attack. Several other approaches have been proposed to address the issue of privacy threat caused by attribute linkage. The main idea is to disconnect the link between QID attributes and sensitive attributes.

In what follows we will discuss three approaches: ℓ-diversity, t-closeness and β-likeness.

2.3.1 ℓ-Diversity

Although k-anonymity protects against identity disclosure, it is insufficient to prevent attribute disclosure. To solve this problem, Machanavajjhala et al. [28] introduced a new notion of privacy principle, called ℓ-diversity, to prevent attribute linkage. It not only maintains the minimum group size, but also maintains the diversity of the sensitive attributes. The ℓ-diversity model for privacy is defined as follows:

Definition 2. *Let* qid *be a value of the* QID *attribute in published table* T^**. A* qid*-block is defined to be a set of tuples such that their* qid *values are generalized into the unified value. We say a* qid*-block is* ℓ*-diversity if it contains at least* ℓ *well-represented values for sensitive attribute. A table is* ℓ*-diversity if every* qid*-block is* ℓ*-diversity.*

The principle of ℓ-diversity is to ensure ℓ well-represented values for the sensitive attribute in every qid-block. Distinct ℓ-diversity ensures that each qid-block has at least ℓ distinct values for sensitive attribute. Distinct ℓ-diversity does not prevent probabilistic inference attacks. It may happen that in an anonymized class one value appears much more frequently than other values, enabling the adversary to conclude that an entity in the qid-block is very likely to have that value. For example, in one qid-block, there are ten tuples. In the "Disease" attribute, one of them is "Cancer", one is "HIV", and the remaining eight are "Flu". This satisfies 3-diversity, but the attacker can still get the conclusion that the person's disease is "Flu" with the accuracy of 80$\%$.

In 2006, Xiao and Tao [29] defined the ℓ-diversity partition that guarantees the ℓ-diversity. A partition consists of several subset of T^* such that each tuple in T^* belongs to exactly one subset.

We refer to these subsets as q_j $(1 \leq j \leq m)$. Namely, $\cup_{j=1}^{m} q_j = T$ and, for any $1 \leq j_1 \neq j_2 \leq m$, $q_{j_1} \cap q_{j_2} = \emptyset$.

Definition 3. *(ℓ-diversity partition) A partition of the table T^* with m subsets is ℓ-diversity if each subset q_j $(1 \leq j \leq m)$ satisfies the following condition. Let v be the most frequent value in q_j, and $c_j(v)$ the number of tuples in this subset. Then*

$$c_j(v)/\left|q_j\right| \leq 1/\ell \tag{2.1}$$

where $|q_j|$ is the size (the number of tuples) of the subset q_j.

Later in the thesis, we mean ℓ-diversity partition whenever we use the terminology ℓ-diversity.

An ℓ-diversity partition exists if and only if the original data T satisfies the $eligibility\ condition$ [28]: at most n/ℓ tuples are associated with the every sensitive value, where n is the size of T. The ℓ-diversity has some advantages. It does not require the knowledge of the full distribution of the sensitive values. We only need the number of tuples with most frequent sensitive values. The larger ℓ is, the more information is hidden in the data. However, for all different sensitive value, it require them to have unified privacy requirement. This is unnecessary and hard to achieve. For example, we have a data set containing 10000 patients with sensitive attribute being disease. In this data, one of patients has HIV, 100 of them have Cancer and the rest of them have Flu. Obviously we have $n = 10000$. Following from the above eligibility condition, for any ℓ larger than 1 we can not achieve the ℓ-diversity since, for the sensitive value Flu, we have

$$10000 - 1 - 100 = 9899 > 10000/\ell$$

when $\ell > 1$. The ℓ-diversity cannot be satisfied if the distribution is skewed, or small ℓ must be used, which does not provide sufficient protection.

2.3.2 t-Closeness

Li et al. [30] observed that when the overall distribution of a sensitive attribute is skewed, ℓ-diversity does not prevent from 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. We can not assign any value of $\ell > 2$ to solve this problem. If $\ell = 3$, the possibilities of these two sensitive values are both smaller than or equal to $1/3$ according to Definition 3. The QID group is not fully filled. However, this group presents a serious privacy threat because any record owners in the group could be inferred as having HIV with 50% confidence, compared to 5% in the overall table.

To prevent skewness attack, Li et al. proposed a privacy criterion, called t-closeness, which requires the distribution of a sensitive attribute in any group on QID to be close to the distribution of the attribute in the overall table.

Definition 4. *A subset is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all subsets have t-closeness.*

In this definition, the *Earth Mover's Distance(EMD)* [31] function is used to measure the distance between two distributions of sensitive values. We will not give the definition of this function here. We refer to [31] for the details.

There are several limitations and weakness in t-closeness. First, it lacks the flexibility of specifying different protection levels for different sensitive values. We can only specify the value t representing the overall distribution changes. Second, it is not clear how the value t is related to the information gain. Its relation to the level of privacy is also very complicated. Third, enforcing t-closeness would greatly degrade the data utility because it requires the distribution of sensitive values to be the same in all QID groups. This would significantly damage the correlation between QID and sensitive attributes. One way to reduce the damage is to relax the requirement by adjusting the thresholds with the increased risk of skewness attack, or to employ the probabilistic privacy models [32].

2.3.3 β - Likeness

In 2012, Jianneng Cao and Panagiotis Karas stated that just like ℓ-diversity is open to many ways of measuring the number of "well-represented" value in qid-block, the t-closeness model is open to diverse ways of measuring the cumulative difference between the overall distribution and that a privacy model should provide grounds for effective and human-understandable policy. They pointed out that any functions aggregate absolute difference, including EMD, do not provide a clear privacy guarantee. So they introduced the β-likeness [33], which is an appropriately robust privacy model for microdata anonymization. It guarantees that adversary's confidence on the likelihood of a certain SA value should not increase by a predefined threshold.

Definition 5. *For table T with sensitive attribute SA, let $V = \{v_1, \ldots, v_m\}$ be the SA domain, and $P = (p_1, \ldots, p_m)$ the overall SA distribution in T. An equivalence class with SA distribution $Q = (q_1, \ldots, q_m)$ is said to satisfy β-likeness if and only if $max_i\{D(p_i, q_i), p_i \in P, q_i \in Q, p_i < q_i\} \leq \beta$, where D is a distance function between p_i and q_i and $\beta > 0$ is a threshold.*

They define the distance function as $D(p_i, q_i) = \frac{q_i - p_i}{p_i}$, so they can get the relative difference instead of the absolute difference. They greatly improve the performance of t-closeness. Two anonymization schemes are used. One is based on generalization, and the other one is based on perturbation. They use unified threshold for every different sensitive value. ℓ-diversity has one absolute bond for the possibility distribution $1/\ell$. And β-likeness use a unified relative bond for the possibility distribution. $D(p_i, q_i) = \frac{q_i - p_i}{p_i} \leq \beta$. The threshold requires the possibility distribution in modified data must be related to the original possibility distribution. The sensitivity level is not necessary related to the original possibility distribution. For example, as we know, HIV is very sensitive for a medical data set. If we have one original medical data set, the possibility of HIV in it is relatively high, it does not mean HIV is less sensitive in this data set. They also have noticed the phenomenon that when the distribution of SA value is skewed, it is highly possible to get unsatisfactory solution by generation.

In the next chapter, to overcome the disadvantage of the ℓ-diversity, we will propose a multiple-sized bucketization approach to protect the personal privacy. We require that the sizes of buckets are only related to the records in the buckets. The more sensitive records are put into the larger buckets. The less sensitive records are put into the smaller buckets.

Chapter 3

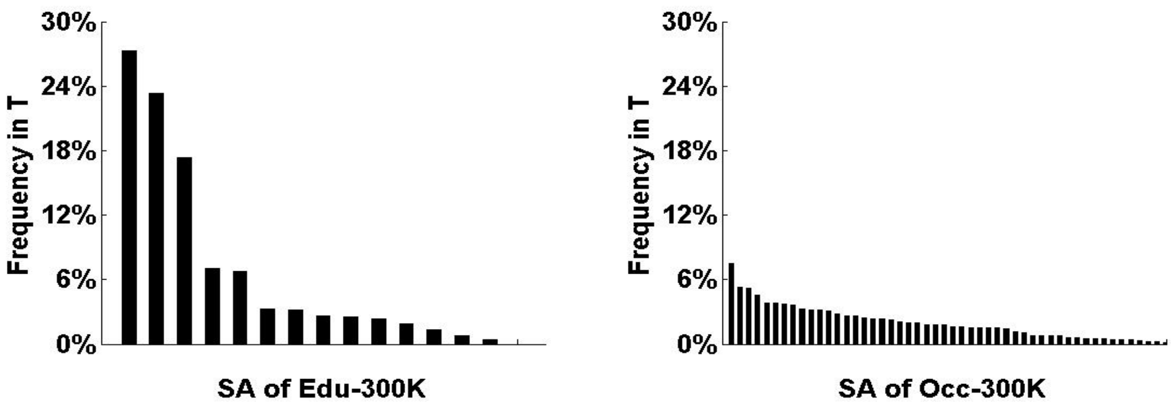
Privacy and Utility Specification

In the rest of this thesis, we will describe our protocol for multi-sized bucketization approach to address the limitations of ℓ-diversity defined in Section 2.3.1. To do so, we first define the privacy requirement and the information loss metric in this chapter. Then we will define the problem we are going to study in this thesis. The main motivations come from limitations of the ℓ-diversity.

3.1 Limitations of ℓ-diversity

It is sometimes difficult and unnecessary to achieve ℓ-diversity. Moreover, ℓ-diversity is insufficient to prevent personal information disclosure. For ℓ-diversity, each subset requires not only having enough different sensitive values, but also the different sensitive values being distributed evenly. Thus it is unable to handle sensitive values that have skewed distribution and varied sensitivity. However, in many real-life applications the privacy thresholds vary for different sensitive values. For example, we assume that HIV is more sensitive than Flu, so HIV requires a smaller threshold $1/\ell$, thus, a larger ℓ for ℓ-diversity principle must set according to the most sensitive value. According to the eligibility condition of ℓ-diversity, it is very difficult to achieve ℓ-diversity for large ℓ. Such specification is unnecessarily hard to satisfy, which leads to either excessive distortion or no data being published. For example, the CENSUS data have 300k records with "Occupation" attribute or "Education" attribute as a sensitive attribute. As the Figure 3.1 shows, the frequency distribution of "Education" is more skewed. In the case of considering "Occupation" as a sensitive attribute, the maximum and the minimum frequency are 7.5% and 0.18%, so the maximum ℓ for ℓ-diversity is 13 because of the eligibility condition is defined by the most frequent sensitive attribute $\ell \leq$ 1/7.5% = 13.33. Therefore, it is impossible to protect the sensitive records which have higher privacy protection requirement by having bigger ℓ, such as ℓ = 25.

Even if it is possible to achieve such ℓ-diversity, enforcing ℓ-diversity with big ℓ for all different sensitive values leads to a large information loss of answering queries after the bucketization [34] [35]. Bucketization hides the association between QID values and the SA values of every record in the buckets. This is widely used in information protection of large data set. For example, if we put ten records into one bucket, we only know ten of them share the ten SAs, we loss the information about their original SAs. The bucket size indirectly determines the information loss. If we put 500 records into one bucket, the information loss of it is not as small as if we put them into 100 buckets, which have 5 records in each bucket.



(a) EDU (b) OCC

Figure 3.1: Frequency Distribution of SA

3.2 Bucketization

We consider a bucketization problem in which buckets of different sizes can be formed to satisfy different privacy requirements. The large buckets are used for records having more sensitive attributes and the small buckets are used for records having less sensitive attributes. This approach is called multiple-sized bucketing. In this approach, the records in T are grouped into different size buckets. Each bucket of QID is related to a bucket of SA. Consider a data table $T(QID, SA)$, where QID is a set of attributes $\{A_1, \cdots, A_d\}$, called the quasi-identifier, and SA is the sensitive attribute. The domain of SA is $\{x_1, \cdots, x_m\}$. The symbol o_i denotes the number of records for x_i in T and $f_i = o_i/|T|$ denotes the frequency of x_i in T, where $|T|$ is the number of records in T. When we publish the data, we will put records into different buckets, with an unique bucket ID, called "BID". Let T^* denote the published version of T. We use g to refer to both a bucket and the bucket ID of a bucket, depending on the context. T^* is published in two tables, $QIT(QID, BID)$ and $ST(BID, SA)$. Every record r in T is grouped into a bucket g. Attacker wants to infer the SA value of a target individual t. The adversary has access to T^*. For each SA value x_i, $Pr(x_i|t, T^*)$

denotes the probability that an individual t is inferred to have a sensitive value x_i. For now, we consider an adversary with the following auxiliary information: a t's record is contained in T, t's values on QID, i.e. $t[QID]$, and the algorithm used to produce T^*.

For a target individual t with $t[QID]$ contained in a bucket g, $|g, x_i|$ denotes the number of occurrence of (g, x_i) in ST and $|g|$ denotes the size of bucket g. The probability of inferring the SA value of x_i is $Pr(x_i|t, g)$, which is equal to $|g, x_i|/|g|$.

$$Pr(x_i|t, g) = |g, x_i|/|g|$$

We use $Pr(x_i|t, T^*)$ to define maximum $Pr(x_i|t, g)$ for any bucket g containing $t[QID]$.

$$Pr(x_i|t, T^*) = max(Pr(x_i|t, g)) \qquad \forall g$$

Example 4. *A hospital maintains a data set for answering count queries on the medical data such as T in Table 3.1, which contains four columns, Gender, Age, Zipcode and Disease. Among them, the Gender, Age and Zipcode are QID. The Disease is SA. Names of data holders have already been deleted to protect the privacy of patients.*

Gender	*Age*	*Zipcode*	*Disease*
Male	*40*	*54321*	*Brain Tumor*
Female	*20*	*54321*	*Flu*
Female	*20*	*54324*	*HIV*
Male	*32*	*54322*	*Flu*
Female	*57*	*61234*	*Cancer*
Female	*22*	*61434*	*HIV*
...	...	...	...

Table 3.1: Data Set of Example 4

Table 3.2(a) and Table 3.2(b) show the tables of QIT and ST for one bucketization. To infer the SA value of Alice with $QID = \langle Female, 61434\rangle$, the adversary first locates the bucket that contains

(a) QIT of anonymized table T^*

Gender	*Age*	*Zipcode*	*BID*
Male	*40*	*54321*	*1*
Female	*20*	*54321*	*1*
Female	*20*	*54324*	*1*
Male	*32*	*54322*	*2*
Female	*57*	*61234*	*2*
Female	*22*	*61434*	*2*
...	...	...	...

(b) ST of anonymized table T^*

BID	*Disease*
1	*Brain Tumor*
1	*Flu*
1	*HIV*
2	*Flu*
2	*Cancer*
2	*HIV*
...	...

Table 3.2: Published Table T^* of Example 4

$\langle Female, 61434\rangle$, *i.e.,* $BID = 2$. *There are three diseases in this bucket, Flu, Cancer and HIV, each occurring once. So* $Pr(x_i|Alice, 2) = 33.3\%$, *where* x_i *is Flu, Cancer or HIV. Both two buckets satisfy the* 3*-diversity partition rule mentioned in the Definition 3, for all diseases,* $Pr(x_i|t, g) \leq 1/3$. *It means this partition satisfies the* 3*-diversity principle. Now the adversary only knows the patients have one of the three diseases, but does not know which one of three diseases is. So this bucketization causes the information loss.*

3.3 Privacy Specification

To overcome constraints of ℓ-diversity mentioned in the Section 2.3.1, which is unified privacy requirements for different sensitive value, we propose the corresponding privacy requirement for each sensitive value x_i as follows:

Definition 6. *[F'-Privacy] For each* SA *value* x_i, f_i'-privacy *specifies the requirement that* $Pr(x_i|t, T^*) \leq f_i'$, *where* f_i' *is a real number in the range (0, 1].* F'-privacy *is a collection of* f_i'*-privacy for all* SA *values* x_i.

For example, the publisher may set $f_i' = 1$ for some x_i's that are not sensitive at all, set f_i' manually to a small value for the highly sensitive values x_i, and set $f_i' = min\{1, af_i + c\}$ for the rest of SA values whose sensitivity grows linearly with their frequency, where a and c are constants. In our approach, we assume that f_i' is specified but does not depend on how f_i' is specified and f_i' is not necessarily related to f_i. Since we do not use unified privacy requirement as for ℓ-diversity, the relatively more sensitive records, such as the ones having HIV, which has higher privacy requirement, are put into the bigger buckets, and less sensitive records, such as the ones having Flu, which has lower privacy requirement, are put into smaller buckets. This meets the privacy protection requirement with lower information loss. It makes the F'-privacy specification suitable for handling SA of skewed distribution and varied sensitivity. We will evaluate this claim on real life data sets in Chapter 7. For the rest of thesis, we assume $f_i' \geq f_i$ for all x_i. This assumption can be justified by the following statement:

Lemma 1. *If a bucketization* T^* *satisfying* F'*-privacy exists, then* $f_i' \geq f_i$ *for all* x_i.

Proof. Suppose that we put the data into n buckets g_j $(j = 1, \ldots, n)$, each bucket size $|g_j| < |T^*|$. $Pr(x_i|t, T^*)$ is the maximum $Pr(x_i|t, g_j)$ of any buckets g_j. We know the occurrence of x_i is o_i.

$$o_i = \sum_j Pr(x_i|t, g_j)|g_j| \leq \sum_j Pr(x_i|t, T^*)|g_j| = Pr(x_i|t, T^*) \sum_j |g_j|$$

so we can get

$$f_i = \frac{o_i}{\sum_j |g_j|} \leq Pr(x_i|t, T^*) = f_i'$$

Therefore, f_i' is bigger than or equal to f_i. □

It follows from the above lemma that, if there exists one x_j such that $f'_j < f_j$, there is no bucketization to satisfy it.

3.4 Utility Metric

One concern for data publishing is the information loss when the privacy requirement is satisfied. For each bucket g, every record in it is associated to the SA values in g with an equal chance. In the original data set, each record can be treated as a bucket. The accuracy to infer its sensitive attribute is 100%. After the bucketization, if we have two records in one bucket, one of them has Flu, and the other one has Cancer. The accuracy of them having Flu or Cancer is 50%. For the same data set, the records in small buckets have high accuracy to be associated with the SA, and the records in large buckets have low accuracy to be associated with the SA. Thus the bucket size $|g|$ can represent the accuracy level of associating sensitive values with records. We define the information loss as follows:

Definition 7. *Let T^* consist of a set of buckets $\{g_1, \cdots, g_b\}$. The $Information\ Loss\ (IL)$ of T^* is defined by*

$$IL(T^*) = \frac{\sqrt{\sum_{i=1}^{b}(|g_i| - 1)^2}}{|T^*| - 1} \tag{3.1}$$

The immediate result we can get from this definition is as follows:

Lemma 2. *For any bucketization T^* consisting of a set of buckets $\{g_1, \cdots, g_b\}$, the values of $IL(T^*)$ lie in the range of $[0, 1]$.*

Proof. The raw data T^* is one extreme case in which each record itself is a bucket, then $|g_i| = 1$, therefore $IL = 0$. The single bucket containing all records is the other extreme case where $|g_1| = |T^*|$ and $IL = 1$. We now consider all other cases of T^*. Obviously, $IL > 0$ We only need to prove that

$$\sqrt{\sum_{i=1}^{b}(|g_i| - 1)^2} < |T^*| - 1, \qquad \text{that is,} \qquad \sum_{i=1}^{b}(|g_i| - 1)^2 < (|T^*| - 1)^2.$$

Knowing $|T| = \sum_{i=1}^{b} |g_i|$, by direct calculation we can prove the above inequality. □

Since $|T|$ is constant, to minimize IL, we shall minimize the following loss metric:

$$Loss(T^*) = \sum_{i=1}^{b}(|g_i| - 1)^2 \tag{3.2}$$

Note that the loss metric $Loss$ has the additivity property: if $T^* = T_1^* \cup T_2^*$, then $Loss(T^*) = Loss(T_1^*) + Loss(T_2^*)$. This is very useful when we use the Top down bucketing algorithm in Section

6.2. This definition will also be extensively used for the information loss of the buckets in next chapter when we compare the information losses to decide whether we should divide the data into two parts or not.

3.5 Problem Definition

In this section, we consider the general form of the bucketization problem where the number of different size buckets are unknown and are determined by the minimization of the loss function.

Let B_j be a set of all S_j-sized buckets and the quantity of elements in B_j is denoted by b_j, where $S_1 < \cdots < S_k$ and $b_j > 0$, $j = 1, \cdots, k$. We say that $\cup_j B_j$ is a solution with respect to (T, F') if there is a distribution of records in T to the buckets in $B_1, \cdots, B_k$ such that all bucket slots are filled and no frequency of a value x_i in a bucket is more than its f'_i. Following from Equation (3.2), the collection of buckets specified by $\cup B_j$ has the loss $\sum_{j=1}^{q} b_j(S_j - 1)^2$, we denote this loss by $Loss(\cup B_j)$. A solution is optimal if it has the minimum loss among all solutions with respect to (T, F'). The generalized problem is defined below.

Definition 8. *(Optimal Multiple-sized Bucket Setting) Given a table T and F'-$privacy$, the generalized bucketing problem is to find $(b_1, \cdots, b_k, S_1, \cdots, S_k)$ such that $\cup_j B_j$ has the minimum $Loss(\cup B_j)$ with respect to (T, F').*

Notice that the input to this problem is only T and F'; the number k of different bucket sizes is unknown and must be determined.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
o_i	2	2	2	4	4	4	4	7	7
f_i	0.056	0.056	0.056	0.11	0.11	0.11	0.11	0.19	0.19
f'_i	0.187	0.187	0.187	0.353	0.353	0.353	0.353	0.603	0.603
S^*	6	6	6	3	3	3	3	2	2

Table 3.3: Data Set T of Example 5

Example 5. *Assume we have a data set T with 36 records as shown in Table 3.3. The privacy requirement F' is defined by $f'_i = 3f_i + 0.02$. S^* in Table 3.3 represents the possible smallest buckets which records can be put into without violating the privacy requirement. The optimal solution is the solution which can get the smallest information loss without violating privacy requirements. So we can get the optimal buckets setting with smallest information loss, defined in the Equation (3.2):*

$$Loss(T^*) = \sum_{i=1}^{b}(|g_i| - 1)^2 = 2 * (6-1)^2 + 4 * (3-1)^2 + 4 * (2-1)^2 = 70$$

$Bucket_1$	x_1	x_2	x_3	x_8	x_8	x_9
$Bucket_2$	x_1	x_2	x_3	x_8	x_9	x_9
$Bucket_3$	x_4	x_5	x_6			
$Bucket_4$	x_4	x_5	x_6			
$Bucket_5$	x_4	x_5	x_6			
$Bucket_6$	x_4	x_5	x_6			
$Bucket_7$	x_8	x_9				
$Bucket_8$	x_8	x_9				
$Bucket_9$	x_8	x_9				
$Bucket_{10}$	x_8	x_9				

Table 3.4: Optimal Bucketing of Example 5

shown in Table 3.4. We also show the non-optimal solution as shown in the next example. In this buckets setting, we get six buckets of six records. The information loss is

$$Loss(T^*) = 6 * (6 - 1)^2 = 150.$$

The example clearly demonstrate the optimal bucketing has the smallest information loss. Next, we will introduce the one-sized bucketing, two-sized bucketing and multiple-sized bucketing accordingly. One-sized bucketing means there is only one kind of bucket size while two-sized bucketing represent a bucket setting having two kinds of bucket size. Multiple-sized bucketing protocol is the the protocol we are looking for this problem.

Chapter 4

One-sized Bucketing

In this chapter, we discuss the simplified version of multiple bucketizaiton: one-sized bucketing. We first introduce the data assignment for one-sized bucketing. Then we define the validation conditions for one-sized bucketing.

4.1 Data Assignment

In this section, we consider the bucketing with only one kind of bucket size, that is, a set of b buckets of same size S. So we have $B = \{g_0, \ldots, g_{b-1}\}$. Let $|B|$ denote the total capability of all buckets in B, i.e., $|B| = b * S$. Suppose that we want to assign the records in T to the buckets in B. We assume that B has exactly the same capacity as the number of records in T, i.e., $|T| = |B|$. For a given F'-privacy, which is defined in Definition 6, we say that there exists a solution for (T, F') if there is a distribution of records in T to the buckets in B such that all bucket slots are filled and no frequency of a value x in the bucket is more than its required frequency f'. To assign the records as evenly as possible, we introduce a Round-Robin Bucketing (RRB) method. In the RRB, the records for x_i are assigned to buckets in circular order with same possibility. We start from the first positions of buckets. After all of them are filled, we go to fill the next positions of all buckets until all the records are assigned to the buckets. Therefore, the maximum and minimum occurrences of records with same sensitive value in any bucket differ by at most 1. A RRB solution is an assignment obtained by RRB method. The order of assignment for o_i does not matter in the one size bucketing.

For each value x_i, $1 \leq i \leq m$, we assign the t-th record of x_i to the bucket g_s, where $s = (o_1 + \cdots + o_{i-1} + t) \mod b$, where o_i is the number of occurrence of x_i in T. In other words, the records for x_i are assigned to the buckets in a round-robin manner. It is easy to see that the number of records for x_i assigned to a bucket is either $\lfloor |o_i|/b \rfloor$ or $\lceil |o_i|/b \rceil$.

Example 6. *We are given a table containing 36 tuples with o_i given in the Table 3.3. Suppose that the F'-privacy is given by $f'_i = 3f_i + 0.02$. For the least frequent sensitive value, we have $o_1 = 2$, $f_1 = 0.005556$. So $f'_1 = 0.1866$ and $S = \lceil \frac{1}{f'_1} \rceil = 6$. Consider the bucket setting $S = 6, b = 6$. We put the records into the buckets. The order of records assignment is not important in the one size bucketing, the key point is assigning the records into buckets as evenly as possible. For the two-sized bucketing and multiple-sized bucketing, the order of assignment is important, so we will introduce the record partitioning in Section 5.2. After the record partitioning, the records assignment for two-sized bucketing can be treated as one-sized bucketing, the order of records assignment becomes not important. Step one, we assign x_9 and obtain the result as shown in the Table 4.1(a). Step two, we deal with o_8, which has the same frequency as o_9. We get the Table 4.1(b). We assign the rest tuples accordingly. Eventually we complete the assignment and obtain in Table 4.2.*

(a) Step one

$Bucket_1$	x_9	x_9				
$Bucket_2$	x_9					
$Bucket_3$	x_9					
$Bucket_4$	x_9					
$Bucket_5$	x_9					
$Bucket_6$	x_9					

(b) Step two

$Bucket_1$	x_9	x_9	x_8			
$Bucket_2$	x_9	x_8	x_8			
$Bucket_3$	x_9	x_8				
$Bucket_4$	x_9	x_8				
$Bucket_5$	x_9	x_8				
$Bucket_6$	x_9	x_8				

Table 4.1: Procedure of Assignment of Example 6

$Bucket_1$	x_9	x_9	x_8	x_6	x_5	x_3
$Bucket_2$	x_9	x_8	x_8	x_6	x_5	x_3
$Bucket_3$	x_9	x_8	x_7	x_6	x_4	x_2
$Bucket_4$	x_9	x_8	x_7	x_6	x_4	x_2
$Bucket_5$	x_9	x_8	x_7	x_5	x_4	x_1
$Bucket_6$	x_9	x_8	x_7	x_5	x_4	x_1

Table 4.2: Record Assignment for RRB of Example 6

4.2 Validity Checking

Lemma 3. *(Validating One-Size Bucket Setting) Let B denote a set of b buckets of size S, T denote a set of records, and F' denote the parameter for F'-privacy. The following statements are equivalent:*

(1). There exists a solution B, a set of b buckets of size S for given T and F'. In this case, we simply say $Valid(B, T, F') = true$.

(2). There is a assignment of RRB from T to B with respect to F'.

(3). For each value x_i, the number of records for it is:

$$\frac{\lceil o_i/b \rceil}{S} \leq f'_i \tag{4.1}$$

(4). For each value x_i, the number of records for it is

$$o_i \leq \lfloor f'_i S \rfloor b \tag{4.2}$$

Proof. We are going to show $(4) \Rightarrow (3) \Rightarrow (2) \Rightarrow (1) \Rightarrow (4)$. Observe that if r is a real number and i is an integer, $r \leq i$ if and only if $\lceil r \rceil \leq i$, and $i \leq r$ if and only if $i \leq \lfloor r \rfloor$. Using this observation, we have:

$$\frac{\lceil o_i/b \rceil}{S} \leq f'_i \Leftrightarrow \lceil o_i/b \rceil \leq f'_i S \Leftrightarrow \lceil o_i/b \rceil \leq \lfloor f'_i S \rfloor \Leftrightarrow o_i/b \leq \lfloor f'_i S \rfloor \Leftrightarrow o_i \leq \lfloor f'_i S \rfloor b \tag{4.3}$$

This shows the equivalence of (4) and (3). To prove $(3) \Rightarrow (2)$, notice that $\frac{\lceil o_i/b \rceil}{S}$ is the maximum frequency of x_i in a bucket generated by RRB. Statement (3) implies that this assignment is valid. $(2) \Rightarrow (1)$ follows because every valid RRB is a valid assignment.

Notice that F'-privacy implies that the number of occurrence of x_i in a bucket of size S is at most $\lfloor f'_i S \rfloor$. Thus for any valid assignment, the total number of occurrence o_i in the b buckets of size S is no more than $\lfloor f'_i S \rfloor b$. This complete the proof of $(1) \Rightarrow (4)$. □

When we have a possible buckets setting $B(b, S)$, for every sensitive values, we apply the Lemma 3 to check the validation. When we are given original data T and privacy requirement F', if we want to know whether there is a possible data assignment for the bucket setting. We only need to check the occurrence of each sensitive value whether it is no more than the $\lfloor f'_i S \rfloor b$. If so, it means we have a validated bucket setting $B(b, S)$ for the records assignment. Let us apply the validation checking to the Example 6.

Example 7. *In Example 6, we used the bucket setting $B(b = 6, S = 6)$. We will check the validation condition for each sensitive value:*

- x_1-x_3*:* $o_i = 2$*,* $f_i = 0.0056$ *and* $f'_i = 0.187$*. So* $o_i = 2 < \lfloor f'_i S \rfloor b = \lfloor 0.187 \times 6 \rfloor \times 6 = 6$ *holds.*
- x_4-x_7*:* $o_i = 4$*,* $f_i = 0.11$ *and* $f'_i = 0.353$*. So* $o_i = 4 < \lfloor 0.353 \times 6 \rfloor \times 6 = 12$ *holds.*
- x_8-x_9*:* $o_i = 7$*,* $f_i = 0.19$ *and* $f'_i = 0.603$*. So* $o_i = 7 < \lfloor 0.603 \times 6 \rfloor \times 6 = 18$ *holds.*

Therefore, the bucket setting $b = 6, S = 6$ is valid.

We now check whether another bucket setting is valid. We take the bucket setting $B(b = 9, S = 4)$. In this case, we have:

- x_1-x_3*:* $o_i = 2$*,* $f_i = 0.0056$ ***and*** $f'_i = 0.187$*.* ***So*** $o_i = 2 < \lfloor f'_i S \rfloor b = \lfloor 0.187 \times 4 \rfloor \times 9 = 0$ ***does not hold.***
- x_4-x_7*:* $o_i = 4$*,* $f_i = 0.11$ ***and*** $f'_i = 0.353$*.* ***So*** $o_i = 4 < \lfloor 0.353 \times 4 \rfloor \times 9 = 9$ ***holds.***
- x_8-x_9*:* $o_i = 7$*,* $f_i = 0.19$ ***and*** $f'_i = 0.603$*.* ***So*** $o_i = 7 < \lfloor 0.603 \times 4 \rfloor \times 9 = 18$ ***holds.***

Therefore, the bucket setting $b = 9, S = 4$ *is not valid.*

Lemma 3 is the foundation of validation of two-sized bucketing and multiple-sized bucketing discussed later.

Chapter 5

Two-sized Bucketing

In this chapter, we treat the case of two-sized bucketing. We first state the validation condition. Then we propose the recording partition algorithm for two-sized bucketing. To improve the efficiency of the two-sized bucketing, we also implement the two pruning algorithms.

5.1 Validity Checking

We consider a two-sized bucket setting where the buckets have two different sizes: b_1 buckets of size S_1 and b_2 buckets of size S_2, where $S_2 > S_1$ and $b_j \geq 0$. Let B_j denote the set of buckets of size S_j and let $|B_j| = b_j S_j$, $j = 1, 2$. As before, a solution requires that all bucket slots are filled and the frequency of any value in each bucket is no more than its threshold f'. If such solution exists, we denote as $Valid(B_1 \cup B_2, T, F') =$ "Y". We consider the following two problems. In the *validity checking* problem, given B_1, B_2, T, F', we want to know whether $Valid(B_1 \cup B_2, T, F') =$ "Y". In the *bucket generation* problem, we assume $Valid(B_1 \cup B_2, T, F') =$ "Y" and we want to assign the records in T to the buckets in B_1 and B_2.

Given a table T, B_1, and B_2, where B_j is a set of b_j buckets of size S_j, $j = 1, 2$, and privacy constraint F', it is easy to see that $Valid(B_1 \cup B_2, T, F') =$ "Y" if and only if there exists a partition of T denoted by $\{T_1, T_2\}$, such that $Valid(B_1, T_1, F') =$ "Y" and $Valid(B_2, T_2, F') =$ "Y". We are going to use this observation to test whether $Valid(B_1 \cup B_2, T, F') =$ "Y".

For a given table T, recall that o_i is the number of records for a value x_i. If we specify the F'- privacy requirement, it follows from Lemma 3, for each x_i, the set B_j contains no more than $u_{ij} = \lfloor f'_i S_j \rfloor b_j$ records in it, $j = 1, 2$. This is the theoretical upper bound on the number of records for x_i can be allocated to B_j without violating the f'_i constraint assuming unlimited supply of x_i records. We now define $a_{ij} = min\{u_{ij}, o_i\}$, which is the practical bound limited by the actual supply of x_i records. In the following theorem, we give the checkable necessary and sufficient conditions for the existence of a solution for $(B_1 \cup B_2, T, F')$.

Theorem 1. *(Validating Two-sized Bucket Setting) Given a table T, B_1, B_2 and the privacy constraint F', where B_j is a set of S_j-sized b_j buckets and $S_1 < S_2$, $Valid(B_1 \cup B_2, T, F') =$ "Y" if and only if all of the following relations hold.*

$$\forall i : a_{i1} + a_{i2} \geq o_i \quad (Privacy\ Constraint(PC)) \tag{5.1}$$

$$j = 1, 2 : \sum_i a_{ij} \geq |B_j| \qquad (Fill\ Constraint(FC)) \tag{5.2}$$

$$|T| = |B_1| + |B_2| \quad (Capacity\ Constraint(CC)) \tag{5.3}$$

Inequality (5.1) is the "no overflow rule": the occurrence of x_i does not exceed the theoretical bound imposed by the F'-privacy. Inequality (5.2) is the "fill up rule": it is possible to fill up B_j without violating the F'-privacy. Equation (5.3) says that the buckets have the right total capacity for T.

Proof. Intuitively, relation (5.1) says that the number of occurrence of x_i does not exceed the upper bound $a_{i1} + a_{i2}$ imposed by F'-privacy on all buckets collectively, that is, the Privacy Constraint. Equation (5.2) says that under this upper bound constraint it is possible to fill up the buckets in B_j without leaving unused slots, that is, the Fill Constraint. Equation (5.3) says that the total bucket capacity matches the data cardinality, namely Capacity Constraint. Clearly, all these conditions are necessary for a valid assignment. This completes the proof for "only if". The proof for "if" part is given by the algorithm in Section 5.2 to find a valid assignment of the records in T to the buckets in B_1 and B_2 under the assumption of (5.1)-(5.3). □

5.2 Record Partitioning

Suppose that PC, FC and CC in Theorem 1 hold. We are going to show how to find a partition $\{T_1, T_2\}$ of T such that $Valid(B_1, T_1, F') = true$ and $Valid(B_2, T_2, F') = true$. This provides the sufficiency proof for Theorem 1 since this leads naturally to $Valid(B_1 \cup B_2, T, F') = true$. By finding the partition $\{T_1, T_2\}$, we also provide an algorithm for assigning records from T to the buckets in $B_1 \cup B_2$, that is, simply applying RRB to each of (T_j, B_j), $j = 1, 2$. The partition $\{T_1, T_2\}$ we look for must be such that the condition in Lemma 3 holds for each of (B_1, T_1, F') and (B_2, T_2, F'). The partition $\{T_1, T_2\}$ can be created as follows.

First, we consider distributing records to the buckets in B_1. Let T_1 contain any a_{i1} records with the value x_i for each x_i. Inequality (5.2) implies $|T_1| \geq |B_1|$. If we move out any $|T_1| - |B_1|$ records from T_1, the resulting T_1 satisfies inequality (4.2) in Lemma 3.

Next, let T_2 contain all records in $T - T_1$. Inequality (5.1) implies $o_i - a_{i1} \leq a_{i2} \leq u_{i2}$ for all x_i, that is, T_2 contains no more records of x_i than the maximum number imposed by the F'-privacy. If $|T_1| = |B_1|$, we are done.

PopulatingBuckets(T, B_1, B_2)
Input: T, B_1, B_2
Output: the set of records T_1 for B_1 and the set of records T_2 for B_2

```
1: let T_1 contain any a_{i1} records from T for each x_i, and let T_2 be T - T_1
2: let x'_1, ..., x'_m, x'_{m+1} be the longest prefix of the list such that \sum_{i=1}^{m}(a_{i2} - n_{i2}) <= |B_2| - |T_2| and
   \sum_{i=1}^{m+1}(a_{i2} - n_{i2}) > |B_2| - |T_2|
3: let w = |B_2| - |T_2| - \sum_{i=1}^{m}(a_{i2} - n_{i2}).
4: for all 1 <= i <= m do
5:    move (a_{i2} - n_{i2}) records of x'_i from T_1 to T_2
6: end for
7: move w records of x'_{m+1} from T_1 to T_2
8: return T_1 and T_2
```

Program 5.1: Algorithm of Determining Records for B_1 and B_2

Assume that $|T_1| > |B_1|$. Then we have $|T_2| < |B_2|$ according to Equation (5.3). From Inequality (5.2) there must be some x_i for which less than a_{i2} records are found in T_2. For any such x_i value, we move records of x_i from T_1 to T_2 until the number of records for x_i in T_2 reaches the maximum a_{i2} or $|T_2| = |B_2|$, whichever comes first. As long as $|T_2| < |B_2|$, relation(5.2) implies that some x_i have less than a_{i2} records in T_2. We can move records for such x_i from T_1 to T_2 without violating the F'-privacy. Eventually, $|T_2| = |B_2|$. Such T_1 and T_2 form the required partition of T.

The Program 5.1 provides the method of partition T_1 and T_2. At Line 2, $x'_1, \cdots, x'_m, x'_{m+1}$ is the longest prefix of the list such that:

$$\sum_{i=1}^{m}(a_{i2} - n_{i2}) \leq |B_2| - |T_2|$$
$$\sum_{i=1}^{m+1}(a_{i2} - n_{i2}) > |B_2| - |T_2|$$

Let $w = |B_2| - |T_2| - \sum_{i=1}^{m}(a_{i2} - n_{i2})$. For $1 \leq i \leq m$, we move $(a_{i2} - n_{i2})$ of x'_i from T_1 to T_2, and move w records of x'_{m+1} from T_1 to T_2.

Assume that a_{i1}, a_{i2}, o_i have been collected. Step 1 takes one scan of T. Step 2 and 3 take $m \log m$, where m is the number of distinct values. Step 5 - 7 take one scan of T because each record is moved at most once. In fact, the algorithm can be made more efficient by only keeping the number of records for each x_i in all tables T, T_1 and T_2. This is because the privacy constraint depends only on the number of records of x_i, not on the actual records. With this optimization, Step 1 and Step 5 - 7 take a time proportional to the number of distinct values, m. Thus the complexity of the algorithm is $m \log m$.

Example 8. *We are given a table containing 50 tuples with o_i given in the Table 5.1. Suppose that the F'-privacy is given by $f'_i = 2f_i + 0.05$. Consider the bucket setting $B_1(S_1 = 4, b_1 = 9), B_2(S_2 =$*

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
o_i	1	1	1	1	1	1	1	1	6	6	6	6	9	9
f_i	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.12	0.12	0.12	0.12	0.18	0.18
f_i'	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.29	0.29	0.29	0.29	0.41	0.41

Table 5.1: Data Set of Example 8

(a) The bucket for B_1

x_9	x_{10}	x_{12}	x_{13}
x_9	x_{11}	x_{12}	x_{14}
x_9	x_{11}	x_{13}	x_{14}
x_9	x_{11}	x_{13}	x_{14}
x_9	x_{11}	x_{13}	x_{14}
x_{10}	x_{11}	x_{13}	x_{14}
x_{10}	x_{12}	x_{13}	x_{14}
x_{10}	x_{12}	x_{13}	x_{14}
x_{10}	x_{12}	x_{13}	x_{14}

(b) The buckets for B_2

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}

Table 5.2: Record Assignment of Example 8

$14, b_2 = 1)$. Note CC in Theorem 1 holds. Let us compute a_{i1} and a_{i2}. Suppose there are $b_1 = 9$ buckets of sizes $S_1 = 4$ and $b_2 = 1$ buckets of size $S_2 = 14$. $|B_1| = 36$ and $|B_2| = 14$. We apply Theorem 1 to verify that there is a solution for $(B_1 \cup B_2, T, F')$. It is easy to see that equality (5.3) holds. To verify inequality (5.1), we compute a_{ij} for x_i and B_j. Recall that $a_{i1} = min\{u_{ij}, o_i\}$ and $a_{i2} = min\{u_{i2}, o_i\}$. We have

- *x_1-x_8: $o_i = 1$, $f_i = 0.02$ and $f_i' = 0.09$. So $u_{i1} = \lfloor f_i' S_1 \rfloor b_1 = \lfloor 0.09 \times 4 \rfloor \times 9 = 0$, $a_{i1} = 0$.*
- *x_9-x_{12}: $o_i = 6$, $f_i = 0.12$ and $f_i' = 0.29$. So $u_{i1} = \lfloor 0.29 \times 4 \rfloor \times 9 = 9$, $a_{i1} = 6$.*
- *x_{13}-x_{14}: $o_i = 9$, $f_i = 0.18$ and $f_i' = 0.41$. So $u_{i1} = \lfloor 0.41 \times 4 \rfloor \times 9 = 9$, $a_{i1} = 9$.*
- *x_1-x_8, $u_{i2} = \lfloor f_i' S_2 \rfloor b_2 = \lfloor 0.09 \times 14 \rfloor \times 1 = 1$, $a_{i2} = 1$.*
- *x_9-x_{12}, $u_{i2} = \lfloor 0.29 \times 14 \rfloor \times 1 = 4$, $a_{i2} = 4$.*
- *x_{13}-x_{14}, $u_{i2} = \lfloor 0.41 \times 14 \rfloor \times 1 = 5$, $a_{i2} = 5$.*

It can be verified that PC and FC in Theorem 1 hold. To find the partitioning $\{T_1, T_2\}$, initially T_1 contains $a_{i1} = 0$ record for each of x_1-x_8, $a_{i1} = 6$ records for each of x_9-x_{12}, and $a_{i1} = 9$ records for each of x_{13}-x_{14}. T_2 contains the remaining records in T. Since T_1 contains 42 records, but size of B_1 is 36, we need to move 6 records from T_1 to T_2 without exceeding the upper bound a_{i2} for T_2.

This can be done by moving one record for each of $x_9 - x_{14}$ from T_1 to T_2. Table 5.2 shows a record assignment generated by RRB for (B_1, T_1) and (B_2, T_2).

After we introduce the validation condition and records partition for two-sized bucketing, we will introduce the pruning algorithms for two-sized bucketing in next section.

5.3 Pruning Algorithm for Two-sized Bucketing

In this section, we tackle the problem of finding the optimal bucket setting b_1, b_2, S_1, S_2 (or simply say B_1, B_2) using the notation in Section 3.5 such that the loss function defined in Section 3.4 is minimized for given T and F'-privacy under the requirement: $b_1S_1 + b_2S_2 = |T|$. A naive algorithm is to enumerate all (b_1, b_2, S_1, S_2) and apply Theorem 1 to test if a solution exists, and return the solution that gives the minimum of the loss function. This algorithm is not efficient because both b_1 and b_2 can take a large range and S_1, S_2 related to the F'-privacy can be very small. For example, for the real data set of CENSUS, which we will use in the Chapter 7, there are 500 thousand records. If we treat the occupation as the sensitive attribute and we think $f' = 0.05$ can satisfy our privacy protection requirement of the most sensitive value, then the largest bucket size constrained by the privacy will be 20. There are also other bucket sizes smaller than 20. As we can see, b_i is very large, around $500,000/20 = 25,000$. If we check all the possible setting one by one, it takes a lot of time. Thus we propose a more efficient algorithm in later section.

Let M and M' denote the minimum and maximum possible bucket sizes constrained by the privacy constraints, that is, $M \leq S_1 < S_2 \leq M'$. The lower bound M can be determined by the largest f'_i, i.e., $M = \min_i\{\lceil 1/f'_i \rceil\}$. The upper bound M' can be determined by the maximum allowable loss of a record. We have $M' \geq \max_i\{\lceil 1/f'_i \rceil\}$ if there is a solution. We will assume that M' is given. In general, M' is a relatively small value because a large M' leads to too much information loss. Note that the valid bucket setting may not exist in the range of bucket size.

5.3.1 Indexing Bucket Setting

Our strategy is to enumerate all pairs (S_1, S_2) such that $M \leq S_1 < S_2 \leq M'$. For each pair (S_1, S_2), we search for the pair (b_1, b_2) such that

$$|T| = b_1S_1 + b_2S_2 \tag{5.4}$$

and the loss metric defined by Equation (3.2) is minimized. In this case, we explicitly write out the loss metric

$$Loss(b_1, b_2, S_1, S_2) = Loss(B_1 \cup B_2) = b_1(S_1 - 1)^2 + b_2(S_2 - 1)^2. \tag{5.5}$$

We call such setting $\{b_1, b_2, S_1, S_2\}$ the optimal setting. For a given pair (S_1, S_2), there are many possible pairs (b_1, b_2). We need to organize all pairs (b_1, b_2) and to prune the ones that do not satisfy (5.4) or do not minimize the function (5.5) as much as possible. To do so, we introduce index to all such pairs.

Considering the sizes S_1 and S_2, a pair (b_1, b_2) is feasible if satisfying (5.4). As we have discussed in Section 5.1, the setting (b_1, b_2, S_1, S_2) is valid means there is a solution satisfies the privacy requirement. A valid pair will be kept if function (5.5) is smallest among all valid pairs in the index of (S_1, S_2) since it can be used to generate the optimal setting.

Let us sort all feasible pairs (b_1, b_2) as follows: (b_1, b_2) *precedes* (b_1', b_2') if $b_1 > b_1'$; in this case, $b_2 < b_2'$ because $b_1S_1 + b_2S_2 = b_1'S_1 + b_2'S_2$. We define $\Gamma(S_1, S_2)$ to be the sorted list of feasible pairs for S_1 and S_2. Below, we will show that the i-th pair in $\Gamma(S_1, S_2)$ can be generated directly using the position without scanning the list. We will use this property to locate all valid pairs by binary search on the index without storing the list.

The first pair in the index $\Gamma(S_1, S_2)$ is denoted by (b_1^0, b_2^0). The setting (b_1^0, b_2^0) must satisfy the requirement (5.4) and thus b_2^0 is the smallest. In a way (b_1^0, b_2^0) is the solution to $\min\{v_2\}$ subject to $S_1v_1 + S_2v_2 = |T|$, where v_1 and v_2 are variables of positive integers. According to this index, the earlier pair of (b_1, b_2) get more buckets with small sizes, it means, a smaller Loss than the latter pair. So the first feasible pair is an optimal solution.

Let (b_1, b_2) be a feasible pair and let $(b_1^0 - \Delta_1, b_2^0 + \Delta_2)$ be the next feasible pair. Note that

$$S_1(b_1^0 - \Delta_1) + S_2(b_2^0 + \Delta_2) = |T|$$

and

$$S_1b_1^0 + S_2b_2^0 = |T|$$

These equalities imply $S_1\Delta_1 = S_2\Delta_2$. Note that both Δ_1 and Δ_2 must be integers. To meet this constraint, $S_2\Delta_2$ must be a common multiple of S_1 and S_2. Since we want the smallest Δ_2, $S_2\Delta_2$ must be the least common multiple of S_1 and S_2, denoted by $LCM(S_1, S_2)$. Then Δ_2 and the corresponding Δ_1 are given by

$$\Delta_2 = LCM(S_1, S_2)/S_2; \qquad \Delta_1 = LCM(S_1, S_2)/S_1. \tag{5.6}$$

Hence $\Gamma(S_1, S_2)$ has the form:

(b_1^0, b_2^0)
$(b_1^0 - \Delta_1, b_2^0 + \Delta_2)$
...
$(b_1^0 - k\Delta_1, b_2^0 + k\Delta_2)$

Here k can be determined by the minimum number of S_1-sized buckets. Since the decrement step for b_1 is Δ_1, the minimum number of S_1-sized buckets must be smaller than Δ_1, so $0 \leq (b_1^0 - k\Delta_1) < \Delta_1$. This gives rise to $b_1^0/\Delta_1 - 1 < k \leq b_1^0/\Delta_1$. The only integer k satisfying this condition is $\lfloor b_1^0/\Delta_1 \rfloor$.

Example 9. *Let $|T| = 28, S_1 = 2, S_2 = 4$. Then we have $LCM(S_1, S_2) = 4$, $\Delta_2 = 4/4 = 1$ and $\Delta_1 = 4/2 = 2$. Therefore, we get $b_1^0 = 14, b_2^0 = 0$ and $k = \lfloor 14/2 \rfloor = 7$. So the sorted list of feasible pairs for the given S_1 and S_2 is*

$$\Gamma(S_1, S_2) = [(14, 0), (12, 1), (10, 2), (8, 3), (6, 4), (4, 5), (2, 6), (0, 7)].$$

Suppose $S_1 = 3$ and $S_2 = 5$. Then we have $\Delta_1 = 5$, $\Delta_2 = 3$, $b_1^0 = 1, b_2^0 = 5$ and $k = \lfloor 1/5 \rfloor = 0$. So $\Gamma(S_1, S_2) = [(1, 5)]$.

Remark 1. *$\Gamma(S_1, S_2)$ has several important properties for dealing with a large data set. Firstly, we can access the i-th element of $\Gamma(S_1, S_2)$ without storing or scanning the list. Secondly, we can represent any sublist of $\Gamma(S_1, S_2)$ by the parameter k. To clarify it, an interval $[i, j]$ denotes a sublist, where i is the starting position and j is the ending position of the sublist. For example, the interval $[1, 2]$ denotes the list of two elements: $(b_1^0 - \Delta_1, b_2^0 + \Delta_2)$, $(b_1^0 - 2\Delta_1, b_2^0 + 2\Delta_2)$. Thirdly, the common sublist of two sublists L and L' of $\Gamma(S_1, S_2)$, denoted by $L \cap L'$, is given by the intersection of the intervals of L and L'.*

Note that there is no need to explicitly materialize the list $\Gamma(S_1, S_2)$ because we have known that all we need is the first pair (b_1^0, b_2^0), Δ_1, Δ_2 and k. Sometimes the list can be very long. For example, we have data set of 300k records. The assumed bucket sizes are same as it in Example 9, namely, $S_1 = 2$ and $S_2 = 4$. So we get $LCM(S_1, S_2) = 4$, $\Delta_1 = 2$ and $b_1^0 = 150k$. Then the length of possible setting (b_1, b_2) is as long as $b_1^0/\Delta_1 = 75k$. So a linear scan is not efficient. Ideally we want to examine as few feasible (b_1, b_2) as possible. Thus we explore two pruning strategies: one based on the loss minimization and the other one based on the privacy requirement:

5.3.2 Loss-Based Pruning

Our first strategy is to prune the pairs in $\Gamma(S_1, S_2)$ that do not have the minimum loss with respect to (S_1, S_2) by exploiting the following monotonicity of the Loss metric (5.5).

Lemma 4 (Monotonicity of loss). *If (b_1, b_2) precedes (b_1', b_2') in $\Gamma(S_1, S_2)$, then*

$$Loss(B_1 \cup B_2) < Loss(B_1' \cup B_2'),$$

where set B_j contains b_j buckets of size S_j, and set B_j' contains b_j' buckets of size S_j, $j = 1, 2$.

Proof. Because $S_1 < S_2$, the total number of records $|T|$ is fixed, $|T| = b_1 S_1 + b_2 S_2$, it is obvious that the Loss metric defined by (5.5) is smaller when b_1 is larger. It means we get more buckets with small sizes and less buckets with big sizes. According to the definition of the index, b_1 is larger than b_1' if (b_1, b_2) precedes (b_1', b_2') and this implies the statement. □

From Lemma 4, for a given pair (S_1, S_2), if we examine $\Gamma(S_1, S_2)$ sequentially, and if a feasible setting is found for the first time, this setting must be optimal for (S_1, S_2). Lemma 4 can also be exploited to prune pairs across different (S_1, S_2). Let $Best_{loss}$ be the minimum loss found so far and (S_1, S_2) be the next pair of bucket sizes to be considered. From Lemma 4, all the pairs in $\Gamma(S_1, S_2)$ that have a loss less than $Best_{loss}$ must form a prefix of $\Gamma(S_1, S_2)$. Let k' be the cutoff point of this prefix, where $b_1' = b_1^0 - k'\Delta_1$ and $b_2' = b_2^0 + k'\Delta_2$. The integer k' is the maximum integer satisfying $b_1'(S_1-1)^2 + b_2'(S_2-1)^2 < Best_{loss}$, given by

$$k' = \max\{0, \lfloor \frac{Best_{loss} - b_1^0 S_1 - 1)^2 - b_2^0(S_2-1)^2}{\Delta_2(S_2-1)^2 - \Delta_1(S_1-1)^2} \rfloor\} \tag{5.7}$$

Let $\Gamma(k')$ denotes the prefix of $\Gamma(S_1, S_2)$ that contains the first $k'+1$ pairs. It is sufficient to consider this list instead of the whole list of feasible pairs. This method is called loss-based pruning. The pruning algorithm revises the sorted list of feasible pairs by the cut off point based on $Best_{loss}$.

The algorithm for finding the optimal (b_1, b_2, S_1, S_2) is given in Program 5.2. Recall that M and M' denote the minimum and maximum bucket sizes, respectively. The algorithm consists of a nested loop that enumerates all pairs (S_1, S_2) within the range of $M \leq S_1 < S_2 \leq M'$ in Lines 3 and 4. For each pair (S_1, S_2), it goes through the ordered list of all feasible pairs with the first pair being (b_1^0, b_2^0) and the step sizes Δ_1 and Δ_2 determined by Equation (5.6). In Line 10, we get the k' according to Equation (5.7). Using the loss based pruning, we significantly shorten the list we need to check. If $Valid(B_1 \cup B_2, T, F') =$ "Y" as shown in Line 12, it updates the best solution so far, if necessary, and terminates the search for the current S_1 and S_2 (by letting $k' = -1$). After examining all pairs of S_1, S_2, it returns the best setting $Best_{setting}$.

5.3.3 Privacy-Based Pruning

In previous subsection we can see that the loss minimization pruning reduces the length of the list $\Gamma(k')$ by the Loss-based pruning. We still need to search the list sequentially until the first valid pair is found. Our second strategy is to identify the first valid pair in $\Gamma(k')$ *directly* by exploiting a certain monotonicity property of privacy constraints.

Definition 9. *We say that a property P is* monotone *on a sorted list if whenever P holds for (b_1, b_2), it holds for (b_1', b_2'), where (b_1, b_2) proceeds (b_1', b_2') in Γ, and* anti-monotone *on Γ if whenever P fails on (b_1, b_2), it fails on (b_1', b_2'), where (b_1, b_2) proceeds (b_1', b_2') in Γ.*

Importantly, such a monotonicity divides the sorted list Γ into two sublists such that P holds in one of them and fails in the other, and the splitting point of the two sublists can be found by performing a binary search on Γ. In the rest of this section, we show that Equations (5.2) and (5.1) are monotone or anti-monotone, and use such properties to identify the exact location of the valid pairs in $\Gamma(k')$.

Two-Sized Loss-Based Bucketing(T, F', M, M')
Input: T, F'-privacy, minimum and maximum bucket sizes M, M'
Output: optimal (b_1, b_2, S_1, S_2)

$Best_{loss} \leftarrow \lceil \frac{|T|}{M'} \rceil (M'-1)^2$
$Best_{setting} \leftarrow NULL$
for all $S_1 \in \{M, \cdots, M'-1\}$ **do**
  for all $S_2 \in \{S_1+1, \cdots, M'\}$ such that $\Gamma(S_1, S_2)$ is not empty **do**
    $(b_1^0, b_2^0) \leftarrow$ the first feasible pair in $\Gamma(S_1, S_2)$
    $\Delta_2 \leftarrow LCM(S_1, S_2)/S_2$
    $\Delta_1 \leftarrow LCM(S_1, S_2)/S_1$
    $b_1 \leftarrow b_1^0$
    $b_2 \leftarrow b_2^0$
    k' is defined in Equation (5.7)
    repeat
      if $Valid(B_1 \cup B_2, T, F') = "Y"$ **then**
        if $Best_{loss} > Loss(b_1, b_2, S_1, S_2)$ **then**
          $Best_{setting} \leftarrow (b_1, b_2, S_1, S_2)$
          $Best_{loss} \leftarrow Loss(b_1, b_2, S_1, S_2)$
        end if
        $k' \leftarrow -1$
      else
        $b_1 \leftarrow b_1 - \Delta_1$
        $b_2 \leftarrow b_2 + \Delta_2$
        $k' \leftarrow k' - 1$
      end if
    until $k' < 0$
  end for
end for
return $Best_{setting}$

Program 5.2: Two-sized Loss-Based Pruning Algorithm

Lemma 5. *Equation (5.2) is monotone on $\Gamma(k')$ for $j = 1$ and anti-monotone on $\Gamma(k')$ for $j = 2$.*

Proof. We write out Equation (5.2) for $j = 1$ and $j = 2$ explicitly.

$$\sum_i \min_i\{\lfloor f'_i S_1 \rfloor b_1, o_i\} \geq S_1 b_1 \tag{5.8}$$

$$\sum_i \min_i\{\lfloor f'_i S_2 \rfloor b_2, o_i\} \geq S_2 b_2 \tag{5.9}$$

Consider two pairs (b_1, b_2) and (b'_1, b'_2) on $\Gamma(k')$, where (b_1, b_2) proceeds (b'_1, b'_2), that is, $b_1 > b'_1$ and $b_2 < b'_2$. As b_1 decreases to b'_1, both $\lfloor f'_i S_1 \rfloor b_1$ and $S_1 b_1$ decreases by a factor by b'_1/b_1, but o_i remains unchanged. Therefore, if Equation (5.8) holds for (b_1, b_2), it holds for (b'_1, b'_2). Then Equation (5.8) is monotone on $\Gamma(k')$. For a similar reason, if Equation (5.9) fails on (b_1, b_2), it remains to fail on (b'_1, b'_2); thus Equation (5.9) is anti-monotone on $\Gamma(k')$. □

In order to clearly describe our algorithm below, we introduce the following notations.

Notation 1. *Equation (5.8) divides $\Gamma(k')$ into two sublists Λ_1^- and Λ_1^+. Here Λ_1^- donates the sublist containing all pairs not satisfying the equation and Λ_1^+ donates the sublist containing all pairs satisfying the equation. Similarly, Equation (5.9) also divides $\Gamma(k')$ into two sublists Λ_2^+ and Λ_2^-, where Λ_2^+ donates the sublist containing all pairs satisfying the equation and Λ_2^- donates the sublist containing all pairs not satisfying the equation.*

It is possible that one of Λ_1^- and Λ_1^+ or one of Λ_2^- and Λ_2^+ may be empty. According to Notation 1, $\Lambda_2^+ \cap \Lambda_1^+$ contains exactly the pairs that satisfy both Equation (5.8) and Equation (5.9) (order preserved).

Example 10. *We are given the data set $|T| = 28$, the privacy requirement is $f'_i = 2f_i$. The distribution of sensitive attribute x_i is shown in Table 5.3. We focus the possible buckets set $S_1 = 2$, $S_2 = 4$. As shown in the Example 8, The feasible pair (b_1, b_2) is*

$$\Gamma(S_1, S_2) = [(14, 0), (12, 1), (10, 2), (8, 3), (6, 4), (4, 5), (2, 6), (0, 7)].$$

	x_1	x_2	x_3	x_4	x_5
o_i	4	4	4	8	8
f_i	0.1428	0.1428	0.1428	0.286	0.286
f'_i	0.286	0.286	0.286	0.57	0.57

Table 5.3: Data Set of Example 10

Applying the Equation (5.8) to the data set, we get the fill constrain for the first kind of bucket:

$$3\min\{\lfloor 0.286 \times 2 \rfloor b_1, 4\} + 2\min\{\lfloor 0.57 \times 2 \rfloor b_1, 8\} \geq 2b_1,$$

so we get simple version of fill constrain:

$$\min\{b_1, 8\} \geq b_1.$$

It means when $b_1 \leq 8$, the Equation (5.9) holds. Otherwise, when $b_1 > 8$, the equation does not hold. Thus

$$\Lambda_1^+ = [(8,3),(6,4),(4,5),(2,6),(0,7)],$$

$$\Lambda_1^- = [(14,0),(12,1),(10,2].$$

Using the Equation (5.9), we get the fill constrain for the second kind of bucket

$$3\min\{\lfloor 0.286 \times 4 \rfloor b_2, 4\} + 2\min\{\lfloor 0.57 \times 4 \rfloor b_2, 8\} \geq 4b_2,$$

then we get when $b_2 > 7$, the equation does not hold. It means Λ_2^+ is the whole list, the Λ_2^- is empty.

The monotonicity of Equation (5.8) and the anti-monotonicity of Equation (5.9) imply that we can find Λ_1^- and Λ_1^+ by a binary search on $\Gamma(k')$, and find Λ_2^+ and Λ_2^- by a binary search on $\Gamma(k')$. These binary searches take $O(\log k')$ evaluations of Equations (5.8) and (5.9).

We will do the similar analysis for Equation (5.1). For each value x_i, we rewrite it as:

$$\min\{\lfloor f_i' S_1 \rfloor b_1, o_i\} + \min\{\lfloor f_i' S_2 \rfloor b_2, o_i\} \geq o_i. \tag{5.10}$$

Observe that, as we scan the list $\Gamma(k')$, b_1 decreases and b_2 increases. Therefore, for each x_i, the first term on the left side in Equation (5.10) divides $\Gamma(k')$ into two sublists: Ω_{1i}^+ contains all pairs satisfying $\lfloor f_i' S_1 \rfloor b_1 \geq o_i$, and Ω_{1i}^- contains all pairs satisfying $\lfloor f_i' S_1 \rfloor b_1 < o_i$. One of these two sublists may be empty. Similarly, the second term on the left side in Equation (5.10) divides $\Gamma(k')$ into two sublists: Ω_{2i}^- contains all pairs satisfying $\lfloor f_i' S_2 \rfloor b_2 < o_i$, and Ω_{2i}^+ contains all pairs satisfying $\lfloor f_i' S_2 \rfloor b_2 \geq o_i$.

Example 11. *We will use the same data set of Example 9. In the case of $i = 1$, we get*

$$\lfloor f_1' S_1 \rfloor b_1 = \lfloor 0.286 \times 2 \rfloor b_1 = 0 * b_1 = 0 < o_1 = 4.$$

It implies Ω_{11}^+ is empty and Ω_{11}^- is the whole list.
In this case, we also have:

$$\lfloor f_1' S_2 \rfloor b_2 = \lfloor 0.286 \times 4 \rfloor b_2 = b_2 < o_1 = 4.$$

It leads to $b_2 < 4$. Thus, we get

$$\Omega_{21}^- = [(14,0),(12,1),(10,2),(8,3)]$$

$$\Omega_{21}^+ = [(6,4),(4,5),(2,6),(0,7)].$$

Equation (5.10) holds for all pairs in either Ω_{1i}^+ or Ω_{2i}^+. The remaining part of $\Gamma(k')$ is the intersection of $\Omega_{2i}^- \cap \Omega_{1i}^-$. Below we assume that this part is not empty. For the pairs in this list, Equation (5.10) degenerates into

$$\lfloor f_i' S_1 \rfloor b_1 + \lfloor f_i' S_2 \rfloor b_2 \geq o_i \tag{5.11}$$

Consider two consecutive pairs (b_1, b_2) and $(b_1 - \Delta_1, b_2 + \Delta_2)$ in $\Omega_{2i}^- \cap \Omega_{1i}^-$. If the following condition holds

$$\lfloor f_i' S_2 \rfloor \Delta_2 \geq \lfloor f_i' S_1 \rfloor \Delta_1 \tag{5.12}$$

then Equation (5.11) holding for (b_1, b_2) implies it holds for $(b_1 - \Delta_1, b_2 + \Delta_2)$. If Equation (5.12) fails, then Equation (5.11) failing for (b_1, b_2) implies that it fails for $(b_1 - \Delta_1, b_2 + \Delta_2)$. The next lemma summarizes these observations.

Lemma 6. *If Equation (5.12) holds, Equation (5.10) is monotone on $\Omega_{2i}^- \cap \Omega_{1i}^-$; otherwise, Equation (5.10) is anti-monotone on $\Omega_{2i}^- \cap \Omega_{1i}^-$.*

Let Ω_{12i}^- denote the sublist of $\Omega_{2i}^- \cap \Omega_{1i}^-$ such that Equation (5.11) holds for its elements. Let Ω_i denote the part in $\Gamma(k')$ in which Equation (5.10) holds for each x_i. The next lemma, which follows from Lemma 6 and the above discussion, summarizes the location of Ω_i.

Lemma 7. *The sublist Ω_i consists of the prefix Ω_{1i}^+ and the suffix $\Omega_{12i}^- \cup \Omega_{2i}^+$ of $\Gamma(k')$; if Equation (5.12) fails, Ω_i consists of the prefix $\Omega_{1i}^+ \cup \Omega_{12i}^-$ and the suffix Ω_{2i}^+ of $\Gamma(k')$.*

Let $\cap_i \Omega_i$ denote the intersection of Ω_i over all x_i. This intersection contains exactly the pairs satisfying Equation (5.10) for all x_i.

5.3.4 The Pruning Algorithm

We now present an efficient algorithm for the optimal two-size bucketing. Combining the loss-based pruning and privacy-based pruning, $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ contains exactly the valid pairs in $\Gamma(k')$, i.e., the pairs that satisfy all of Equations (5.1)-(5.3).

Theorem 2. *If the optimal bucket setting (b_1, b_2) for (S_1, S_2) exists, it is the first pair in $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$.*

To compute $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$, we observe that Λ_1^+, Λ_2^+ and Ω_i depend on Λ_2^+, Λ_1^+, Ω_{1i}^+, Ω_{1i}^-, Ω_{2i}^-, Ω_{2i}^+, all of which can be computed by a binary search on $\Gamma(k')$. Therefore, $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ can be computed in time $O(m \log k')$. Moreover, since binary search and intersection of sublists only deal with the ranges of sublists, instead of actual lists, this computation requires little space.

Theorem 3. *For each (S_1, S_2), $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ can be computed in time $O(m \log k')$ and in space $O(m)$, where k' is defined in Equation (5.7).*

2SizeBucketing(T, F', M, M')
Input: T, F'-privacy, minimum and maximum bucket sizes M, M'
Output: optimal bucket setting (b_1, b_2, S_1, S_2)

1: $Best_{loss} \leftarrow \lceil |T|/M' \rceil (M'-1)^2$
2: $Best_{setting} \leftarrow NULL$
3: **for all** $S_1 \in \{M, \cdots, M'-1\}$ **do**
4:     **for all** $S_2 \in \{S_1+1, \cdots, M'\}$ **do**
5:         compute $b_1^0, b_2^0, \Delta_1, \Delta_2, k'$
6:         compute k' using Equation (5.7)
7:         **if** $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ is not empty **then**
8:             let (b_1, b_2) be the first pair in it
9:             let B_j be the set of b_j buckets of size S_j, $j = 1, 2$
10:             **if** $Best_{loss} > Loss(B_1 \cup B_2)$ **then**
11:                 $Best_{setting} \leftarrow (b_1, b_2, S_1, S_2)$
12:                 $Best_{loss} \leftarrow Loss(B_1 \cup B_2)$
13:             **end if**
14:         **end if**
15:     **end for**
16: **end for**
17: return $Best_{setting}$

Program 5.3: Two-sized Full Pruning Algorithm

Program 5.3 finds the optimal bucket setting (b_1, b_2, S_1, S_2) based on Theorem 2. The input is a table T, a privacy parameter F', and the minimum and maximum bucket sizes M and M'. At Line 1 and 2, it initializes $Best_{loss}$ to an upper bound of loss obtained assuming that every bucket has the size M', i.e., $\lfloor |T|/M' \rfloor (M'-1)^2$, where $\lfloor |T|/M' \rfloor$ is the number of such buckets and $(M'-1)^2$ is the Information Loss of a bucket of size M'. Lines 3 and 4 enumerate all pairs (S_1, S_2) with $M \leq S_1 < S_2 \leq M'$. For each pair (S_1, S_2), Line 5 computes $b_1^0, b_2^0, \Delta_1, \Delta_2, k$ and Line 6 determines the length k' of the prefix $\Gamma(k')$ of $\Gamma(S_1, S_2)$ based on (S_1, S_2) and $Best_{loss}$. Line 7 computes $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ as discussed above. At Lines 8-12, if $\Lambda_2^+ \cap \Lambda_1^+ \cap (\cap_i \Omega_i)$ is not empty, the loss of the first pair in it is computed and is used to update $Best_{loss}$ if the loss is smaller than $Best_{loss}$.

Chapter 6

Multi-Sized Bucketing

In this chapter, we extend the solution for the two size problem to the multiple size problem. We first adjust the validation condition for the the multi-sized bucketing. Then we implement the top-down algorithm by recursively calling two-sized bucketing algorithm to an approximate solution for it.

6.1 Validity checking

To find the validation condition for multiple sized bucketing, we must first extend the Theorem 1 to validate a three size bucket setting. The Example 12 shows that a direct extension of Inequalities (5.1 - 5.3) does not work for multiple size problem.

Example 12. *Let $|B_1| = |B_2| = 20$, $|B_3| = 30$, and $|T| = 70$. There are 11 values in the SA: $x_1, \cdots, x_{11}$. For $1 \leq i \leq 10$, x_i has 5 records, and x_{11} has 20 records, i.e., $o_i = 5$ when $1 \leq i \leq 10$, and $o_{11} = 20$. Further suppose that for $1 \leq i \leq 10$, $u_{i1} = u_{i2} = 0$, $u_{i3} = 5$, and $u_{11,1} = u_{11,2} = u_{11,3} = 20$. We can see that the following inequalities hold:*

$$\forall i : a_{i1} + a_{i2} + a_{i3} \geq o_i$$

$$j = 1, 2, 3 : \sum_i a_{ij} \geq |B_j|$$

$$|T| = |B_1| + |B_2| + |B_3|$$

However, since $u_{i1} = u_{i2} = 0$, $1 \leq i \leq 10$, x_i cannot be assigned to B_1 and B_2, thus, 50 of them can only be assigned to B_3, which exceeds the capacity of B_3, 30. Therefore, there is no solution.

So we first propose an Integer Linear Programming for the multiple size bucketing problem. Then we introduce a top down algorithm.

We show that the generalized problem can be solved by integer linear programming. Let M and M' denote the minimum and maximum bucket sizes, and let $n = M' - M + 1$. We define $S_j = M + j - 1$, where $j = 1, \cdots, n$. Notice $S_1 = M$, $S_n = M'$, and $S_{j+1} = S_j + 1$ for $1 \leq j \leq n-1$. For $j = 1, \cdots, n$, let b_j be variables of non-negative integers representing the number of S_j-sized buckets, let B_j denote the set of such buckets, with $|B_j| = b_j S_j$. Let $x_1, \cdots, x_m$ be the sensitive values with the occurrence times $o_1, \cdots, o_m$ respectively. For $1 \leq i \leq m$ and $1 \leq j \leq n$, let v_{ij} denote the times of the occurrence of x_i in B_j. Let $u_{ij} = \lfloor f'_i S_j \rfloor b_j$, which is the maximum occurrence of x_i in B_j imposed by the F'-privacy, according to Lemma 3. To find a solution of the multiple-sized bucketing problem is to determine the values of v_{ij} and b_j such that the following constraints are satisfied:

- $\forall i : \sum_{j=1}^{n} v_{ij} = o_i$
- $\forall j : \sum_{i=1}^{m} v_{ij} = S_j b_j$
- $\forall i, j : v_{ij} \leq u_{ij}$
- $\sum_{j=1}^{n} S_j b_j = |T|$

v_{ij} and b_j are variables of non-negative integers and S_j, o_i, u_{ij} are constants. Our objective is

$$\min \sum_{j=1}^{n} b_j (S_j - 1)^2$$

The first constraint is the sum of the number of sensitive values in every buckets is exactly equal to the total number o_i. The second constraint is for each bucket with same size, the sum of quantity of different sensitive value is equal to the capacity of this kind of buckets. The third one is the limitation of privacy protection as we discussed in Lemma 3. The last constraint is about the total capability of all buckets of different size, it is equal to the total number of records.

This is an integer linear programming. Let $b_1, \cdots, b_n$ be the optimal bucket numbers found by solving the above programming problem. Let $b_{t1}, \cdots, b_{tk}$ be the subsequence of $b_1, \cdots, b_n$ such that $b_{tj} \neq 0$. We define B_j as the set of b_{tj} S_{tj}-sized buckets, $j = 1, \cdots, k$. For each value x_i, we assign v_{ij} records to B_j, $j = 1, \cdots, k$. Then we apply RRB to B_j to distribute the assigned records to each bucket in B_j.

6.2 Top-Down Algorithm

In general, integer linear programming problems are NP-hard, thus the above solution is not efficient for solving problems with a large m and n. We present an efficient heuristic algorithm by greedily and repeatedly applying the 2-size bucketing. The algorithm, $TopDownBucketing$, is described in

TDBucketing (T, B, F', M, M')

$(b_1, b_2, S_1, S_2) \leftarrow$ 2SizeBucketing($D, P, \min\_size, \max\_size$)
if $(b_1, b_2, S_1, S_2) \neq NULL$ **then**
  let B_i be a set of b_i S_i-sized buckets, $i = 1, 2$
  We apply data partitioning in the (D, B_1, B_2), we get (D_1, D_2)
  if $Loss(D_1, B_1) + Loss(D_2, B_2) < Loss(D, B)$ **then**
    TDBucketing(D_1, B_1, P, M, M')
    TDBucketing(D_2, B_2, P, M, M')
  else
    return(D, B)
  end if
else
  return(D, B)
end if

Program 6.1: Top-down Bucketing Algorithm

Program 6.1. It takes as the input a set of records D, a set of buckets of same size B, the privacy requirement F', the minimum and maximum bucket sizes $\min\_size$ and $\max\_size$. It applies the 2-sized bucketing algorithm to find the best 2-size buckets (B_1, B_2) for D. Let D_1 and D_2 be the sets of records in B_1 and B_2. If $Loss(D_1, B_1) + Loss(D_2, B_2) < Loss(D, B)$, recursively it applies the two size bucketing to (D_1, B_1) and (D_2, B_2) independently; otherwise, the current recursion terminates and returns the current (D, B).

Given the raw data T, the privacy requirement F', and the minimum and maximum bucket sizes M and M', let B be the single bucket containing all the records in T. The heuristic solution is given by calling the Top-down bucketing algorithm: $TDBucketing(T, B, F', M, M')$.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
o_i	2	2	2	2	4	4	4	4	7	8
f_i	0.051	0.051	0.051	0.051	0.102	0.102	0.102	0.102	0.179	0.205
f'_i	0.174	0.174	0.174	0.174	0.327	0.327	0.327	0.327	0.558	0.635

Table 6.1: Data Set of Example 13

Example 13. *We are given a table containing 39 tuples with o_i given in the Table 6.1. The F'-privacy is given by $f'_i = 3 \times f_i + 0.02$, where*

- $f_i = 0.05128$, $f'_i = 0.1738$ *for* $i = 1, 2, 3, 4$;
- $f_i = 0.10256$, $f'_i = 0.32769$ *for* $i = 5, 6, 7, 8$;
- $f_i = 0.17948$, $f'_i = 0.5584$ *for* $i = 9$;
- $f_i = 0.20512$, $f'_i = 0.6353$ *for* $i = 10$.

The result of Top-down bucketing algorithm is shown in the Figure 6.1 (The bold table is the final results of bucketing): First, we get two kinds of buckets: $B_1(b_1 = 3, S_1 = 5)$ *and* $B_2(b_2 = 4, S_2 = 6)$*. We partition the data by applying data partitioning in the data.* B_1 *can not be apart any more. The data of* B_2 *is treated as Data Set and call the Top-down bucketing algorithm again. We will do this recursively until all buckets can not be apart any more.*

- *First around: the input data is the the original data set. We get the bucket setting* $B_1(b_1 = 3, S_1 = 5)$ *and* $B_2(b_2 = 4, S_2 = 6)$*. The Information Loss is* $b_1(S_1 - 1)^2 + b_2(S_2 - 1)^2 = 132$*.*

- *Second around: the input data is* B_1 *and* B_2*.* B_1 *can not be apart any more. We separate* B_2 *to get the bucket setting* $B_3(b_3 = 3, S_3 = 4)$ *and* $B_4(b_4 = 3, S_4 = 4)$*. Combining with* $B_1(b_1 = 3, S_1 = 5)$*, the Information Loss is* $b_1(S_1 - 1)^2 + b_3(S_3 - 1)^2 + b_4(S_4 - 1)^2 = 125$*.*

- *Third around: the input data is* B_3 *and* B_4*.* B_3 *can not be apart any more. We separate* B_4 *to get the bucket setting* $B_5(b_5 = 2, S_5 = 4)$ *and* $B_6(b_6 = 2, S_6 = 2)$*. Combining with the* $B_1(b_1 = 3, S_1 = 5)$ *and* $B_3(b_3 = 3, S_3 = 4)$*, the Information Loss is* $b_1(S_1 - 1)^2 + b_3(S_3 - 1)^2 + b_5(S_5 - 1)^2 + b_6(S_6 - 1)^2 = 118$

As we can see, every time we increase the kinds of buckets, the information loss decrease. After three rounds, we get four kinds of buckets. The Information Loss is only 118. The information loss of the optimal two-size bucketing is 132. It clearly shows the multiple-size bucketing can generate a better buckets setting than two-size bucketing.

In the next chapter, we will imply these algorithms in the real data to evaluate the effectiveness and accuracy of our algorithm.

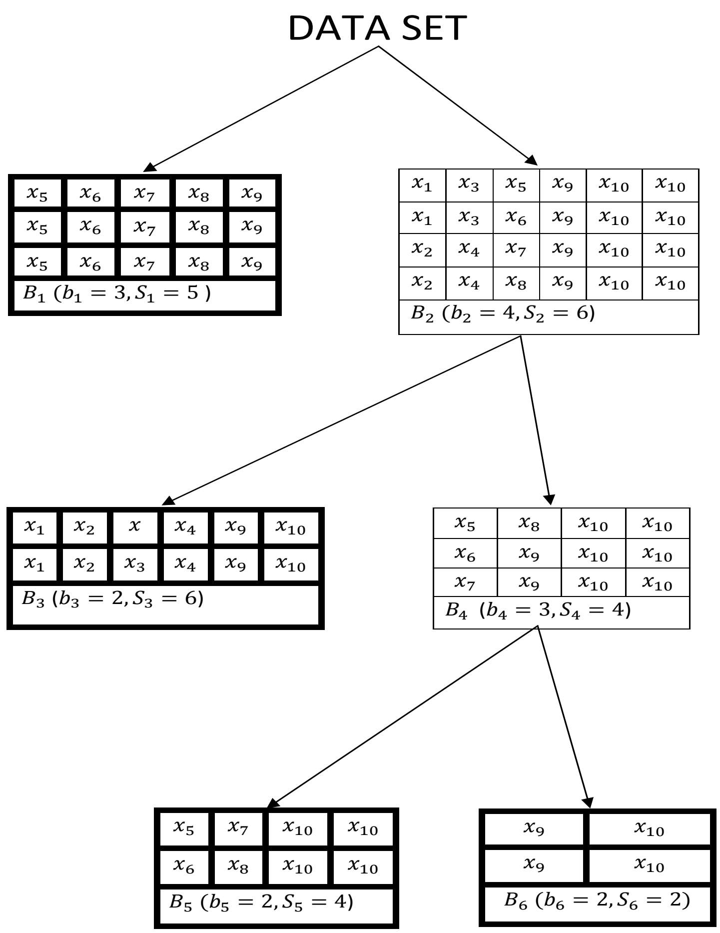


Figure 6.1: Example for Top-down Bucketing Algorithm

Chapter 7

Empirical Studies

We evaluate the effectiveness and efficiency of the algorithms proposed in Chapter 5.3. For this purpose, we utilize the real data set CENSUS containing personal information of 500K American adults. This data set was previously used in [11], [36] and [28]. Table 7.1 shows the eight discrete attributes of the CENSUS data. We are going to test our algorithms on the two selected data sets generated from CENSUS. The first data set *OCC* has *Occupation* as SA and the 7 remaining attributes as the QID-attributes. The second data set *EDU* has *Education* as SA and the 7 remaining attributes as the QID-attributes. We use OCC-n and EDU-n to denote the data sets of OCC and EDU of the cardinality n. Figure 7.1 shows the frequency distribution of SA. The parameters and settings are summarized in Table 7.2 with the default setting in bold face.

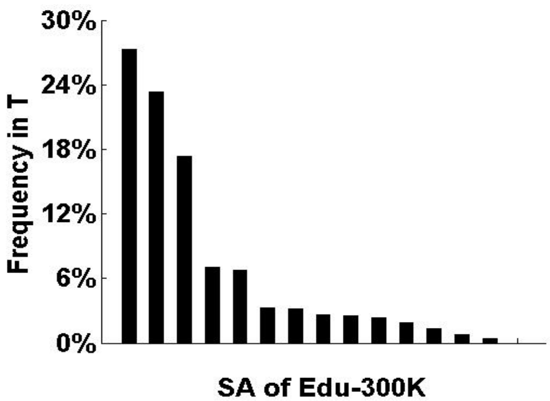
$Attribute$	Domain Size
Age	76
Gender	2
Education	14
Marital	6
Race	9
Work-Class	10
Country	83
Occupation	50

Table 7.1: Table of CENSUS Statistics

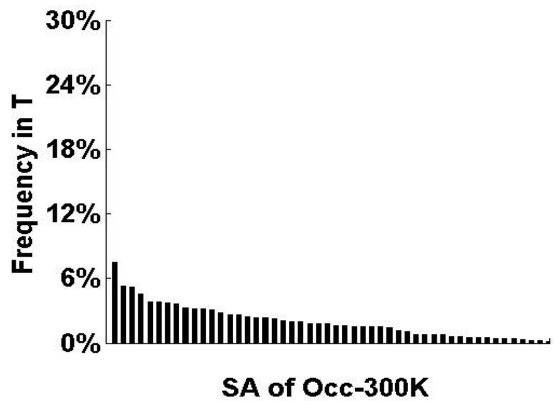
We evaluate our algorithms by three criteria: suitability for handling varied sensitivity, data utility, and scalability.

Parameters	Settings
Cardinality $\|T\|$	100k, 200k, **300k**, 400k, 500k
f'_i-privacy for x_i	$f'_i = min\{1, \theta \times f_i + 0.02\}$
Privacy coefficient θ	2, 4, **8**, 16, 32
M	$min_i\{\lceil 1/f'_i \rceil\}$
M'	50

Table 7.2: Table of Parameter Settings



(a) EDU



(b) OCC

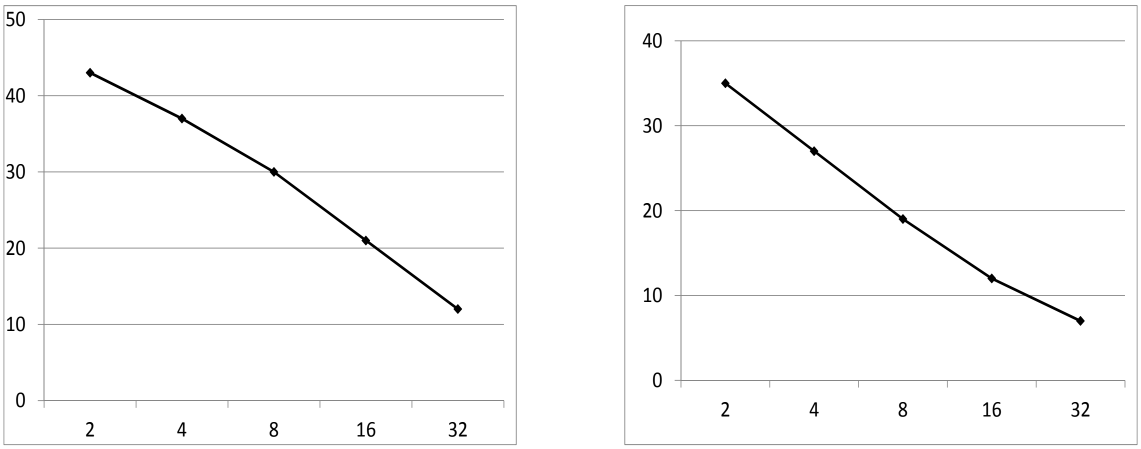
Figure 7.1: Frequency Distribution of SA

7.1 Criterion 1: Handling Varied Sensitivity

Our first objective is to study the suitability of F'-privacy for handling varied sensitivity and skewed distribution of sensitive values. For concreteness, we specify F'-privacy by $f'_i = min\{1, \theta \times f_i + 0.02\}$, where θ is the *privacy coefficient* chosen from $\{2, 4, 8, 16, 32\}$. This specification models a linear relation between the sensitivity f'_i and the frequency f_i for x_i. Since $f'_i \geq f_i$ for all x_i's, a solution satisfying F'-privacy always exists following from Lemma 1 in Section 3.3. In fact, we can find a solution even when we set up the constraint for the maximum bucket size to be $M' = 50$.

For comparison purposes, we apply ℓ-diversity to model the above F'-privacy, where ℓ is set to $\lceil 1/\min_i\{f'_i\} \rceil$. For the OCC-300K and EDU-300K data sets, the minimum f_i of 0.18% and 0.44%, respectively. Figure 7.2 shows the relation between θ and ℓ. Except for $\theta = 32$, a rather large ℓ is required to enforce F'-privacy. As such, the buckets produced by Anonymity [11] have a large size ℓ or $\ell + 1$. Thus, the information loss is rather large. A large ℓ also renders ℓ-diversity too restrictive since $1/\ell \geq \max_i\{f_i\}$ is necessary for the existence of a ℓ-diversity solution. With OCC-300K's maximum f_i being 7.5% and EDU-300K's maximum being 27.3%, this condition is violated for all $\ell \geq 14$ in the case of OCC-300K and for all $\ell \geq 4$ in the case of EDU-300K. This study suggests that

ℓ-diversity is not suitable for handling sensitive values of varied sensitivity and skewed distribution.



(a) OCC (b) EDU

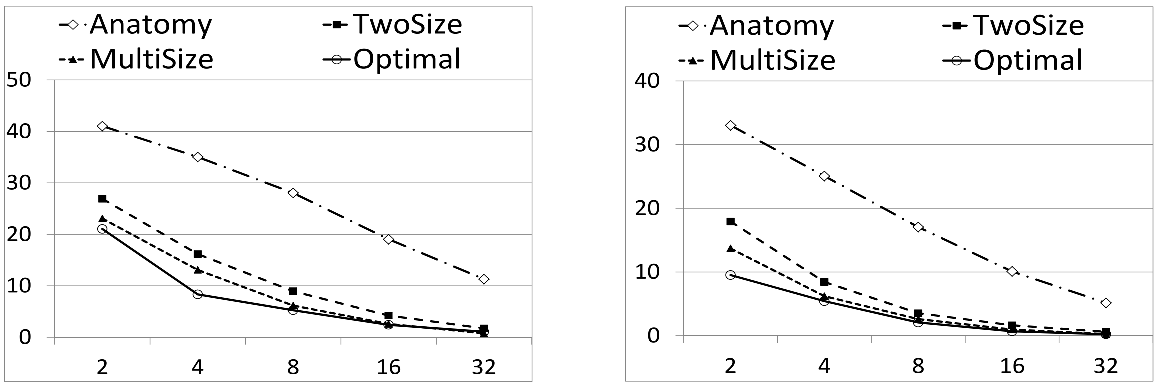
Figure 7.2: The Relation Between ℓ (y-axis) and Privacy Coefficient θ (x-axis)

7.2 Criterion 2: Data Utility

Our second objective is to evaluate the utility of T^*. We consider two utility metrics, Information Loss (Definition 7 in Section 3.4) and *Relative Error (RE)* for count queries previously used in [11]. We compare *Two-Size Bucketing*, denoted by "TwoSize", and *Multi-Size Bucketing*, denoted by "MultiSize", against two other methods. The first one is *Optimal Multi-Size Bucketing*, denoted by "Optimal". This is the exact solution to the optimal multi-sized bucket setting problem solved by an integer linear program. "Optimal" provides the theoretical lower bound on $Loss$, but it is feasible only for a small domain size $|SA|$. The second one is *Anonymity* [11] with ℓ-diversity being set to $\ell = \lceil 1/\min_i\{f'_i\}\rceil$. For our algorithms, the minimum bucket size M is set to be $\min_i\{\lceil 1/f'_i\rceil\}$ and the maximum bucket size M' is set to be 50.

7.2.1 Information Loss

Figure 7.3 shows information loss vs the privacy coefficient θ on the default OCC-300K and EDU-300K. The study in Section 7.1 shows that for most F'-privacy polices, the corresponding ℓ-diversity cannot be achieved on the OCC and EDU data sets. For comparison purposes, we compute the information loss for "Anonymity" based on the bucket size of ℓ or $\ell + 1$ while ignoring the privacy constraint. "Anonymity" has a significantly higher information loss than all other methods across all settings of θ due to the large bucket sizes ℓ and $\ell + 1$ are large. The information losses for the other three are almost close. “TwoSize” has only a slightly higher information loss than “MultiSize”, which has only a slightly higher information loss than “Optimal”. This study suggests that the restriction to the two-size bucketing problem causes only a small loss of optimality and that the heuristic solution is a good approximation to the optimal solution of the multi-size bucket setting problem.



(a) OCC (b) EDU

Figure 7.3: Information Loss (y-axis) vs Privacy Coefficient θ (x-axis)

7.2.2 Relative Error

We adapt *count queries* Q of the form:

SELECT COUNT(*) FROM T
WHERE $pred(A_1)$ AND ... AND $pred(A_{q_d})$ AND $pred(SA)$

Here $A_1, \cdots, A_{q_d}$ are randomly selected QID-attributes. The total number of QID-attributes is 7. The query dimensionality q_d is randomly selected from $\{1, \cdots, 7\}$ with equal probability. For any attribute A, $pred(A)$ has the form

$$A = a_1 \text{ OR ... OR } A = a_b,$$

where a_i is a random value from the domain of A. The value of b depends on the expected query selectivity, which was set to be 1% here. The answer act to Q using T is the number of records in T that satisfy the condition in the WHERE clause. We created a pool of 5,000 count queries of the above form. For each query Q in the pool, we compute the estimated answer est using T^* instead of table T. The *Relative Error (RE)* on Q is defined to be $RE = |act - est|/act$, where act is its original data, and est is the estimate computed from the bucketized table. RE reflects the

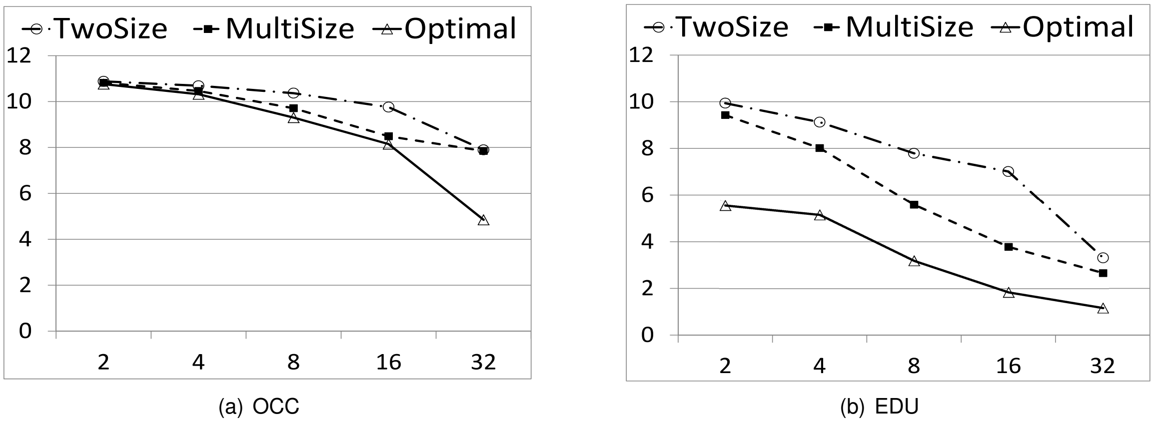


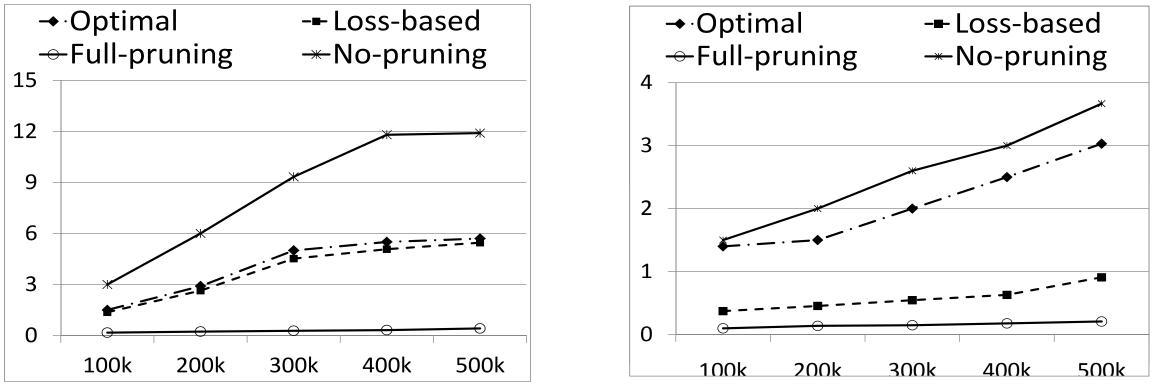
Figure 7.4: Relative Error (%) (y-axis) vs Privacy Coefficient θ (x-axis)

difference between the original data set and modified data set. We report the average RE over all queries in the pool.

Figure 7.4 shows RE vs the privacy coefficient θ on the default OCC-300K and EDU-300K. For the OCC data set, the maximum RE is slightly over 10%. The RE's for "TwoSize", "MultiSize", and "Optimal" are relatively close to each other. For the EDU data set, all RE's are no more than 10%. "MultiSize" improves upon "TwoSize" by about 2%, and "Optimal" improves upon "MultiSize" by about 2%. This study suggests that the solutions of the optimal two-size bucketing and the heuristic multi-size bucketing are highly accurate for answering count queries, with the RE below 10% for most F'-privacy considered. "Anonymity" was not included since there is no corresponding ℓ-diversity solution for most F'-privacy considered (see Section 7.1).

7.3 Criterion 3: Scalability

We now evaluate the scalability for handling large data sets. We focus on *Two-Size Bucketing* because it is a key component of *Multi-Size Bucketing*. "No-pruning" refers to the sequential search of the full list Γ (defined in the Section 5.3.1) without any pruning; "Loss-pruning" refers to the loss-based pruning in Section 5.3.2; "Full-pruning" refers to *Two-Size Bucketing* in Section 5.3.4, which exploits both loss-based pruning and privacy-based pruning. "Optimal" refers to the integer linear program solution to the two-size bucketing problem. We study the *Runtime* with respect to the cardinality $|T|$ and the domain size $|SA|$. The default privacy coefficient setting $\theta = 8$ is used. The algorithm of "Optimal" is implemented in MATLAB 7.11.0 (R2010b), using the computing power of the Gurobi Optimizer, which is widely used in data processing. Algorithms except "Optimal" are implemented in C++. All the programs run on a Windows 64 bits Platform with CPU of 2.53 GHz and memory size of 12GB. Each algorithm was run 100 times and the average time is reported below.

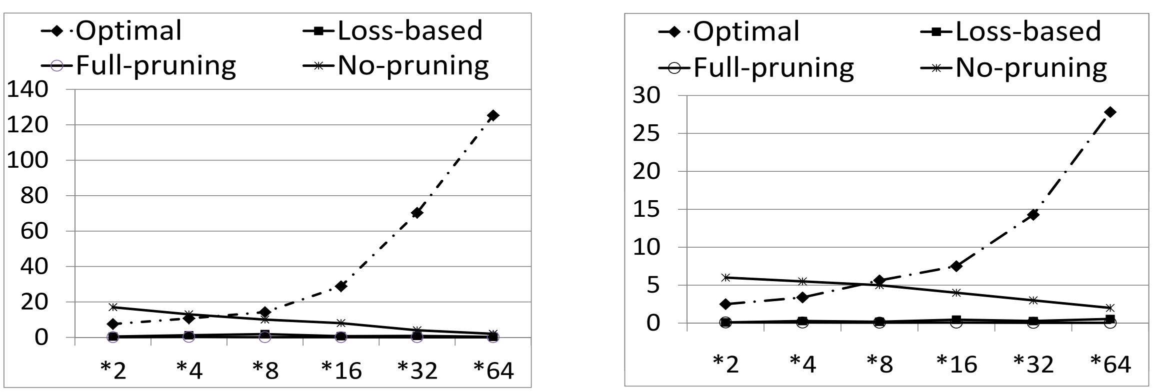


(a) OCC (b) EDU

Figure 7.5: Runtime (seconds) (y-axis) vs Cardinality $|T|$ (x-axis)

7.3.1 Scalability with $|T|$

Figure 7.5 shows $Runtime$ vs the cardinality $|T|$. "Full-pruning" takes the least time and "No-pruning" takes the most time. "Loss-pruning" significantly reduces the time compared to "No-pruning", but has an increasing trend in $Runtime$ as $|T|$ increases because of the sequential search of the first valid pair in the list Γ'. In contrast, a larger $|T|$ does not affect "Full-pruning" much because ""Full-pruning" locates the first valid pair by a binary search over Γ'. "Optimal" takes less time than "No-pruning" because the domain size $|SA|$ is relatively small. The next experiment shows that the comparison is reversed for a large domain size $|SA|$.



(a) OCC (b) EDU

Figure 7.6: Runtime (seconds) (y-axis) vs Scale-up Factor γ for $|SA|$ (x-axis)

7.3.2 Scalability with $|SA|$

We scale up $|SA|$ for OCC-500K and EDU-500K by a factor γ, where γ is ranged over 2, 4, 8, 16, 32 and 64. Assume that the domain of SA has the form $\{0, 1, \cdots, m-1\}$. For each record t in T, we replace $t[SA]$ in t with the value $\gamma \times t[SA] + r$, where r is an integer selected randomly from the range $[0, \gamma - 1]$ with equal probability. Thus the new domain of SA has the size $m \times \gamma$. Figure 7.6 shows $Runtime$ vs the scale-up factor γ. As γ increases, $Runtime$ of "Optimal" increases quickly because the integer linear programming is exponential in the domain size $|SA|$. $Runtime$ of the other algorithms increases little because the complexity of these algorithms is linear in the domain size $|SA|$. Interestingly, as $|SA|$ increases, $Runtime$ of "No-pruning" decreases. A close look reveals that when there are more SA values, f_i and f'_i become smaller and the minimum bucket size M becomes larger, which leads to a short Γ list. A shorter Γ list benefits most the sequential search based "No-pruning".

In summary, we showed that the proposed methods can better handle sensitive values of varied sensitivity and skewed distribution, therefore, retain more information in the data, and the solution is scalable for large data sets.

Chapter 8

Conclusion

Publishing data that allows data analysis to satisfy the demands for disclose sensitive information without compromising individual privacy is an important research field.

In this study we propose a new idea of multiple size bucketization, which has two advantages compared with ℓ-diversity. The first advantage is that it is easy to find a feasible solution for bucketization even when the distribution of data set is very unbalanced. The second advantage is that even if ℓ-diversity can find a solution, our algorithm can get a better bucket setting with less information loss. We also propose an effective top-down algorithm for the multiple size bucketing, which calls the two size algorithm recursively. We propose two pruning strategies to improve the speed of the two size algorithm: 1) Lose-based pruning strategy; 2)Privacy-based pruning strategy. The figures in Section 7.3 show both of them effectively accelerate the speed of the algorithm. The top-down bucketing algorithm also is proven that have got similar bucket setting with the Integer Linear Programming, and the speed of Top-down algorithm is much faster than ILP algorithm.

This project focuses on the static data set to publish. Our future work will try to apply the multiple size bucketization policy to the dynamic data sets. So we can handle a practical problem, where people can add and delete the record in the data set without violating the privacy protection.

Bibliography

[1] Usama Fayyad, Gregory Piatetsky-shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996. 1

[2] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010. 1, 2, 9

[3] S Bowers. 2008 HSBC loses disk with policy details of 370,000 customers. *The Guardian News and Media Ltd UK.*, 8th April 2008. 1

[4] Wong Raymond Chi-Wing, Fu Ada Wai-Chee, Wang Ke, and Pei Jian. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 543–554. VLDB Endowment, 2007. 2

[5] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Wang Ke. (a, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 754–759, New York, NY, USA, 2006. 2

[6] Johannes Gehrke. Models and methods for privacy-preserving data publishing and analysis: invited tutorial. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 316–316, New York, NY, USA, 2005. 2

[7] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association. 2

[8] T. Dalenius. Finding a needle in a haystack - or identifying anonymous census record. *Journal of Official Statistics*, 2:329–336, 1986. 3

[9] L Burnett, K Barlow-Stewart, AL Proos, and H Aizenberg. The ”genetrustee”: a universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10:506–513. 3

[10] L. H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75:377–385, 1980. 3

[11] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, October 2002. 3, 8, 43, 44, 45

[12] M. Barbaro and T. Zeller. A face is exposed for aol searcher. *New York Times*, August 2006. 3

[13] Charu C. Aggarwal and Philip S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer Publishing Company, Incorporated, 2008. 6

[14] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, December 2002. 6

[15] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2):439–450, May 2000. 6

[16] Nabil R. Adam and John C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, December 1989. 7

[17] Confidentiality and Data Access Committee. Report on statistical disclosure limitation methodology. *Technical Report 22, Office of Management and Budget*, December 2005. 7

[18] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC ’08, pages 609–618, New York, NY, USA, 2008. 7

[19] Blum, Avrim, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. pages 128–138, 2005. 7

[20] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS ’03, pages 202–210, New York, NY, USA, 2003. ACM. 7

[21] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third conference on Theory of Cryptography*, TCC’06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag. 7

[22] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 85–94, New York, NY, USA, 2007. ACM. 7

[23] William E Winkler. Overview of record linkage and current research directions. Technical report, BUREAU OF THE CENSUS, 2006. 8

[24] Ke Wang, P.S. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 249–256, 2004. 8

[25] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pages 188–193, 1998. 8

[26] L. Sweeney P. Samarati. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. *Technical Report, SRI International*, 1998. 8

[27] C. Clifton. Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8:281–307, November 2000. 10

[28] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007. 10, 11, 43

[29] Xiaokui Xiao and Yufei Tao. Anatomy: simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 139–150. VLDB Endowment, 2006. 10

[30] Ninghui Li, Tiancheng Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference*, April 2007. 11

[31] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth movers distance as a metric for image retrieval. *International Journal of Computer Vision*, 40:2000, 2000. 12

[32] Manas A. Pathak, Shantanu Rane, Wei Sun 0008, and Bhiksha Raj. Privacy preserving probabilistic inference with hidden markov models. In *ICASSP*, pages 5868–5871. IEEE, 2011. 12

[33] Jianneng Cao and Panagiotis Karras. Publishing microdata with a robust privacy guarantee. *Proceedings of the VLDB Endowment*, 5(11):1388–1399, 2012. 12

[34] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 720–731. VLDB Endowment, 2004. 15

[35] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, pages 216–227, New York, NY, USA, 2002. ACM. 15

[36] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Efficient full-domain k-anonymity. *SIGMOD Conference*, 2005. 43