PRIVACY-PRESERVING DATA PUBLICATION FOR STATIC AND STREAMING DATA

JIANNENG CAO

(*M.Eng., South China University of Technology*)

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
2010

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. TAN Kian-Lee, a respectable and resourceful scholar, who has provided me valuable guidance in every stage of my research work including this thesis. His keen observation, insightful instructions, impressive patience, are the driving forces of my work. In addition, I would like to take this opportunity to thank all those whom I work with in the past five years. A special acknowledgement should be shown to Dr. Barbara Carminati and Prof. Elena Ferrari from the University of Insubria, Italy; I have benefited greatly from the joint work with them about access control over data streams. I am particularly indebted to Dr. Panagiotis Karras, whose strong presentation skills, impressive courage, insightful comments and suggestions, help me to work out problems during the difficult course of my research. My sincere appreciation also goes to Associate Prof. Panos Kalnis (King Abdullah University of Science and Technology, Saudi Arabia), and Dr. Chedy Raïssi (INRIA, Nancy GrandEst, France) for their kind support. Furthermore, I would also like to thank my thesis examination committee members, Associate Prof. CHAN Chee-Yong and Associate Prof. Stephane Bressan, for their valuable comments.

I would extend my thanks to my friends: Cao Yu, Cheng Weiwei, Gabriel Ghinita, Htoo Htet Aung, Li Xiaohui, Li Yingguang, Meduri Venkata Vamsikrishna, Shi Lei, Tran Quoc Trung, Wang Zhenkui, Wu Aihong, Wu Ji, Wu Wei, Xiang Shili, Xiao Qian, Xue Mingqiang, Zhai Boxuan, Zhou Jian, and many others not listed here. Most particularly, I must thank Sheng Chang for so many valuable suggestions in my research work.

Last but not least, I would like to express my heartfelt gratitude to my

beloved family—my wife Zhong Minxian, my parents, and my sisters, for their support and confidence in me in all the past years.

Table of Contents

Summary

The publication of microdata poses a privacy threat: anonymous personal records can be re-identified using third party data. Past research partitions data into equivalence classes (ECs), i.e., groups of records indistinguishable on *Quasi-identifier* values, and has striven to define the *privacy guarantee* that publishable ECs should satisfy, culminating in the notion of t-closeness. Despite this progress, no algorithm tailored for t-closeness has been proposed so far. To fill this gap, we present SABRE, a Sensitive Attribute Bucketization and REdistribution framework for t-closeness. It first greedily partitions a table into buckets of similar sensitive attribute ($\mathcal{SA}$) values, and then redistributes the tuples of each bucket into dynamically determined ECs. Nevertheless, t-closeness, as the state of the art, still fails to translate t, the privacy threshold, into any intelligible privacy guarantee. To address this limitation, we propose β-likeness, a novel robust model for microdata anonymization, which postulates that each EC should satisfy a threshold on the *positive relative difference* between each $\mathcal{SA}$ value's frequency in the EC and that in the overall anonymized table. Thus, it clearly quantifies the extra information that an adversary is allowed to gain after seeing a published EC.

Most of privacy preserving techniques, including SABRE and β-likeness, are designed for static data sets. However, in some application environments, data appear in a sequence (stream) of append-only tuples, which are continuous, transient, and usually unbounded. As such, traditional anonymization schemes cannot be applied on them directly. Moreover, in streaming applications, there is a need to offer strong guarantees on the maximum allowed delay between incoming data and the corresponding anonymized output. To cope

with these requirements, we first present CASTLE (Continuously Anonymizing STreaming data via adaptive cLustEring), a cluster-based scheme that continuously anonymizes data streams and, at the same time, ensures the freshness of the anonymized data by satisfying specified delay constraints. We further show how CASTLE can be easily extended to handle ℓ-diversity. To better protect the privacy of streaming data, we have also revised t-closeness and applied it to data streams. We propose (ω, t)-closeness, which requires that for any EC, there exists a window, which has a size of ω and contains the EC, so that the difference of $\mathcal{SA}$ distribution between the EC and the window is no more than t. Thus, the closeness constraints are restricted in windows instead of a whole unbounded stream, complying with the general requirement that streaming tuples are processed in windows.

We have implemented all the proposed schemes and conducted performance evaluation on them. The extensive experimental results show that our schemes achieve information quality superior to existing schemes, and can be faster as well.

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Organizations such as government agencies or hospitals collect microdata (e.g., medical reports, financial transactions, and residence records), and regularly release them to serve the purposes of research and public benefits. For example, a predication model (e.g., a decision tree) built on medical reports can help clinicians determine the most appropriate care for newly diagnosed cases of diseases. However, such data contain sensitive personal information, and improper disclosure of them puts the privacy of individuals at risk. Consider again the medical reports. The disclosure that someone suffers from diabetes has a negative impact on his/her employment and the coverage of insurance. Therefore, a conflict exists between perceived benefits and the sacrifice of individual privacy in data dissemination.

There are two extremes in handling the conflict: one is disseminating data without any change, thus achieving full data utility at the expense of privacy; the other is withholding the publication, hence sacrificing utility for full privacy. Obviously, neither of these is practical and useful. In this thesis, we adopt an alternative approach by finding a balanced point between privacy and data utility, using available privacy models and our newly developed ones.

Data publication takes place in both static and dynamic settings. In static settings, data are collected, anonymized, and then published only once. In dynamic

circumstances, data arrive continuously, and are anonymized/published in a sequence of times; in some cases a tuple can even appear in multiple anonymizations. Our study involves static data sets, and data streams, a common and important case of dynamic setting.

1.1 Privacy protection for static data sets

In static settings, the privacy of data is guaranteed by the algorithms designed according to different privacy models proposed so far [31, 76]. Each model has its own requirements on the *form* that the data should follow before the publication. The research of privacy protection on static data sets can be seen as a history of progressively more sophisticated models. In the following we briefly present these models related to our thesis in the chronological order, and discuss their functions and limitations.

Age	Sex	Zipcode	Disease
26	Male	53711	Bronchitis
27	Male	53710	Broken arm
27	Male	53712	AIDS
25	Male	53711	Hepatitis
25	Female	53712	Hepatitis
28	Female	53711	Hepatitis

Table 1.1: Microdata about patients

Name	Age	Sex	Zipcode
Bob	26	Male	53711
Mike	27	Male	53710
John	27	Male	53712
Jack	25	Male	53711
Kate	25	Female	53712
Jane	28	Female	53711

Table 1.2: Voter registration list

1.1.1 k-anonymity

The pioneering work for privacy preserving data publication is the concept of k-anonymity [66, 67] proposed by Samarati and Sweeney. They discovered that microdata with identity information (e.g., social security number, name, and telephone number) removed, may still be vulnerable to *linking attack*. Consider patient records in Table 1.1 and voter registration list in Table 1.2. Although all

the records in Table 1.1 have their identity information removed, they can still be re-identified by joining Table 1.1 with Table 1.2 on their shared attributes—*Age*, *Sex*, and *Zipcode*. For example, after the join, we can infer that Bob suffers from *Bronchitis*.

The set of attributes that can be exploited to re-identify individuals by joining/matching them with external databases is called *quasi-identifier* (QI). In the above example, {*Age, Sex, Zipcode*} is the QI. An attribute whose disclosure puts the individual privacy at risk is known as the sensitive attribute ($\mathcal{SA}$). *Disease* in Table 1.1 is such an $\mathcal{SA}$. Under k-anonymity, records of the dataset are partitioned into groups, each with a size of at least k, and the QI values in a same group are replaced by a single *generalized* value. A group of tuples with the same QI value is an *equivalence class* (EC). In this way, all the records in the same group/EC are indistinguishable from each other with regard to QI. Hence, k-anonymity successfully protects against *identity disclosure*, by hiding one person in a crowd of at least $k - 1$ other persons. Let us go on with the running example. Table 1.1 is 3-anonymized to Table 1.3 with two ECs of size 3 each. Consider the first record in Table 1.3. At present, Bob, Mike, and John are all equally linkable to it. Thus, Bob is hidden in the crowd of {Bob, Mike, John}.

EC	Age	Sex	Zipcode	Disease
1	[26-27]	Male	[53710-53712]	Bronchitis
	[26-27]	Male	[53710-53712]	Broken arm
	[26-27]	Male	[53710-53712]	AIDS
2	[25-28]	Person	[53711-53712]	Hepatitis
	[25-28]	Person	[53711-53712]	Hepatitis
	[25-28]	Person	[53711-53712]	Hepatitis

Table 1.3: A 3-anonymous table

Although k-anonymity successfully protects against identity disclosure, it suffers from *homogeneous attack* due to neglecting non-QI sensitive attribute. When the distribution of sensitive attribute ($\mathcal{SA}$) values in an EC is highly skewed, an attacker may infer the sensitive value of an individual with a high confidence. For instance, equivalence class 2 in Table 1.3 contains all tuples with *Hepatitis* as $\mathcal{SA}$ value. Hence, an attacker can infer with 100% confidence that all persons referred by EC 2 have hepatitis, i.e., Jack, Kate, and Jane all have this disease.

1.1.2 ℓ-diversity

To address the limitation of k-anonymity, Machanavajjhala et al. [57] put forward the principle of ℓ-diversity, which postulates that each EC should contain at least ℓ distinct "well represented" $\mathcal{SA}$ values. The intuition behind ℓ-diversity is that each person is linkable to ℓ distinct $\mathcal{SA}$ values, thus the association between the person and his/her specific $\mathcal{SA}$ value is blurred. Since the requirement that values be "well represented" can be explained in multiple ways, there are different instantiations of ℓ-diversity. Please refer to Section 2.1.2 for a survey.

Name	**Weight**	**Age**	**Disease**
Mike	60	40	SARS
Alice	70	50	intestinal cancer
John	60	60	pneumonia
Bob	50	50	bronchitis
Beth	80	50	gastric flu
Carol	70	70	gastric ulcer

Table 1.4: Patient records

EC	**Weight**	**Age**	**Disease**
1	[50-60]	[40-60]	SARS
	[50-60]	[40-60]	pneumonia
	[50-60]	[40-60]	bronchitis
2	[70-80]	[50-70]	intestinal cancer
	[70-80]	[50-70]	gastric flu
	[70-80]	[50-70]	gastric ulcer

Table 1.5: 3-diverse published table

Still, ℓ-diversity fails to protect against attacks by an adversary's unavoidable knowledge of the *overall* $\mathcal{SA}$ distribution in a released table [52]. In particular, a *similarity* attack occurs when the $\mathcal{SA}$ values in an EC are semantically similar. For example, Table 1.4 has $\{$*Weight, Age*$\}$ as QI and *Disease* as $\mathcal{SA}$. Attribute

Name has been deleted from the Table; we put it outside the table only for reference. Table 1.5 is a 3-diverse version of Table 1.4, nevertheless all tuples in EC 1 indicate a *respiratory problem*.

Furthermore, a *skewness* attack may take place when the $\mathcal{SA}$ distribution in an EC differs substantially from that in the published table as a whole. For example, assume a 10-diverse form $\mathcal{T}'$ of a medical records table $\mathcal{T}$, in which 0.1% persons are infected with HIV, and an EC $\mathcal{G} \in \mathcal{T}'$ containing 10 distinct $\mathcal{SA}$ values, with one occurrence of HIV among them. Then the probability of HIV in $\mathcal{G}$ is 10%, while in $\mathcal{T}$ it is 0.1%. This 100-fold increase creates a big undesirable leak of information.

1.1.3 t-closeness

The t-closeness guarantee [52] is conceived to address the limitations of ℓ-diversity. It requires that the distance between the local $\mathcal{SA}$ distribution in any EC and the global $\mathcal{SA}$ distribution in the whole table be at most t, a threshold. t-closeness modeling takes the global $\mathcal{SA}$ distribution as the public knowledge, and considers the difference between the two distributions as the *additional information* that an attacker gains. Intuitively, a smaller t indicates that the two distributions are more similar, thus the information gain is smaller. t-closeness directly protects against *skewness* attack, while it also provides defence against *similarity* attack, depending on the extent to which semantic similarity exists among $\mathcal{SA}$ values in the whole table.

So far, t-closeness schemes [52, 53] are built on k-anonymity instantiations; they extend either Incognito [48] or Mondrian [49] by adding an extra condition: the produced ECs satisfy t-closeness. However, k-anonymity and t-closeness are very different privacy models—the former focuses on the EC sizes, requiring

the number of tuples in each EC to be no less than k; the latter focuses on the $\mathcal{SA}$ distributions, constraining the similarity between $\mathcal{SA}$ distribution in any EC and its global distribution. With such distinct requirements on created ECs, as expected, a good t-closeness-complying scheme may not be derived from k-anonymity schemes. Therefore, the question of designing a scheme tailored for t-closeness remains open.

1.2 Privacy protection for data streams

Data streams are common to many application environments, such as, telecommunication, market-basket analysis, network monitoring, and sensor networks. Mining these continuous data streams [36, 56, 85] helps companies (the owner of data streams) to learn the behavior of their customers, thus bringing unique opportunities. Many companies do not have the in-house expertise of data mining, so it is beneficial to outsource the mining to a professional third party [62]. However, data streams may contain much private information that must be carefully protected. Consider Amazon.com. In a single day, it records hundreds of thousands of online sales transactions, which are received in the form of streaming data. Suppose that the sales transaction stream has the schema $S(tid, cid, goods)$, where tid is transaction identifier, cid is customer identifier, and $goods$ is a list of items bought by the corresponding customer. Suppose that a relation C containing the information about Amazon customers is stored on disk, with schema $C(cid, name, sex, age, zipcode, address, telephone)$. Let SC[1] be the stream generated by joining S with C on cid. Suppose moreover that, to analyze

[1] In real stream systems, typically customer information does not appear in the stream to reduce redundancy. Mining, which needs customer information, requires joining the data stream with local customer databases. In what follows, we consider mining and anonymization on joint streams.

customers' buying behavior (e.g., building a decision tree), the mining is on SC, and Amazon.com outsources it to a professional third-party. To protect the privacy of customers, attributes that explicitly identify customers (such as $name$, $address$ and $telephone$) are projected out of SC. However, as pointed out in Section 1.1.1, the remaining data in SC may still be re-identified by joining QI attributes (e.g., sex, age and $zipcode$) with external public databases (e.g., a voter registration table). Therefore, the streaming transactions in SC need to be carefully anonymized before they are passed to the third-party.

Most of the previous anonymization algorithms are designed specifically for static data sets. They cannot be directly applied on streaming data for the following reasons. First, these techniques typically assume that each record in a data set is associated with a different person, that is, each person appears in the data set only once. Although this assumption is reasonable in a static setting, it is not realistic for streaming data. Second, due to the constraints of performance and storage, backtracking over streaming data is not allowed. However, traditional anonymization schemes scan a data set multiple times, contrary to the one-pass requirement imposed on algorithms for data streams. Furthermore, streaming tuples have a *temporal dimension*. They arrive at a certain rate, they are dynamically processed, and the result is output with a certain delay. In some applications, the output data are immediately used to trigger appropriate procedures. For example, in a sensor network application the output stream can be used to react in real time to some anomalous situations, thus the time to react is very crucial. Therefore, a data stream anonymization scheme should ensure strong guarantees on the *maximum delay* between the input of data and their output. Finally, some privacy models are not directly applicable to data

streams. Models such as t-closeness assume the existence of a global $\mathcal{SA}$ distribution. However, data streams are unbounded, and such a global distribution is unavailable. Therefore, these models themselves need to be modified before being adopted for streaming tuples. As a consequence, all previous anonymization algorithms designed according to their constraints cannot be applied on data streams.

Based on the above analysis, we can safely conclude that we need to specifically design new algorithms for anonymizing stream data rather than simply applying existing ones.

1.3 The thesis contributions

Our contributions are divided into two portions. In the first part, we propose novel privacy models as well as sophisticated algorithms to anonymize static data sets. In the second part, we customize privacy models to meet the unique requirements of data streams, and develop new solutions to continuously anonymize streaming data.

1.3.1 The models and algorithms in static setting

SABRE: A tailored t-closeness framework

The past research on privacy models culminates in t-closeness. Despite this progress, there is no anonymization algorithm tailored for it. Therefore, our first contribution is to fill this gap with SABRE, a Sensitive Attribute Bucketization and REdistribution framework for t-closeness. SABRE operates in two phases. First, it partitions a table into buckets of similar $\mathcal{SA}$ values in a greedy fashion. Then, it redistributes tuples from each bucket into dynamically configured ECs. Following [52, 53], we employ the Earth Mover's Distance (EMD)

as a measure of closeness between distributions, and utilize a property of this measure to facilitate our approach. Namely, a tight *upper bound* for the EMD of the distribution in an EC from the overall distribution can be derived as a function of localized upper bounds for each bucket, provided that the tuples in the EC are picked *proportionally* to the sizes of the buckets they hail from. Furthermore, we prove that if the bucket partitioning obeys t-closeness, then the derived ECs also abide to t-closeness. We develop two SABRE instantiations. The former, SABRE-AK focuses on efficiency. The latter, SABRE-KNN trades some efficiency for information quality. Our extensive experimental evaluation demonstrates that both instantiations achieve information quality superior to schemes that extend algorithms customized for k-anonymity to t-closeness, while SABRE-AK is much faster than them as well.

β-likeness: an enhanced model and its algorithm

Although t-closeness takes a big step forward in privacy preservation than its predecessors, i.e., k-anonymity and ℓ-diversity, it still has its drawbacks. It calculates the distance between two $\mathcal{SA}$ distributions in a *cumulative* way, without any guarantee on the *relative* distance of a single $\mathcal{SA}$ value frequency between an EC and the whole table. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the domain of sensitive attribute $\mathcal{SA}$ in a table $\mathcal{DB}$, and $\mathcal{P} = (p_1, p_2, \ldots, p_m)$ and $\mathcal{Q} = (q_1, q_2, \ldots, q_m)$ be the $\mathcal{SA}$ distributions in $\mathcal{DB}$ and an EC, respectively. t-closeness does not provide any guarantee on the *relative* distance between p_i and q_i for single $\mathcal{SA}$ value $v_i \in \mathcal{V}$, $i = 1, 2, \ldots, m$. Thus, it fails to provide the privacy on individual $\mathcal{SA}$ values.

Based on the above observation, we introduce the concept of β-likeness, a novel, robust model for microdata anonymization that eschews the drawbacks

(see Section 4.1 for details) of t-closeness. In β-likeness, a threshold is imposed on the *relative* difference of *each* $\mathcal{SA}$ value frequency between an EC and the overall table. Thereby, β-likeness provides a clear and comprehensible privacy guarantee that limits the information gain an adversary is allowed to obtain with respect to any $\mathcal{SA}$ value of interest. Moreover, we design BUREL, an anonymization algorithm tailored for the particular requirements of β-likeness. BUREL borrows ideas from SABRE; it first BUcketizes tuples into buckets, then REdistributes tuples from buckets to ECs to attain β-likeness. Our extensive experimental study demonstrates that our β-likeness model and algorithm achieve a better trade-off between information and privacy than the state-of-the-art t-closeness schemes, even if privacy is measured by the criterion of t-closeness; in addition, it is more effective and efficient in its task than an alternative task extended from a k-anonymization algorithm.

1.3.2 The models and algorithms in data streams

k-anonymity of data streams and its scheme CASTLE

Our work on anonymizing streaming data starts with simple privacy model, i.e., k-anonymity, then goes on with more sophisticated ones, such as ℓ-diversity and t-closeness. We customize k-anonymity for the unique requirements of data streams (see Section 1.2). Then we present *CASTLE*, a scheme that Continuously k-Anonymizes STreaming data via adaptive cLustEring. CASTLE exploits quasi-identifier attributes to define a metric space: tuples are modeled as points in this space. Incoming tuples are grouped into clusters and all tuples belonging to the same cluster are released with the same generalization. Clustering of tuples is further constrained by the freshness of the output data—the delay between a tuple's input and its output is at most equal to a given parameter

δ.

CASTLE is extended to support ℓ-diversity on data streams in a straightforward manner by a cluster merge process. For each *expiring* tuple, i.e., tuple that will violate the freshness constraint soon, we check the cluster holding it. If the whole cluster as a single EC satisfies the diversity requirement, we simply output all its tuples by its generalization. Otherwise, we merge the cluster with its nearest neighbors, until such requirement is satisfied.

(ω,t)-closeness and its algorithm $\mathsf{SABRE_W}$

Besides k-anonymity and ℓ-diversity, we have also adopted t-closeness in data streams. The t-closeness model [52] assumes the presence of a global $\mathcal{SA}$ distribution, and takes it as the baseline of prior knowledge. However, data streams are continuous and unbounded, thus such a global distribution is unavailable. Thereby, we revise the definition of t-closeness, by restricting closeness constraint only in each window instead of the whole data set. We propose (ω, t)-closeness: given any EC, and a window that has a size of ω and contains the EC, the difference of their $\mathcal{SA}$ distributions is no more than t, a threshold. Based on our static t-closeness framework SABRE, we accompany (ω, t)-closeness with a customized algorithm $\mathsf{SABRE_W}$, whose soundness is supported by a solid theory foundation. Furthermore, we evaluate by experiments $\mathsf{SABRE_W}$ and schemes extended from k-anonymity algorithms; the results show that $\mathsf{SABRE_W}$ is superior to them with respect to both information quality and elapsed time.

1.4 The organization of the thesis

Just like our contributions, the thesis consists of two parts—one part for static setting; the other for data streams. Before the formal introduction of specific work, we will first provide some background knowledge in Chapter 2. It includes a survey on such popular privacy models as k-anonymity, ℓ-diversity, and t-closeness; important algorithms proposed so far according to these models are reviewed by discussing their contributions and limitations. After the survey on related work, in the same chapter we briefly discuss about data streams, their applications, unique characteristics, and underlying supporting engines. In addition, we also present information loss metrics that will be used throughout the thesis to measure the information quality of anonymized data.

Chapter 3 and Chapter 4 are set apart for static data set. We put forward a sophisticated t-closeness framework SABRE in Chapter 3. Specific t-closeness algorithms can be instantiated from it based on user defined applications. We provide two instantiations of SABRE, assuming that the anonymized data set is for multiple purposes. The experiment results show that they are superior to existing algorithms with regard to information quality, while one of them is much faster. Chapter 4 presents β-likeness, an enhanced privacy model compared with t-closeness. β-likeness measures the relative difference on each single $\mathcal{SA}$ value between an EC and the whole data set. Thus, it provides a clear relationship between parameter β and the privacy it affords. An algorithm BUREL customized for β-likeness is proposed.

We devote Chapter 5 and Chapter 6 to data streams. Chapter 5 presents CASTLE, a cluster-based scheme that continuously anonymizes streaming tuples, meanwhile, ensuring the freshness of output data. Although CASTLE is initially proposed for k-anonymity, it can be extended to support ℓ-diversity in

a straightforward way. Chapter 6 introduces a t-closeness-resembling privacy model for streaming data. It confines $\mathcal{SA}$ closeness constraint within each window instead of the whole unbounded data stream; it requires streaming tuples to be anonymized and output once they are expiring. In addition, a customized algorithm conforming to the privacy model has been designed.

At the end of the thesis, in Chapter 7 we conclude our works and discuss interesting items in our agenda for future research.

Research in the thesis has been partially published in international journals and conferences. Chapter 3 and Chapter 6 are from our work [27] accepted by *VLDB Journal*. The work in Chapter 5 has been accepted as a poster [24] in *ICDE 2008* and will appear in *IEEE Transactions on Dependable and Secure Computing* as a regular paper [26]. The work of Chapter 4 is under review.

CHAPTER 2

BACKGROUND

Before the formal introduction of our sophisticated anonymization schemes and novel privacy models, we first discuss the background knowledge that is closely related to our thesis. At the beginning, we review works on microdata anonymization; in particular, we will focus on k-anonymity, ℓ-diversity, and t-closeness, since they are representative models. After that, we briefly introduce data streams, discussing their unique characteristics, applications, and supporting engines. Finally, we present the information loss metrics that will be used throughout this thesis as a guide/heuristic for anonymization.

2.1 A survey on microdata anonymization

This section starts with two definitions: *Quasi-identifier* and *Equivalence Class*. They are fundamental concepts and widely used in privacy preservation data publication. Next, we will study the privacy models together with approaches designed according to their specific requirements.

Definition 2.1 (Quasi-identifier)**.** *Consider a database table $\mathcal{DB}(A_1, A_2, \ldots, A_n)$. The* quasi-identifier *(QI) of $\mathcal{DB}$ is a subset of its attributes, $\{A_1, A_2, \ldots, A_d\} \subseteq \{A_1, A_2, \ldots, A_n\}$ that can, joined with an external database, reveal the identities of the tuples involved.*

Definition 2.2 (Equivalence Class)**.** *An* equivalence class *(EC) is a group of published tuples that have the same (generalized)* QI *values.*

2.1.1 k-anonymity

The first privacy preserving model that anonymizes data while preserving their integrity was the k-anonymity model [67]. Under k-anonymity, tuples are grouped into ECs of no less than k tuples, with indistinguishable QI values. Still, the problem of *optimal* (i.e., minimal-information-loss) k-anonymization is NP-hard [12, 58] for $k \geq 3$ and more than one QI attribute. Thus, past research has proposed several heuristics for k-anonymization. Such schemes transform the data by *generalization* and/or *suppression*. A generalization replaces, or *re-codes*, all values of a QI attribute in an EC by a single *range* that contains them. For example, QI `gender` with values *male* and *female* can be generalized to *person*, and QI *age* with values *20*, *25* and *32* can be generalized to *[20, 32]*. Suppression is an extreme case of generalization that deletes some QI values or even tuples from the released table. Generalization for a *categorical* attribute is typically facilitated by a hierarchy over its values.

Generalization recodings can be classified as follows: A *global recoding* [19, 39, 43, 48, 67] maps all tuples with the same QI values to the same EC[1]. On the other hand, a *local recoding* [11, 24, 40, 83] allows tuples of the same QI values to be mapped to different generalized values (i.e., different ECs). Intuitively, ECs generated by a local recoding may, but those generated by a global recoding may *not*, *overlap* each other. The flexibility of local recoding

[1]Each tuple is one point in the metric space defined by considering each QI-attribute as one dimension. Thus, an EC can be seen as the minimum bounding box that covers all the points in it.

allows for anonymizations of higher information quality [40, 48, 49]. Furthermore, a *single-dimensional* recoding considers the domain of each QI attribute independently of the others [48] (hence forms a *grid* over the combined QI domains); on the other hand, a *multidimensional* recoding freely defines ECs over the combined domains of all QI attributes [49].

Recently, k-anonymity has been extended in multiple directions. Privacy protection towards predefined workloads has been introduced— [39] is designed specifically for classification by considering the information gain in splitting ECs; [50] caters for selected mining tasks besides classification, thus more general. However, both schemes are limited, once the workloads are unknown at the moment of data publication. In addition, k-anonymity has also been explored in dynamic settings. Wang and Fung [74] anonymize sequentially released views of the same underlying table. Schemes [38, 61] enable multiple releases of a table that has been incrementally updated.

2.1.2 ℓ-diversity

The k-anonymity model suffers from a critical limitation. While the objective of anonymization is to conceal sensitive information about the subject involved, k-anonymity pays no attention to non-QI *sensitive* attributes ($\mathcal{SA}$s). Thus, a k-anonymized table may contain ECs with so skewed a distribution of $\mathcal{SA}$ values, that an adversary can still infer the $\mathcal{SA}$ value of a record with high confidence. To address this limitation, Machanavajjhala et al. extended k-anonymity to the ℓ-diversity model, which postulates that each EC contain at least ℓ "well represented" $\mathcal{SA}$ values [57]. The requirement that values be "well represented" can be defined in diverse ways. Thus, by *entropy ℓ-diversity*, the entropy of $\mathcal{SA}$ values in each EC should be at least $\log \ell$; by *recursive (c, ℓ)-diversity*, it should

hold that $r_1 < c(r_\ell + r_{\ell+1} + \ldots + r_m)$, where r_i is the number of occurrences of the i^{th} most frequent $\mathcal{SA}$ value in a given EC, c a constant, and m the number of distinct sensitive values in that EC. Xiao and Tao propose a third instantiation of ℓ-diversity, which requires that the most frequent sensitive value in any EC occur in at most $1/\ell$ of its records [80]. This special interpretation is similar to (α, k)-Anonymity [78] once setting $\alpha = 1/\ell$.

The proposal of the ℓ-diversity model was not accompanied by an anonymization algorithm tailored for it. In response to this need, Ghinita et al. [40, 41] provide a local-recoding ℓ-diversification framework that resolves the arising high-dimensional partitioning problem via a space-filling curve, such as the Hilbert curve [59]. Furthermore, Byun et al. [23] propose diversity-aware data re-publication in the case of tuple insertion only. m-invariance [81] enhances the re-publication by supporting both tuple insertion and deletion. Bu et al. [22] make a further improvement by considering tuple update, i.e., the $\mathcal{SA}$ value of an individual may change over time.

The ℓ-diversity model is designed with a *categorical* $\mathcal{SA}$ in mind; it does not directly apply to the case of a *numerical* $\mathcal{SA}$. Namely, a diversity of numerical $\mathcal{SA}$ values does not guarantee privacy when their *range* in an EC is narrow (i.e., the values are close to each other); such a narrow range can provide accurate enough information to an adversary. To address this deficiency, Zhang et al. [86] propose a model that requires the range of a numerical $\mathcal{SA}$'s values in an EC to be wider than a threshold. However, an adversary may still be able to infer a numerical $\mathcal{SA}$ value with high confidence, if most numerical $\mathcal{SA}$ values in an EC are close, no matter how wide their total range is (i.e., the EC may simply contain a few outliers). Thus, Li et al. [51] propose a scheme requiring that $\frac{|g_c|}{|\mathcal{G}|} \leq 1/m$, where $\mathcal{G}$ is a given EC, g_c any group of close tuples in $\mathcal{G}$, and m a

parameter.

The deficiency of ℓ-diversity outlined above is most conspicuous with numerical $\mathcal{SA}$s, but not restricted to them only. It can also apply to *semantically* similar values of categorical $\mathcal{SA}$. In general, ℓ-diversity fails to guarantee privacy whenever the *distribution* of $\mathcal{SA}$ values within an EC differs substantially from their *overall* distribution in the released table, allowing *skewness* and *similarity* attacks.

2.1.3 t-closeness

Li et al. propose the t-closeness model, which requires that the difference, measured by an appropriate metric, of the $\mathcal{SA}$ distribution within any EC *from* the overall distribution of that $\mathcal{SA}$ be no more than a given threshold t [52]. According to the t-closeness model, an adversary who knows the overall $\mathcal{SA}$ distribution in the published table gains only limited more information about an EC by seeing the $\mathcal{SA}$ distribution in it.

To our knowledge, three t-closeness-attaining techniques have been proposed to date. The first of them [52] extends the Incognito method for k-anonymization [48]. It operates in an iterative manner, employing a predefined generalization hierarchy over the domain of each QI attribute. In the first round, it determines the level in the generalization hierarchy of each *single* QI attribute above which t-closeness is met. In the second round, it uses the findings of the first round to establish those *combinations* of *two* QI attributes, generalized at different levels over their respective hierarchies, that achieve t-closeness (a lattice structure represents such combinations). The scheme proceeds in this manner, examining subsets of QI attributes of size increased by one at each iteration, until it establishes the valid generalizations over *all* QI attributes that

satisfy t-closeness, and selects the best of those. Unfortunately, this approach shares the drawbacks of Incognito as an algorithm for k-anonymization: it is limited to single-dimensional global recoding. Thus, it achieves low information quality, while its worst-case time complexity is exponential in the number of QI attributes.

Likewise, the second t-closeness-obtaining scheme [53] extends the Mondrian k-anonymization method [49]. Mondrian recursively partitions the combined domain of all QI attributes, carrying out a split only if the resultant partitions have sizes of at least k. It is extended to t-closeness with an extra condition: a splitting is allowed only if the resultant partitions also obey t-closeness with respect to the overall distribution. While this method is more efficient than the Incognito-based one, it still fails in terms of information quality, as it does not cater to special features of t-closeness.

Recently, a scheme for t-closeness-like anonymization has been proposed [63]. Still, it uses perturbation (i.e., postrandomization [45]) and adds noise to anonymize the data; thus, it does not guarantee the *integrity* of the data, which is a basic common feature of the generalization-based techniques we examine in this thesis. Furthermore, [63] does not enforce the t threshold as a *maximum* difference constraint, but only as an *average* distance metric; it compares distributions measured over perturbed QI values (not over ECs) to that of the overall table; and it employs KL-divergence instead of EMD as a distance metric. Thus, the model of [63] does not provide the same worst-case privacy guarantees as t-closeness.

2.1.4 Other privacy models

Evfimievski et al. [37] introduce ρ_1-to-ρ_2 privacy principle, which imposes a bound ρ_2 on the *posterior* probability (i.e., probability after release) of certain properties in the data, given a bound ρ_1 on the *prior* probability (i.e., before data release). This model is modified in [72], where the posterior confidence should simply not exceed the prior one by more than Δ. Still, both these models measure the *absolute* confidence gain (i.e., information leak), hence do not sufficiently protect the privacy of infrequent values either. For example, both these schemes treat a probability increase from 60% to 80% as tantamount to an increase from 1% to 21%, even though the latter is an increase by 2000% and the former by only 33%. Besides, these schemes apply perturbation on the data, hence impair their integrity.

A newly proposed privacy model, δ-disclosure [21], requires that for any $\mathcal{SA}$ value v_i with frequency p_i in the original table, its frequency in any EC, q_i, should be such that $|\log(\frac{q_i}{p_i})| < \delta$. However, δ-disclosure does not distinguish between an increase and a decrease in the adversary's confidence on an $\mathcal{SA}$ value. Moreover, $\log(q_i)$ is defined only for $q_i > 0$; in effect, δ-disclosure strictly requires that each $\mathcal{SA}$ value in the original table should appear in every single EC. This requirement renders the δ-disclosure an exceedingly rigid and overprotective model. Besides, [21] does *not* propose an anonymization algorithm tailored for the δ-disclosure model; it only points out that the k-anonymization algorithm in [50], applied on the models of ℓ-diversity, t-closeness, and δ-disclosure, yields anonymizations poor in terms of information loss; it is inappropriate for [21] to directly compare the privacy gain with the utility gain [54]. Furthermore, [21] also questions the basic assumption that each tuple should be associated with a unique, homogeneous EC, as opposed to multiple,

heterogeneous ones. This question is revisited in [79] with a methodology for heterogeneous generalization, which can also be used on top of homogeneous anonymizations to improve their utility.

Recently, [46] suggested a methodology for transforming a group of $\mathcal{SA}$ values to follow a specified distribution, by permuting existing $\mathcal{SA}$ values and adding fake ones. Still, this technique damages the integrity of the data too. [75] suggested FF-anonymity, a privacy model that distinguishes between sensitive and non-sensitive information only at the *value* level; an attribute may contain both sensitive and non-sensitive values. Besides, [75] assumes that only non-sensitive information is observable by an adversary, and that generalizing a sensitive value to a non-sensitive hierarchy level conceals its sensitivity. Yet such a generalization reveals that sensitivity is hidden behind it. For example, the very act of generalizing `AIDS` to `virus` suggests that a sensitive value exists behind the generalized one. This argument is akin to that made by [77] in another context.

2.2 Data streams

In the past few years, databases of some companies such as *Amazon.com* grow at a rate of millions of records each day. Typically these data appear as a sequence (stream) of append-only tuples. They arrive at high-speed continuously and are unbounded. There is no control over their arriving order. Online processing of such data brings unique commercial opportunities to the companies, thus it is becoming an indispensable part of business operations. To efficiently manage data streams, quite a few engines are designed. Borealis [5] is a distributed stream processing system, which is based on Aurora [6] and Medusa [84]. STREAM [15] is a "general-purpose" data stream management

system (DSMS). TelegraphCQ [30] is specially designed to process adaptive data flow with an extension to support shared continuous queries. Other examples are Alert [69], Tribeca [70], OpenCQ [55], NiagaraCQ [32], CAPE [87], and so on.

Data streams have a wide range of applications. Examples include but are not limited to network traffic analysis (e.g., click streams and network security), sensor network, transaction log analysis, and financial analysis. Data streams have special processing requirements, due to its unique characteristics compared with traditional databases. It is impossible to store a complete unbounded stream, so registered queries are imposed over summary structures (e.g., synopses [15]), thus the returned query answers are approximate. Because of the limitations on storage and performance, backtracking over streaming data is not allowed, and online algorithms are restricted to making only one pass over streaming data. Till now, a large amount of works have investigated these newly raised research issues. Some of them are related to models and languages (see [47] for a survey), some focus on continuous query processing problems, e.g., load shedding, join problems and efficient window-based operators [17], and many concentrate on data stream mining [36, 56, 85], and so on.

2.3 Information loss metrics

The anonymization problem calls for the enforcement of privacy principle (e.g., k-anonymity, ℓ-diversity, and t-closeness) on a data set, while sacrificing as little of the information in the data as possible. To quantify the information quality compromised for the sake of privacy, we need an appropriate information loss metric. Past literature has proposed various metrics, such as the *Classification Metric* [43] and the *Discernibility Metric* [19]. The best metric to use depends

on the intended use of the data. We assume that the anonymized data is to be used for multiple purposes, which may not be known in advance; hence we adopt a General Loss Metric (GLM) [26, 40, 43, 83].

Let $QI = \{A_1, A_2, \ldots, A_d\}$ and $\mathcal{G}$ be an EC. For a numerical attribute $NA \in QI$, let $[\mathcal{L}_{NA}, \mathcal{U}_{NA}]$ be its domain range and $[l^{\mathcal{G}}_{NA}, u^{\mathcal{G}}_{NA}]$ the minimum sub-range containing all its values in $\mathcal{G}$; then the information loss with respect to NA in $\mathcal{G}$ is defined as:

$$\mathcal{IL}_{NA}(\mathcal{G}) = \frac{u^{\mathcal{G}}_{NA} - l^{\mathcal{G}}_{NA}}{\mathcal{U}_{NA} - \mathcal{L}_{NA}}$$

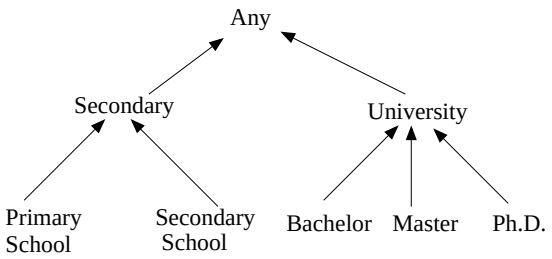


Figure 2.1: Domain generalization hierarchy of education

For a categorical attribute CA, we assume a generalization hierarchy $\mathcal{H}_{CA}$ over its domain. Figure 2.1 illustrates such an example, where the leaves represent the specific values in the domain of attribute `education`, and each internal node represents a generalized value of all its descendants. If a is the lowest common ancestor in $\mathcal{H}_{CA}$ of all CA values in $\mathcal{G}$, then the information loss with respect to CA in $\mathcal{G}$ is defined as:

$$\mathcal{IL}_{CA}(\mathcal{G}) = \begin{cases} 0, & |\mathsf{Leaves}(a)| = 1 \\ \frac{|\mathsf{Leaves}(a)|}{|\mathsf{Leaves}(\mathcal{H}_{CA})|}, & \text{otherwise} \end{cases}$$

where $\mathsf{Leaves}(a)$ is the set of leaves under the subtree of $\mathcal{H}_{CA}$ rooted at a, and $\mathsf{Leaves}(\mathcal{H}_{CA})$ is the total set of leaves in $\mathcal{H}_{CA}$. The total information loss of $\mathcal{G}$ is then:

$$\mathcal{IL}(\mathcal{G}) = \sum_{i=1}^{d} w_i \times \mathcal{IL}_{A_i}(\mathcal{G})$$

where w_i is the weight of A_i and $\sum_{i=1}^{d} w_i = 1$. In our experiments, we treat each A_i as equally important, hence assign $w_i = 1/d$. The total information loss on a database table $\mathcal{DB}$, partitioned into a set $S_{\mathcal{G}}$ of ECs, is defined as:

$$\mathcal{AIL}(S_{\mathcal{G}}) = \frac{\sum_{\mathcal{G} \in S_{\mathcal{G}}} |\mathcal{G}| \times \mathcal{IL}(\mathcal{G})}{|\mathcal{DB}|}$$

2.4 Summary

This chapter studies related anonymization methods, briefly discusses the data streams, and introduces the information loss measure. These form the background knowledge of our thesis.

CHAPTER 3

SABRE: A SENSITIVE ATTRIBUTE BUCKETIZATION AND REDISTRIBUTION FRAMEWORK FOR t-CLOSENESS

Organizations such as ministries or hospitals regularly release microdata (e.g., census data or medical records) to serve the public benefits. However, as discussed in Chapter 1, such data are vulnerable to *linking attack*. Past research has striven to define the *privacy principle* to limit it. The main highlights of the efforts consist of the k-anonymity [66, 67], ℓ-diversity [57], and, most recently, the *t*-closeness [52] models. k-anonymity successfully protects against *identity disclosure*, but suffers from homogeneous attack (see Section 1.1.1). While ℓ-diversity [57] has addressed the limitation existing in k-anonymity, it is still vulnerable to *skewness* and *similarity* attacks (see Section 1.1.2). The t-closeness model takes a step forward in privacy enforcement than its predecessors. In spite of that, all the existing t-closeness schemes are straightforwardly extended from k-anonymity instantiations, lacking a customization towards the specific features of the t-closeness model. To fill this gap, in this chapter we propose SABRE, a Sensitive Attribute Bucketization and REdistribution framework for t-closeness.

3.1 Introduction

The t-closeness model aims to forestall the type of attacks against ℓ-diversity (i.e., *skewness* and *similarity* attacks), by requiring that the $\mathcal{SA}$ distribution in any EC differs from its overall distribution by *at most* a given threshold t, according to an appropriate distance metric. The value of t constrains the *additional* information an adversary *gains* after seeing a single EC, measured with respect to the information provided by the full released table. The t-closeness guarantee directly protects against a *skewness* attack, while it also provides defense against a *similarity* attack, depending on the extent to which semantic similarity exists among the $\mathcal{SA}$ values in the whole table [52].

The t-closeness model poses the problem of bringing a microdata table to a form that complies with it while degrading data quality as little as possible. This problem is *distinct* from those posed by other privacy models. Each model poses a particular *tradeoff* between privacy and information quality, which needs to be resolved in an effective and efficient manner. However, the two extant schemes for t-closeness [52, 53] are extensions of algorithms designed for k-anonymity; they employ either the *Incognito* [48] or the *Mondrian* [49] technique for k-anonymization, merely adding to them the extra condition that the produced ECs should satisfy t-closeness. Still, a good t-closeness anonymization does not[1] necessarily derive from a good k-anonymization. Thus, unfortunately, the techniques in [52, 53] limit the effectiveness of achieving t-closeness by building themselves on top of k-anonymizations, and fail in terms of efficiency by performing too many brute-force t-closeness satisfaction checks. The question of an algorithm tailored for t-closeness-abiding anonymization remains open.

[1] An analogous observation was made with respect to the particular problem posed by ℓ-diversity in [41].

Therefore, we provide SABRE, a Sensitive Attribute Bucketization and REdistribution framework for t-closeness. SABRE operates in two phases. First, it partitions a table into buckets of similar $\mathcal{SA}$ values in a greedy fashion. Then, it redistributes tuples from each bucket into dynamically configured ECs. Following [52, 53], we employ the Earth Mover's Distance (EMD) as a measure of closeness between distributions, and utilize a property of this measure to facilitate our approach. Namely, a tight *upper bound* for the EMD of the distribution in an EC from the overall distribution can be derived as a function of localized upper bounds for each bucket, provided that the tuples in the EC are picked *proportionally* to the sizes of the buckets they hail from. Furthermore, we prove that if the bucket partitioning obeys t-closeness, then the derived ECs also abide to t-closeness. We develop two SABRE instantiations. The former, SABRE-AK focuses on efficiency. The latter, SABRE-KNN trades some efficiency for information quality. Our extensive experimental evaluation demonstrates that both instantiations achieve information quality superior to schemes that extend algorithms customized for k-anonymity to t-closeness, while SABRE-AK is much faster than them as well.

The rest of this chapter is organized as follows. In the next section, we discuss the Earth Mover's Distance. Section 3.3 introduces an observation from which SABRE is derived. We propose SABRE framework and outline its two instantiations in Section 3.4. In section 3.5, we present the results of an extensive performance study. We discuss our findings in Section 3.6 and conclude this chapter in Section 3.7.

3.2 The earth mover's distance metric

t-closeness model postulates that the $\mathcal{SA}$ distribution in any EC differ from that

in the whole table by no more than a threshold t. Neither the Kullback-Leibler (KL) nor the variational distance is appropriate for evaluating the difference of two distributions, as they do not consider semantic relationships of $\mathcal{SA}$ values [52]. Here, we adopt the same metric as [52]—Earth Mover's Distance [65], to measure the difference between two distributions.

The *Earth Mover's Distance* (EMD) is suggested as a metric for quantifying the difference between distributions. Intuitively, it views one distribution as a mass of *earth* piles spread over a space, and the other as a collection of *holes*, in which the mass fits, over the same space. The EMD between the two is defined as the minimum *work* needed to fill the holes with earth, thereby *transforming* one distribution to the other.

Let $\mathcal{P} = (p_1, p_2, \ldots, p_m)$ be the distribution of "holes", $\mathcal{Q} = (q_1, q_2, \ldots, q_m)$ that of "earth", d_{ij} the *ground distance* of q_i from p_j, and $F = [f_{ij}], f_{ij} \geq 0$ a flow of mass of earth moved from element q_i to p_j, $1 \leq i, j \leq m$. The EMD is the *minimum* value of the work required to transform $\mathcal{Q}$ to $\mathcal{P}$ by F:

$$WORK(\mathcal{P}, \mathcal{Q}, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \times f_{ij}$$

For the chapter to be self-contained, in the following, we present the EMD formulas given in [52].

In case of a numerical $\mathcal{SA}$, let its *ordered* domain be $\{v_1, v_2, \ldots, v_m\}$, where v_i is the i^{th} smallest value ($\mathcal{P}$ and $\mathcal{Q}$ are distributions over these values). The distance between two values v_i, v_j in this domain is defined by the number of values between them in the total order, as $\frac{|i-j|}{m-1}$. Then the minimal work for transforming $\mathcal{Q}$ to $\mathcal{P}$ can be calculated by sequentially satisfying the *earth* needs of each *hole* element, moving earth from/to its immediate neighbor pile [52]. Thus, the EMD between $\mathcal{P}$ and $\mathcal{Q}$ is defined as:

$$EMD(\mathcal{P}, \mathcal{Q}) = \frac{1}{m-1} \sum_{i=1}^{m-1} \left| \sum_{j=1}^{i} (q_j - p_j) \right|$$

In case of a categorical $\mathcal{SA}$, we assume a generalization hierarchy $\mathcal{H}$ over its domain. For example, Figure 3.1 depicts a hierarchy of *respiratory and digestive diseases*. The distance between two (leaf) values v_i and v_j is defined as $\frac{h(v_i,v_j)}{h(\mathcal{H})}$, where $h(\mathcal{H})$ is the height of $\mathcal{H}$, and $h(v_i, v_j)$ that of the lowest common ancestor of v_i and v_j in $\mathcal{H}$. To define EMD, we first define the following recursive function of the collective *extra earth* residing among the leaves under node n in $\mathcal{H}$.

$$extra(n) = \begin{cases} q_i - p_i, & \text{if } n \text{ is a leaf } v_i \\ \sum_{c \in child(n)} extra(c), & \text{otherwise} \end{cases}$$

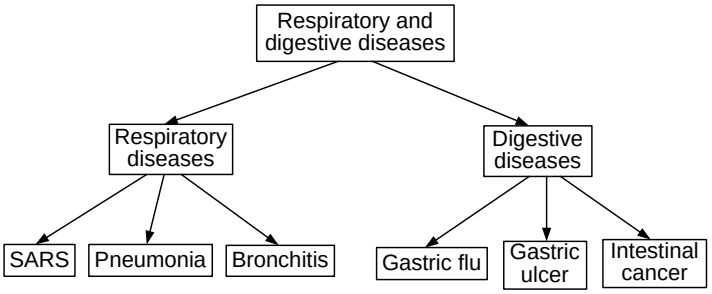


Figure 3.1: The hierarchy for disease

The value of $extra(n)$ denotes the exact amount of earth that should be moved in/out of node n. Furthermore, we define the accumulated amount of earth to be moved inwards and outwards for an *internal* node of $\mathcal{H}$:

$$\begin{aligned} neg_e(n) &= \sum_{c \in child(n) \wedge extra(c) < 0} |extra(c)| \\ pos_e(n) &= \sum_{c \in child(n) \wedge extra(c) > 0} extra(c) \end{aligned}$$

Then the *minimum* of the above quantities signifies the cost of all *pending* earth movements *among* the leaves under node n, after their *cumulative* earth excess/deficit has been corrected:

$$cost(n) = \frac{h(n)}{h(\mathcal{H})} \min(pos_e(n), neg_e(n))$$

Where $h(n)$ is the height of node n in the hierarchy of $\mathcal{H}$. Then, the total EMD between $\mathcal{P}$ and $\mathcal{Q}$ is:

$$EMD(\mathcal{P}, \mathcal{Q}) = \sum_n cost(n)$$

where n is a non-leaf node in $\mathcal{H}$.

Name	Weight	Age	Disease
Mike	60	40	SARS
Alice	70	50	intestinal cancer
John	60	60	pneumonia
Bob	50	50	bronchitis
Beth	80	50	gastric flu
Carol	70	70	gastric ulcer

Table 3.1: Patient records

EC	Weight	Age	Disease
1	[50-60]	[40-60]	SARS
	[50-60]	[40-60]	pneumonia
	[50-60]	[40-60]	bronchitis
2	[70-80]	[50-70]	intestinal cancer
	[70-80]	[50-70]	gastric flu
	[70-80]	[50-70]	gastric ulcer

Table 3.2: 3-diverse published table

Example 3.1. *Assume Table 3.1 is the input table,* $\{$`weight, age`$\}$ *the* QI*,* `disease` *the* $\mathcal{SA}$*, and Table 3.2 the published table. Let* R*,* D *and* RD *represent* respiratory diseases*,* digestive diseases*, and* respiratory and digestive diseases*, respectively. The* $\mathcal{SA}$ *distribution in Table 3.1 is* $\mathcal{P} = (\frac{1}{6}, \frac{1}{6}, \ldots, \frac{1}{6})$*, while that in EC 1 of Table 3.2 is* $\mathcal{Q} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0)$*. Then* $extra(SARS) = extra(pneumonia) = extra(bronchitis) = \frac{1}{6}$*. Thus,* $extra(R) = \frac{1}{2}$*,* $pos_e(R) = \frac{1}{2}$*, and* $neg_e(RD) = 0$*, hence* $cost(R) = 0$*. Likewise,* $extra(D) = -\frac{1}{2}$ *and* $cost(D) = 0$*. In effect,* $extra(RD) = 0$*, and* $pos_e(RD) = neg_e(RD) = \frac{1}{2}$*. Therefore,* $cost(RD) = 1 \times \min(pos_e(RD), neg_e(RD)) = \frac{1}{2}$*, and* $EMD(\mathcal{P}, \mathcal{Q}) = cost(R) + cost(D) + cost(RD) = 0.5$.

3.3 Observations and challenges

This section describes the observations, from which our SABRE framework is derived. Table 3.3 gathers together the notations we will use throughout the chapter.

Notation	Denotation
$\mathcal{DB}$	A microdata table (original table)
$\mathcal{SA}$	The sensitive attribute in $\mathcal{DB}$
$\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$	The domain of $\mathcal{SA}$
N_i	The number of tuples with v_i in $\mathcal{DB}$
$p_i = N_i/\lvert\mathcal{DB}\rvert$	The distribution of value v_i in $\mathcal{DB}$
$\mathcal{P} = (p_1, p_2, \ldots, p_m)$	Overall distribution of $\mathcal{SA}$ in $\mathcal{DB}$
$\mathcal{G}$	An equivalence class
$\mathcal{Q} = (q_1, q_2, \ldots, q_m)$	The distribution of $\mathcal{SA}$ in $\mathcal{G}$

Table 3.3: Employed notations

SABRE consists of two phases. In the first one, *bucketization*, it partitions $\mathcal{DB}$ into a set of buckets, so that each $\mathcal{SA}$ value appears in only one bucket, defined as follows.

Definition 3.1 (bucket partition)**.** *Given a table $\mathcal{DB}$, sensitive attribute $\mathcal{SA}$, we say that a set of buckets φ forms a* bucket partition *of $\mathcal{DB}$ if and only if $\bigcup_{\forall \mathcal{B} \in \varphi} \mathcal{B} = \mathcal{DB}$ and each $\mathcal{SA}$ value appears in* exactly one *bucket.*

In the second phase, *redistribution*, SABRE reallocates tuples from buckets to ECs. For the sake of exposition, we first consider the requirement that the number of tuples assigned to an EC from a certain bucket is proportional to that bucket's size. This *proportionality requirement* is defined as follows.

Definition 3.2 (proportionality requirement)**.** *Given a table $\mathcal{DB}$ and a bucket partition thereof φ, assume that an EC, $\mathcal{G}$, is formed with a subset of tuples x_i from bucket $\mathcal{B}_i \in \varphi$, $i = 1, 2, \ldots, |\varphi|$. $\mathcal{G}$ abides to the* proportionality requirement *with respect to φ, if and only if the sizes of x_i are proportional to those of $\mathcal{B}_i$, i.e., $|x_1| : |x_2| : \ldots : |x_{|\varphi|}| = |\mathcal{B}_1| : |\mathcal{B}_2| : \ldots : |\mathcal{B}_{|\varphi|}|$.*

Assume we create a partitioning $\varphi' = \{b_1, b_2, \ldots, b_m\}$, in which bucket b_i includes those and only those tuples in $\mathcal{DB}$ that have $\mathcal{SA}$ value v_i. Then we

select x_i tuples from bucket b_i, $i = 1, 2, \ldots, |\varphi|$, to form an EC $\mathcal{G}$. In this case, if $\mathcal{G}$ follows the proportional requirement with respect to φ', then it also holds that $|x_1| : |x_2| : \ldots : |x_m| = N_1 : N_2 : \ldots : N_m$, where $N_i = |b_i|$ is the number of tuples with $\mathcal{SA}$ value v_i in $\mathcal{DB}$. In effect, $\mathcal{G}$ is 0-close to $\mathcal{DB}$; thus, an adversary gains no extra information by seeing $\mathcal{G}$. Still, a complete enforcement of 0-closeness for all ECs would severely degrade information quality. This is not what we aim for; we wish to allow for some loss of privacy, delimited by the t-closeness constraint, in order to preserve more in terms of information quality. Thus, we need to opt for a more flexible arrangement in our scheme.

To that end, we can start out with buckets of *more than one* distinct $\mathcal{SA}$ value. We slice $\mathcal{DB}$ into an alternative bucket partition $\varphi = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{|\varphi|}\}$, in which each bucket $\mathcal{B}_i$ may contain multiple *semantically close* $\mathcal{SA}$ values. In this case, an EC $\mathcal{G}$ that satisfies the proportionality requirement with respect to the buckets in φ does *not* necessarily obey the relationship $|z_1| : |z_2| : \ldots : |z_m| = N_1 : N_2 : \ldots : N_m$, where z_i is the set of tuples with $\mathcal{SA}$ value v_i in $\mathcal{G}$, $i = 1, 2, \ldots, m$. After all, when we pick tuples from a bucket $\mathcal{B}_i$ to form $\mathcal{G}$, we do *not* discriminate between different $\mathcal{SA}$ values. The following example illustrates the two EC compositions described above.

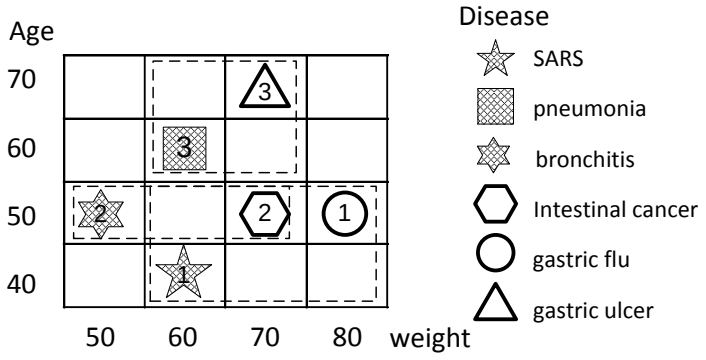


Figure 3.2: Information quality under SABRE

Example 3.2. *Consider Table 3.1, where* {`weight, age`} *is the QI, and* `disease` *is the* $\mathcal{SA}$*. Figure 3.2 shows the QI-space and the distribution of tuples from Table 3.1 (each QI attribute corresponds to a dimension). A bucket partition* φ' *of this table could consist of six buckets of one tuple each,* $b_1, b_2, \ldots, b_6$*, with* $\mathcal{SA}$ *values* SARS*,* pneumonia*,* bronchitis*,* intestinal cancer*,* gastric flu*, and* gastric ulcer*, respectively. Taking one tuple from each of these buckets, we could build a single EC of* 0*-close privacy. Still, such an EC covers the entire QI-space, incurring high information loss. Another bucket partition could consist of two buckets of three semantically similar tuples each,* $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$*, with* SARS*,* pneumonia*, and* bronchitis *in bucket* $\mathcal{B}_1$ *and the rest in* $\mathcal{B}_2$*. We can then build three ECs, by taking one tuple from each of these buckets, as shown in Figure 3.2. Tuples in the same EC are labeled by the same number in the figure. This EC partitioning achieves better information quality, as the minimum bounding boxes of ECs in QI-space are smaller.*

An equivalence class $\mathcal{G}$ constructed from a looser bucket partition achieves higher information quality, but is no longer 0-close. Still, it suffices to construct it in a manner that obeys t-closeness for a given t. In Example 3.2, all three ECs generated from φ are $\frac{1}{3}$-close with respect to the distribution of `disease` in Table 3.1, hence satisfying t-closeness for $t \geq \frac{1}{3}$.

Following the above observations, SABRE first partitions tuples according to their $\mathcal{SA}$ values, and then redistributes the tuples to ECs. In order to ensure t-closeness and good information quality, we need to address the following questions:

1. How should we partition $\mathcal{SA}$ values into buckets? How many buckets should we generate to ensure t-closeness?

2. How many ECs should we generate? How should we choose tuples from

each bucket to form an EC?

Next we present our approaches to these questions.

3.4 The SABRE framework

The SABRE framework consists of two phases: first, a *bucketization* phase partitions the microdata into a set of buckets of similar $\mathcal{SA}$ values (Section 3.4.1), then a *redistribution* phase dynamically determines EC sizes (Section 3.4.2). Section 3.4.3 puts the above phases together and summarizes the SABRE framework.

3.4.1 SABRE's bucketization scheme

We commence the presentation of our bucketization scheme with a property of the proportionality requirement.

Consider a categorical $\mathcal{SA}$ with the domain hierarchy in Figure 3.1. Assume that a table $\mathcal{DB}$ contains 50 tuples with *SARS*, 30 with *Pneumonia*, 20 with *Bronchitis*, 40 with *Gastric flu*, 20 with *Gastric ulcer*, and 20 with *Intestinal cancer*. Suppose a bucket partition $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$ of $\mathcal{DB}$, where $\mathcal{B}_1$ contains all tuples with *SARS*, *Pneumonia* and *Bronchitis*, and $\mathcal{B}_2$ includes tuples with the remaining three $\mathcal{SA}$ values. The overall $\mathcal{SA}$ distribution is $\mathcal{P} = \left(\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right)$. Then an EC, $\mathcal{G}$, with 10 tuples from $\mathcal{B}_1$ and 8 tuples from $\mathcal{B}_2$ satisfies the proportionality requirement with respect to φ. For instance, $\mathcal{G}$ may comprise 4 tuples with *SARS*, 2 with *Pneumonia*, 4 with *Bronchitis*, 0 with *Gastric flu*, 4 with *Gastric ulcer*, and 4 with *Intestinal cancer*. Then the $\mathcal{SA}$ distribution in $\mathcal{G}$ is $\mathcal{Q} = \left(\frac{4}{18}, \frac{2}{18}, \frac{4}{18}, 0, \frac{4}{18}, \frac{4}{18}\right)$.

Now, we can divide the elements (i.e., holes) of $\mathcal{P}$ in two subsets, $H_1 = \left\{\frac{5}{18}, \frac{3}{18}, \frac{2}{18}\right\}$ and $H_2 = \left\{\frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right\}$, and, likewise, the piles of earth in $\mathcal{Q}$ in

$E_1 = \left\{\frac{4}{18}, \frac{2}{18}, \frac{4}{18}\right\}$ and $E_2 = \left\{0, \frac{4}{18}, \frac{4}{18}\right\}$, corresponding to the division of $\mathcal{SA}$ values into respiratory diseases in $\mathcal{B}_1$ and digestive diseases in $\mathcal{B}_2$. Then the volume of holes in H_1 is $\frac{5}{18} + \frac{3}{18} + \frac{2}{18} = \frac{10}{18}$, equal to the volume of earth in E_1, $\frac{4}{18} + \frac{2}{18} + \frac{4}{18} = \frac{10}{18}$. Likewise, the volume of holes in H_2 equals that of earth in E_2. In effect, the transformation from $\mathcal{Q}$ to $\mathcal{P}$ can be decomposed in two *independent* subtasks: filling the holes in H_1 with earth from E_1, and those in H_2 with earth from E_2. We name such a subtask *earth transportation in a bucket*, defined as follows.

Definition 3.3 (earth transportation in a bucket)**.** *Assume a bucket partition φ of a table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, and any equivalence class $\mathcal{G}$ that follows the proportionality requirement with respect to φ. Without loss of generality, assume bucket $\mathcal{B} \in \varphi$ contains the $\mathcal{SA}$ values $v_1, v_2, \ldots, v_j$. Then,* earth transportation in $\mathcal{B}$ with regard to $\mathcal{G}$ *is the transformation from $(q_1, q_2, \ldots, q_j)$ to $(p_1, p_2, \ldots, p_j)$, where q_i is the distribution of v_i in $\mathcal{G}$ and p_i is the distribution of v_i in $\mathcal{DB}$, $i = 1, 2, \ldots, j$.*

We denote the *cost* of this earth transportation in bucket $\mathcal{B}$ with regard to EC $\mathcal{G}$ as $CET(\mathcal{B}, \mathcal{G})$. Once a table $\mathcal{DB}$ is given, $p_1, p_2, \ldots, p_j$ are fixed. But the values of $q_1, q_2, \ldots, q_j$ depend on the EC $\mathcal{G}$ at hand. For instance, in our running scenario, the distribution of the 10 tuples from $\mathcal{B}_1$ in $\mathcal{G}$ among values *SARS*, *Pneumonia*, and *Bronchitis* is $q_1 = \frac{4}{18}$, $q_2 = \frac{2}{18}$, and $q_3 = \frac{4}{18}$. If the 10 tuples are all *Bronchitis*, then $q_1 = q_2 = 0$ and $q_3 = \frac{10}{18}$. Actually, they could be any 10 tuples from $\mathcal{B}_1$. Still, we are interested in the worst-case value of $CET(\mathcal{B}, \mathcal{G})$ over all possible ECs following the proportionality requirement to φ. Thus, we define an upper bound of $CET(\mathcal{B}, \mathcal{G})$ as follows.

Definition 3.4 (upper-bound cost in a bucket)**.** *Assume a bucket partition φ of a table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, and a bucket $\mathcal{B} \in \varphi$. Then, we define*

$CET^U_{\mathcal{B}}$*, the* upper-bound cost of earth transportation in $\mathcal{B}$*, as the highest possible value of* $CET(\mathcal{B}, \mathcal{G})$ *over* all possible *equivalence classes* $\mathcal{G}$ *that follow the proportionality requirement to* φ*.*

$$CET^U_{\mathcal{B}} = \max_{\forall \mathcal{G}}\{CET(\mathcal{B}, \mathcal{G})\}$$

In the following, we present three theorems that form the foundation of SABRE. Let φ be a bucket partition of table $\mathcal{DB}$, and $\mathcal{G}$ be an EC following proportionality requirement with respect to φ. Theorem 3.1 formalizes the intuition gained from the above scenario—it essentially tells us that earth transportation in a bucket can be independent from that of any other bucket. Given a bucket $B \in \varphi$, theorem 3.2 determines $CET^U_{\mathcal{B}}$. Based on the above two, theorem 3.3 states that we can compute the cost of transforming the $\mathcal{SA}$ distribution in $\mathcal{G}$ to that in $\mathcal{DB}$, by the summation of the upper bounds related with all buckets in φ.

Theorem 3.1 (Independence)**.** *Let* $\mathcal{G}$ *be an EC that follows the proportionality requirement with respect to a bucket partition* φ *of table* $\mathcal{DB}$ *with sensitive attribute SA. Given any bucket* $\mathcal{B} \in \varphi$*, the earth transportation in* $\mathcal{B}$ *with regard to* $\mathcal{G}$ *is independent from buckets in* $\varphi \setminus \{\mathcal{B}\}$*.*

Proof. Without loss of generality, assume that $\mathcal{B}$ contains tuples with $\mathcal{SA}$ values of $v_1, v_2, \ldots, v_j$, and let z_i be the set of tuples in $\mathcal{G}$ with $\mathcal{SA}$ value of v_i, $i = 1, 2, \ldots, j$. We consider $\{p_1, p_2, \ldots, p_j\}$, $p_i = \mathcal{N}_i/|\mathcal{DB}|$, as the set of holes, and $\{q_1, q_2, \ldots, q_j\}$, $q_i = |z_i|/|\mathcal{G}|$, as the piles of earth. Given that $\mathcal{G}$ follows the proportionality requirement with respect to φ, the number of tuples from $\mathcal{B}$ assigned to $\mathcal{G}$ is $\sum_{i=1}^{j} |z_i| = \frac{|\mathcal{B}|}{|\mathcal{DB}|} \cdot |\mathcal{G}|$. Then,

$$\sum_{i=1}^{j} q_i = \frac{1}{|\mathcal{G}|}\sum_{i=1}^{j} |z_i| = \frac{|\mathcal{B}|}{|\mathcal{DB}|} = \frac{1}{|\mathcal{DB}|}\sum_{i=1}^{j} N_i = \sum_{i=1}^{j} p_i$$

Therefore, the volume of earth equals the volume of holes, hence earth transportation between them can be done locally, i.e., the transformation from $(q_1, q_2,$

$\ldots, q_j)$ to $(p_1, p_2, \ldots, p_j)$ can be independent from the earth transportation of any other bucket in $\varphi \setminus \{\mathcal{B}\}$. □

Theorem 3.2 (Upper bound)**.** *Let φ be a bucket partition of table $\mathcal{DB}$ with sensitive attribute SA; assume that bucket $\mathcal{B} \in \varphi$ contains $\mathcal{SA}$ values of v_1, v_2, …, v_j. Then $CET^U_{\mathcal{B}}$, the upper-bound cost of earth transportation in $\mathcal{B}$, is determined as follows:*

- *For a categorical $\mathcal{SA}$,*

$$CET^U_{\mathcal{B}} = \frac{h(n)}{h(\mathcal{H})} \cdot \left(\sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\} \right)$$

  , where $\mathcal{H}$ is the domain hierarchy of $\mathcal{SA}$ and n is the lowest common ancestor of $v_1, \ldots, v_j$.

- *For a numerical $\mathcal{SA}$,*

$$CET^U_{\mathcal{B}} = \max_{\ell=1,2,\ldots,j} \left\{ \sum_{i=1}^{j} d_{\ell i} \times p_i \right\}$$

  , where $d_{\ell i}$ is the distance between v_ℓ and v_i.

Proof. Again, we consider $\{p_1, p_2, \ldots, p_j\}$ as a collection of holes, and $\{q_1, q_2, \ldots, q_j\}$ as piles of earth, where q_i is the distribution of v_i in $\mathcal{G}$ (an EC following proportionality requirement with respect to φ). By Theorem 3.1, $\sum_{i=1}^{j} p_i = \sum_{i=1}^{j} q_i$. **Categorical** $\mathcal{SA}$. We divide the set of holes into two subsets: The subset of holes "missing earth", $H_1 = \{p_\ell | p_\ell > q_\ell, 1 \leq \ell \leq j\}$, and that of holes "in excess of earth", $H_2 = \{p_\ell | p_\ell \leq q_\ell, 1 \leq \ell \leq j\}$. Likewise, we separate the set of earth-piles in two corresponding subsets: that of "deficient" piles, $E_1 = \{q_\ell | p_\ell > q_\ell, 1 \leq \ell \leq j\}$, and that of "superfluous" piles, $E_2 = \{q_\ell | p_\ell \leq q_\ell, 1 \leq \ell \leq j\}$. The earth transportation in $\mathcal{B}$ involving $\mathcal{G}$ is done by two steps.

In the first step, we fill up hole p_i with earth q_i, $i = 1, 2, \ldots, j$. Since the distance between p_i and q_i is 0, the cost of this step is 0. Still, all the earth of deficient piles in E_1 is used up; thus, in order to fill up the holes missing earth in H_1, we need extra earth of $neg_e(n) = \sum_{\forall q_\ell \in E_1}(p_\ell - q_\ell)$. Symmetrically, the holes in excess of earth in H_2 are completely filled, and the superfluous earth-piles in E_2 have extra earth of exactly $pos_e(n) = \sum_{\forall q_\ell \in E_2}(q_\ell - p_\ell)$. From the independence of earth transportation within $\mathcal{B}$, i.e., from $\sum_{i=1}^{j} p_i = \sum_{i=1}^{j} q_i$, it follows that $neg_e(n) = pos_e(n)$, as we would expect.

In the second step, we have to move $neg_e(n)$ earth from the superfluous piles in E_2 to the holes missing earth in H_1. Since the distance between any two elements in $\{v_1, \ldots, v_j\}$ is at most $\frac{h(n)}{h(\mathcal{H})}$, the cost of the whole earth movement is at most $\frac{h(n)}{h(\mathcal{H})} \cdot neg_e(n)$. However, $neg_e(n) \leq \sum_{\forall p_\ell \in H_1} p_\ell$. Besides, because there is at least one hole in H_2 (i.e., at least one superfluous pile), it follows that $\sum_{\forall p_\ell \in H_1} p_\ell \leq \sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\}$. Putting it all together, we get $CET(\mathcal{B}, \mathcal{G}) \leq \frac{h(n)}{h(\mathcal{H})} \cdot \left(\sum_{i=1}^{j} p_i - \min\{p_1, p_2, \ldots, p_j\}\right)$.

Numerical $\mathcal{SA}$. We scan the holes $\{p_1, p_2, \ldots, p_j\}$ sequentially, and fill up every hole in need of earth that we encounter. For each such hole, we use earth from its *nearest* pile, resolving ties arbitrarily. If the nearest pile is used up, we move earth from its second nearest pile. This continues until the hole is filled up. Thus, we transfer earth in the most affordable way. Let q_ℓ, $1 \leq \ell \leq j$, be the last pile that the process takes earth from. After q_ℓ is used up, all the holes are filled. Using q_ℓ, we divide the holes $\{p_1, p_2, \ldots, p_j\}$ into two groups: H_1, holes that are entirely filled by earth hailing from q_ℓ, and H_2, the rest. We name the above as *sequential process*, and denote its cost by seq_c. The following situation, which has all earth concentrated in the ℓ^{th} pile, corresponds to distribution $(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_j)$, where $\tilde{q}_\ell = \sum_{i=1}^{j} p_i$ and $\tilde{q}_i = 0$ if $i \neq \ell$. The cost

of the given situation (by transforming $(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_j)$ to $(p_1, p_2, \ldots, p_j)$) is an upper bound of seq_c. That is because filling a hole in H_1 sequential process has the same cost as the given situation; on the other hand, filling a hole in H_2 the former costs less than the latter, because in the former case some earth is brought to the hole from a pile that is, by the definition of H_2, closer to it than pile q_ℓ. The cost of the given situation is $\sum_{i=1}^{j}(d_{\ell i} \times p_i)$. We cannot know which pile q_ℓ is, thus we consider the worst-case scenario, i.e., the maximum out of all ℓ. In effect, the cost to transform $\{q_1, q_2, \ldots, q_j\}$ to $\{p_1, p_2, \ldots, p_j\}$ is upper-bounded by $\max\limits_{\ell=1,2,\ldots,j}\left\{\sum\limits_{i=1}^{j} d_{\ell i} \times p_i\right\}$.

Both upper bounds are tight. The bounds are reached when the tuples from $\mathcal{B}$ assigned to $\mathcal{G}$ all have the same $\mathcal{SA}$ value v_ℓ. For categorial $\mathcal{SA}$, v_ℓ is the least frequent value among $v_1, v_2, \ldots, v_j$, with $p_\ell = \min\{p_1, p_2, \ldots, p_j\}$. For numerical $\mathcal{SA}$, v_ℓ is the value for which $\sum_{i=1}^{j}(d_{\ell i} \times p_i)$ is maximized. □

Theorem 3.3 (Additivity)**.** *Let $\mathcal{G}$ be* any *EC that follows the proportionality requirement with respect to a bucket partition φ of table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$. Then the EMD of transforming the $\mathcal{SA}$ distribution $\mathcal{Q}$ in $\mathcal{G}$ to the distribution $\mathcal{P}$ in $\mathcal{DB}$ is tightly upper bounded by $\sum\limits_{\forall \mathcal{B} \in \varphi} CET^U_{\mathcal{B}}$.*

Proof. In order to transform $\mathcal{Q}$ to $\mathcal{P}$, we need to carry out the earth transportation in each bucket $\mathcal{B} \in \varphi$. Since these transportations do not affect each other (Theorem 3.1), the EMD required to transform $\mathcal{Q}$ to $\mathcal{P}$ is the sum of their costs, and the upper bound of this sum is the sum of the upper bounds to the individual costs. The bound is tight, and reached when the cost of earth transportation in each bucket $\mathcal{B}$ arrives at its tight upper bound, i.e., all tuples allocated to $\mathcal{G}$ from $\mathcal{B}$ share a same $\mathcal{SA}$ value (Theorem 3.2). □

Example 3.3. *Consider once again our running scenario. We still assume that* SARS*,* pneumonia*, and* bronchitis *are in $\mathcal{B}_1$, and $\mathcal{B}_2$ comprises tuples of the*

remaining three $\mathcal{SA}$ values. We select 10 *tuples from* $\mathcal{B}_1$ *and* 8 *tuples from* $\mathcal{B}_2$ *to form* $\mathcal{G}$*. If the* 10 *tuples from* $\mathcal{B}_1$ *all have $\mathcal{SA}$ value* bronchitis*, then* $(q_1, q_2, q_3) = (0, 0, \frac{10}{18})$ *and the cost of earth transportation in* $\mathcal{B}_1$ *reaches its upper bound* $\frac{1}{2} \cdot [(p_1 + p_2 + p_3) - \min\{p_1, p_2, p_3\}] = \frac{2}{9}$*, where* $(p_1, p_2, p_3) = (\frac{5}{18}, \frac{3}{18}, \frac{2}{18})$*. This upper bound is realized by moving* $\frac{5}{18}$ *earth from* q_3 *to* p_1 *(cost is* $\frac{1}{2} \times \frac{5}{18}$*),* $\frac{3}{18}$ *earth from* q_3 *to* p_2 *(cost is* $\frac{1}{2} \times \frac{3}{18}$*), and* $\frac{2}{18}$ *earth from* q_3 *to* p_3 *(cost is* $0 \times \frac{2}{18}$*). Likewise, if the* 8 *tuples from* $\mathcal{B}_2$ *are all with* gastric ulcer *(or* intestinal cancer*), then the cost of earth transportation in* $\mathcal{B}_2$ *reaches its upper bound* $\frac{1}{2} \cdot [(p_4 + p_5 + p_6) - \min\{p_4, p_5, p_6\}] = \frac{1}{6}$*. With* 10 *tuples of* bronchitis *and* 8 *of* gastric ulcer*, the $\mathcal{SA}$ distribution in* $\mathcal{G}$ *is* $\mathcal{Q} = (0, 0, \frac{10}{18}, 0, \frac{8}{18}, 0)$*. The overall $\mathcal{SA}$ distribution in $\mathcal{DB}$ is* $\mathcal{P} = (\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18})$*. After the earth transportations in* B_1 *and* B_2*,* $\mathcal{Q}$ *is transformed to* $\mathcal{P}$ *with a cost upper bounded by* $CET^U_{\mathcal{B}_1} + CET^U_{\mathcal{B}_2} = \frac{2}{9} + \frac{1}{6} = \frac{7}{18}$.

After the above foundations, we can now discuss the generation of buckets. SABRE partitions $\mathcal{DB}$ hierarchically, based on the $\mathcal{SA}$ values of its tuples, forming a *bucketization tree*. Each node of this tree denotes a bucket containing tuples having a certain subset of $\mathcal{SA}$ values. The leaf nodes of the tree are the buckets that correspond to the actual bucket partition of $\mathcal{DB}$. The tree starts with a single node — the root — which corresponds to the entire table with the whole domain of $\mathcal{SA}$. Then the tree grows in a top-down manner by recursively splitting leaf nodes. In each iteration, we can compute the upper bounded cost of each node/bucket (based on Theorem 3.2). By Theorem 3.3, we determine $\mathcal{U}$, the summation of all the upper bounds. In this way, we select the node that contributes to the largest reduction of $\mathcal{U}$ as the node to be further split. This process terminates when $\mathcal{U}$ becomes smaller than the closeness threshold t. By Theorem 3.3, this termination condition guarantees that the $\mathcal{SA}$ distribution in any EC formed from the final buckets according to the *proportionality requirement*

will not differ from that in $\mathcal{DB}$ by more than t.

We now elaborate on the way we split a leaf bucket/node n. For a categorical $\mathcal{SA}$, let $\mathcal{H}$ be the domain hierarchy of $\mathcal{SA}$. Then each node in the bucketization tree has a corresponding node in $\mathcal{H}$. The root of the tree matches the root of $\mathcal{H}$. When splitting n, its new children are the children of its corresponding node in $\mathcal{H}$.

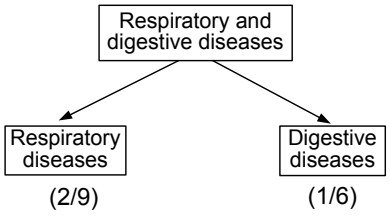


Figure 3.3: Splitting at *root*

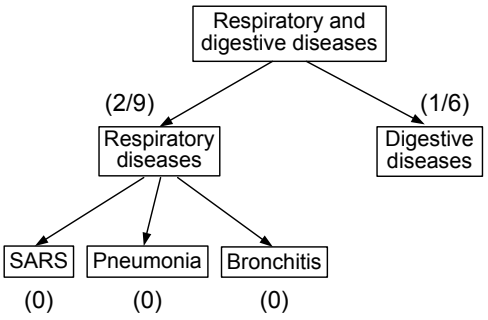


Figure 3.4: Splitting at *respiratory diseases*

Example 3.4. *Let* `disease` *be a categorical $\mathcal{SA}$ with the domain hierarchy of Figure 3.1. Assume a table $\mathcal{DB}$, containing* 5 *tuples with* SARS*,* 3 *with* Pneumonia*,* 2 *with* Bronchitis*,* 4 *with* Gastric flu*,* 2 *with* Gastric ulcer*, and* 2 *with* Intestinal cancer*. The resultant distribution is* $\left(\frac{5}{18}, \frac{3}{18}, \frac{2}{18}, \frac{4}{18}, \frac{2}{18}, \frac{2}{18}\right)$*. Assume a threshold* $t = 0.2$*. We build the bucketization tree in a top-down fashion as follows. At the beginning, the tree comprises the root,* respiratory and digestive diseases*. The upper bound cost of the root is calculated as* $1 \times ((\frac{5}{18} + \ldots + \frac{2}{18} + \frac{2}{18}) - min\{\frac{5}{18}, \ldots, \frac{2}{18}, \frac{2}{18}\}) = 1 \times (1 - \frac{2}{18}) = 0.889$*. Since* $0.889 > 0.2$*, we split it. We add its two children* respiratory diseases, digestive diseases *to the tree, as in Figure 3.3. Now the upper bound cost of node* respiratory diseases *is* $\frac{1}{2} \times \left(\frac{10}{18} - \frac{2}{18}\right) = \frac{2}{9}$*, and that of* digestive diseases *is* $\frac{1}{2} \times \left(\frac{8}{18} - \frac{2}{18}\right) = \frac{1}{6}$*. Since* $\frac{2}{9} + \frac{1}{6}$ *is still larger than* t*, we further split the tree. Splitting* respiratory diseases *would reduce $\mathcal{U}$ by* $\frac{2}{9}$*, whereas splitting* digestive diseases *would reduce it by*

$\frac{1}{6}$*; thus, we opt for the former. We add the three children of node* respiratory diseases *to the tree. Now the sum of upper bounds of all leaf nodes is* $\frac{1}{6} < t$*. Figure 3.4 shows the final tree.*

On the other hand, for a numerical $\mathcal{SA}$, the children of n are dynamically determined. Let lv be the set of $\mathcal{SA}$ values included in n, sorted in ascending order. We split lv to a left child c_1, containing values in lv to the left of the splitting point, and a right child c_2 with the remaining ones. The splitting point is the one that minimizes $CET^U_{c_1} + CET^U_{c_2}$.

Example 3.5. *Let* `salary` *be a numerical* $\mathcal{SA}$ *with values* $1k, 2k, 3k, 4k$*. Assume that a table* $\mathcal{DB}$ *contains* 2 *tuples with* $1k$*,* 3 *with* $2k$*,* 3 *with* $3k$*, and* 2 *with* $4k$*. Then the salary distribution is* $(0.2, 0.3, 0.3, 0.2)$*. We label the four values from* 1 *to* 4 *sequentially. The upper bound cost for a bucketization tree composed only of the root (*$1k-4k$*) is* $d_{11} \times 0.2 + d_{12} \times 0.3 + d_{13} \times 0.3 + d_{14} \times 0.2 = 0 + \frac{1}{3} \times 0.3 + \frac{2}{3} \times 0.3 + 1 \times 0.2 = 0.5$*. If we set the splitting point at* $1k$*, then the left child of* $1k-4k$ *will be* $1k$*, and its right child* $2k-4k$*. The upper bound cost of* $1k$ *is* 0*, and that of* $2k-4k$ *is* $d_{42} \times 0.3 + d_{43} \times 0.3 + d_{44} \times 0.2 = 0.3$*, as shown in Figure 3.5(a). Figures 3.5(b),(c) depict the corresponding trees for splitting along* $2k$ *and* $3k$*. The minimum sum of upper bounds is that in Figure 3.5(b). Thus, for a threshold* $t = 0.25$*, the tree in Figure 3.5(b) is the final tree.*

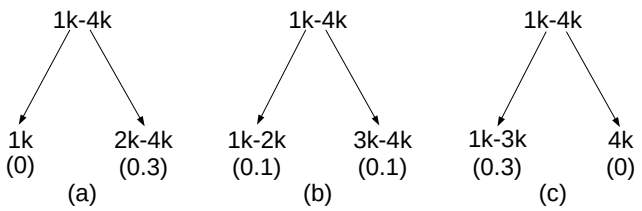


Figure 3.5: Splitting of salary at 1k-4k

Function bucketCat generates buckets for a categorical $\mathcal{SA}$. Input parameter $\mathcal{H}$ is the domain hierarchy of $\mathcal{SA}$; $\mathcal{VP}$ is the list of (v_i, p_i) pairs, the $\mathcal{SA}$ value and its frequency in the whole table, $i = 1, 2, \ldots, m$. $\mathcal{L}$ stores all the leaf nodes of the bucketization tree (steps 1-2). We use a node to represent its corresponding bucket. Step 4 calculates the upper bound cost of the root r. Steps 5-7 compute the potential cost reduction after splitting r (dv denotes the decreased value). The root r is the first node added to $\mathcal{L}$. $\mathcal{U}$, the sum of all upper bounds of the nodes in $\mathcal{L}$, is initialized to CET_r^U (step 8). The leaf node n whose splitting reduces $\mathcal{U}$ at most is split (step 10), n is replaced by its children in $\mathcal{L}$ (steps 11-13), and $\mathcal{U}$ is accordingly reduced (step 14). Steps 17-20 calculate how much $\mathcal{U}$ can be deduced if n's child is split. This process continues iteratively (steps 9-20) until $\mathcal{U} < t$ (steps 15-16). Eventually, each node in $\mathcal{L}$ is associated to a bucket in φ, and φ is returned (steps 21-22).

Function bucketCat ($\mathcal{H}$, $\mathcal{VP}$)

1 Let $\mathcal{L}$ be the set of leaf nodes in the bucketization tree;
2 Initialize $\mathcal{L}$ to be empty;
3 Let r be the root of $\mathcal{H}$;
4 Calculate CET_r^U;
5 **foreach** *child* c *of* r *in* $\mathcal{H}$ **do**
6 calculate CET_c^U;
7 dv_r=$CET_r^U - \sum_{c \in child(r)} CET_c^U$;
8 Add r to $\mathcal{L}$ and initialize $\mathcal{U} = CET_r^U$;
9 **while** $\mathcal{U} \geq t$ **do**
10 Let n be the node in $\mathcal{L}$ with the maximum dv value;
11 Remove n from $\mathcal{L}$;
12 **foreach** *child* c *of* n *in* $\mathcal{H}$ **do**
13 Add c to $\mathcal{L}$;
14 $\mathcal{U} = \mathcal{U} - dv_n$;
15 **if** $\mathcal{U} < t$ **then**
16 break;
17 **foreach** *child* c *of* n **do**
18 **foreach** *child* gc *of* c *in* $\mathcal{H}$ **do**
19 Calculate CET_{gc}^U;
20 $dv_c = CET_c^U - \sum_{gc \in child(c)} CET_{gc}^U$;
21 Let φ be the set of buckets related with nodes in $\mathcal{L}$;
22 Return φ;

Function bucketNum ($\mathcal{VP}$)

1 Let $\mathcal{L}$ be the set of leaf nodes in the bucketization tree;
2 Initialize $\mathcal{L}$ to be empty;
3 Create root r and initialize its $\mathcal{SA}$ values lv_r to be $\mathcal{VP}$;
4 Calculate CET_r^U;
5 Create c_1 and c_2, the left and right children of r;
6 split (r, c_1, c_2);
7 $dv_r = CET_r^U - CET_{c_1}^U - CET_{c_2}^U$;
8 Add r to $\mathcal{L}$ and initialize $\mathcal{U} = CET_r^U$;
9 **while** $\mathcal{U} \geq t$ **do**
10 Find the node n in $\mathcal{L}$ with the maximum value of dv;
11 Remove n from $\mathcal{L}$;
12 **foreach** *child* c *of* n **do**
13 Add c to $\mathcal{L}$;
14 $\mathcal{U} = \mathcal{U} - dv_n$;
15 **if** $\mathcal{U} < t$ **then**
16 break;
17 **foreach** *child* c *of* n **do**
18 Create g_1 and g_2, the left and right children of c;
19 split (c, g_1, g_2);
20 $dv_c = CET_c^U - CET_{g_1}^U - CET_{g_2}^U$;
21 Let φ be the set of buckets related with nodes in $\mathcal{L}$;
22 Return φ;

Similarly, Function bucketNum generates buckets of a numerical $\mathcal{SA}$. Input parameter $\mathcal{VP}$ is the list of $\mathcal{SA}$ values and their frequencies in the whole table, sorted in ascending order of $\mathcal{SA}$ values. Each node in the bucketization tree has a container lv that records all $\mathcal{SA}$ values covered by that node. The root r has lv equal to $\mathcal{VP}$, i.e. r covers all $\mathcal{SA}$ values (step 3). Procedure split calculates the best gain of dividing r into two child nodes (step 6) and r is added as the first node to $\mathcal{L}$ (step 8). CET_r^U (step 4) is assigned as the initial value of $\mathcal{U}$ (step 8). Then, nodes in $\mathcal{L}$ are split iteratively until $\mathcal{U}$ becomes smaller than the closeness threshold t (steps 9-20). In each round, the most cost-reducing node n is chosen from $\mathcal{L}$ to be split (step 10), and replaced by its two children (steps 11-13). The gain of splitting the children of n is also calculated (steps 17-20). The best grandchildren for n are dynamically determined by Procedure split (step 19). Eventually, each node in $\mathcal{L}$ is associated to a bucket, and the set

of all buckets φ is returned (steps 21-22).

Procedure split (n, c_1, c_2)

1 $u_1 = 1.0, u_2 = 1.0$;
2 **foreach** *split point* SP *in* lv_n **do**
3 Clear lv_{c_1} and lv_{c_2};
4 Push all elements in lv_n to the left of SP into lv_{c_1};
5 Allocate all remaining elements of lv_n to lv_{c_2};
6 Calculate $CET^U_{c_1}$ and $CET^U_{c_2}$;
7 **if** $CET^U_{c_1} + CET^U_{c_2} < u_1 + u_2$ **then**
8 $u_1 = CET^U_{c_1}$;
9 $u_2 = CET^U_{c_2}$;
10 $bestP = SP$;
11 Clear lv_{c_1} and lv_{c_2};
12 Push all elements in lv_n to the left of $bestP$ into lv_{c_1};
13 Allocate all the remaining elements of lv_n to lv_{c_2};
14 $CET^U_{c_1} = u_1$;
15 $CET^U_{c_2} = u_2$;

Procedure split dynamically divides a node n into a left child c_1 and a right child c_2. Each possible splitting point is tested, and the one that minimizes $CET^U_{c_1} + CET^U_{c_2}$ is selected (steps 2-10). The elements in lv_n are appropriately assigned to the two children (steps 12-13).

3.4.2 SABRE's redistribution scheme

The bucketization phase delivers a set of buckets φ, such that $\sum_{\forall \mathcal{B} \in \varphi} CET^U_{\mathcal{B}} < t$. To generate an equivalence class $\mathcal{G}$ conforming to the *proportionality requirement*, we need to select $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ tuples from bucket $\mathcal{B}_i$, $i = 1, 2, \ldots, |\varphi|$. However, $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ may not be an integer for some sizes of $\mathcal{G}$ and some $i \in \{1, 2, \ldots, |\varphi|\}$. Setting a constraint to the size of $\mathcal{G}$ so that each $|\mathcal{G}| \cdot \frac{|\mathcal{B}_i|}{|\mathcal{DB}|}$ be an integer may severely limit the allowed EC size, hence defeat the purpose of our study, which is to provide a flexible and quality-aware scheme for t-closeness. For example, assume that $|\mathcal{DB}| = 50,000$ and $\frac{|\mathcal{B}_i|}{|\mathcal{DB}|} = 0.1333$ for some $i \in \{1, 2, \ldots, |\varphi|\}$. Then each EC should have a size of at least $10,000$. Such large ECs generally incur high information loss.

We conclude that we should better relax the proportionality requirement: it suffices that the number of tuples from each bucket $\mathcal{B}_i$ in an EC $\mathcal{G}$ be *approximately* proportional to the size of the bucket. To ensure t-closeness, we determine the size of each EC dynamically. Theorem 3.4 establishes that this is always possible. Before presenting it, we introduce some auxiliary concepts and notations.

Definition 3.5 (φ distribution in an EC)**.** *Let $\mathcal{G}$ be an EC from table $\mathcal{DB}$ with bucket partition φ. The φ distribution in $\mathcal{G}$, denoted by $d(\mathcal{G}, \varphi)$, is $\left(\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}, \ldots, \frac{|x_{|\varphi|}|}{|\mathcal{G}|}\right)$, where x_i is the set of tuples from bucket $\mathcal{B}_i \in \varphi$ in $\mathcal{G}$ and $\bigcup_{i=1}^{|\varphi|} x_i = \mathcal{G}$.*

If we take the whole table $\mathcal{DB}$ as a single EC, then its φ distribution is $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}, \ldots, \frac{|\mathcal{B}_{|\varphi|}|}{|\mathcal{DB}|}\right)$. Furthermore, if $\mathcal{G}$ conforms to the proportionality requirement with respect to φ, then $d(\mathcal{G}, \varphi) = d(\mathcal{DB}, \varphi)$. Given two φ distributions, we *define* d^u_{ij} to be the distance between element i of the former and element j of the latter. Let d_{yz} be the *ground distance* between two $\mathcal{SA}$ values v_y and v_z, and $\mathcal{V}_i$ be the set of $\mathcal{SA}$ values in bucket $\mathcal{B}_i \in \varphi$, then we define our d^u_{ij} metric as follows.

$$d^u_{ij} = \begin{cases} max\{d_{yz} | v_y \in \mathcal{V}_i, v_z \in \mathcal{V}_j\}, & i \neq j \\ 0, & i = j \end{cases}$$

We can transform the $\mathcal{SA}$ distribution $\mathcal{Q}$ in an EC $\mathcal{G}$ to that in the whole table, $\mathcal{P}$ in two steps: First, we transform $\mathcal{Q}$ to $\mathcal{Q}'$, the $\mathcal{SA}$ distribution of an EC $\mathcal{G}'$ that follows the proportionality requirement; then, we transform $\mathcal{Q}'$ to $\mathcal{P}$. Lemma 3.1 proves that $\mathcal{Q}'$ exists and gives the upper bound cost of the transformation from $\mathcal{Q}$ to $\mathcal{Q}'$. Theorem 3.4 builds on Lemma 3.1 and specifies the conditions for EC sizes that satisfy t-closeness.

Lemma 3.1. *Let $\mathcal{G}$ be an EC generated from table $\mathcal{DB}$ with bucket partition φ, and $\mathcal{Q} = (q_1, q_2, \ldots, q_m)$ be the $\mathcal{SA}$ distribution in $\mathcal{G}$. Then there exists an*

$\mathcal{SA}$ distribution $\mathcal{Q}' = (q'_1, q'_2, \ldots, q'_m)$ of an EC following the proportionality requirement with respect to φ, such that the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper bounded by $\mathcal{D} = EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$.

Proof. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the domain of $\mathcal{SA}$. Consider the partition of $\mathcal{V}$, $\overline{\mathcal{V}} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{|\varphi|}\}$, where $\mathcal{V}_i$ is the set of $\mathcal{SA}$ values in $\mathcal{B}_i \in \varphi$. Given $\overline{\mathcal{V}}$, we derive $\overline{\mathcal{Q}} = (\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{|\varphi|})$ and $\overline{\mathcal{Q}'} = (\mathcal{Q}'_1, \mathcal{Q}'_2, \ldots, \mathcal{Q}'_{|\varphi|})$, where $q_y \in \mathcal{Q}_i$ ($q'_y \in \mathcal{Q}'_i$) if and only if $v_y \in \mathcal{V}_i$. Let x_i be the set of tuples from bucket $\mathcal{B}_i$ in $\mathcal{G}$. Then, $\sum_{\forall q_y \in \mathcal{Q}_i} q_y = \frac{|x_i|}{|\mathcal{G}|}$. The φ distribution in $\mathcal{G}$ is $d(\mathcal{G}, \varphi) = \left(\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}, \ldots, \frac{|x_{|\varphi|}|}{|\mathcal{G}|}\right)$, and that in $\mathcal{DB}$ is $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}, \ldots, \frac{|\mathcal{B}_{|\varphi|}|}{|\mathcal{DB}|}\right)$. We need to find a distribution $\mathcal{Q}' = (q'_1, q'_2, \ldots, q'_m)$ so that $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$, $j = 1, 2, \ldots, |\varphi|$. Initially, we set $q'_z = 0$, $z = 1, 2, \ldots, m$. We see $d(\mathcal{DB}, \varphi)$ as a collection of holes to be filled by piles of earth transported from $d(\mathcal{G}, \varphi)$. Moreover, we see $\mathcal{Q}$ and $\mathcal{Q}'$ as piles of earth. The elements $\mathcal{Q}_i \in \overline{\mathcal{Q}}$ ($\mathcal{Q}'_i \in \overline{\mathcal{Q}'}$) can be seen as *clusters* of piles of earth from $\mathcal{Q}$ ($\mathcal{Q}'$). During the earth transportation – from $d(\mathcal{G}, \varphi)$ to $d(\mathcal{DB}, \varphi)$ – whenever δ earth is moved from $\frac{|x_i|}{|\mathcal{G}|}$ to $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|}$ (costing by definition $d^u_{ij} \cdot \delta$), then we also move δ earth from (a pile in) the corresponding cluster $\mathcal{Q}_i$ to (a pile in) cluster $\mathcal{Q}'_j$. Then, after all holes in $d(\mathcal{DB}, \varphi)$ are filled with earth from $d(\mathcal{G}, \varphi)$, the volume of earth in cluster $\mathcal{Q}'_j$ is equal to the volume of hole $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|}$, i.e. $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$, $j = 1, 2, \ldots, |\varphi|$. In other words, $\mathcal{Q}$ is transformed to an $\mathcal{SA}$ distribution $\mathcal{Q}'$ that follows the proportionality requirement with respect to φ. Concerning the cost, we distinguish two cases during the above earth transportation operation:

Case 1: If $i \neq j$, then the cost of earth transportation from (a pile in) cluster $\mathcal{Q}_i$ to (a pile in) cluster $\mathcal{Q}'_j$ is *at most* $d^u_{ij} \cdot \delta$, since d^u_{ij} is, by its definition, the maximum ground distance between *any* pile $q_y \in \mathcal{Q}_i$ and *any* pile $q'_z \in \mathcal{Q}'_j$.

Case 2: If $i = j$, then, at the transportation from (a pile in) cluster $\mathcal{Q}_i$ to (a

pile in) cluster $\mathcal{Q}'_i$, we simply require that earth from pile $q_y \in \mathcal{Q}_i$ only goes to the corresponding pile $q'_y \in \mathcal{Q}'_i$; the ground distance between these piles (q_y and q'_y) is 0, hence the cost of earth transportation is 0 too.

Based on the above two cases, we conclude that the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper-bounded by $\mathcal{D}$. □

Example 3.6. *We return to Example 3.4. Assume that* $t = 0.45$*, so that Figure 3.3 shows the final bucketization tree.* $\mathcal{B}_1$ *relates to* Respiratory diseases *with* $|\mathcal{B}_1| = 10$*.* $\mathcal{B}_2$ *relates to* Digestive diseases*, with* $|\mathcal{B}_2| = 8$*.* $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$ *is a bucket partition of* $\mathcal{DB}$ *with* $d^u_{12} = 1$*. Suppose that* $\mathcal{G}$ *contains* 3 *tuples with* SARS *from* $\mathcal{B}_1$ *and* 2 *tuples with* Gastric flu *from* $\mathcal{B}_2$*. The disease distribution in* $\mathcal{G}$ *then is* $\mathcal{Q} = (q_1, \ldots, q_6) = (\frac{3}{5}, 0, 0, \frac{2}{5}, 0, 0)$*. Initially,* $\mathcal{Q}' = (q'_1, \ldots, q'_6) = (0, \ldots, 0)$*. Thus,* $\overline{\mathcal{Q}} = (\mathcal{Q}_1, \mathcal{Q}_2) = (\{q_1, q_2, q_3\}, \{q_4, q_5, q_6\}) = (\{\frac{3}{5}, 0, 0\}, \{\frac{2}{5}, 0, 0, \})$*, and* $\overline{\mathcal{Q}'} = (\mathcal{Q}'_1, \mathcal{Q}'_2) = (\{q'_1, q'_2, q'_3\}, \{q'_4, q'_5, q'_6\}) = (\{0, 0, 0\}, \{0, 0, 0\})$*. We have* $d(\mathcal{DB}, \varphi) = \left(\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}, \frac{|\mathcal{B}_2|}{|\mathcal{DB}|}\right) = \left(\frac{5}{9}, \frac{4}{9}\right)$*, and* $d(\mathcal{G}, \varphi) = (\frac{|x_1|}{|\mathcal{G}|}, \frac{|x_2|}{|\mathcal{G}|}) = (\frac{3}{5}, \frac{2}{5})$*. To transform* $d(\mathcal{G}, \varphi)$ *to* $d(\mathcal{DB}, \varphi)$*, we move* $\frac{5}{9}$ *earth from* $\frac{|x_1|}{|\mathcal{G}|}$ *to* $\frac{|\mathcal{B}_1|}{|\mathcal{DB}|}$ *(at cost* 0*),* $\frac{2}{5}$ *earth from* $\frac{|x_2|}{|\mathcal{G}|}$ *to* $\frac{|\mathcal{B}_2|}{|\mathcal{DB}|}$ *(cost* 0*), and* $\frac{3}{5} - \frac{5}{9} = \frac{2}{45}$ *from* $\frac{|x_1|}{|\mathcal{G}|}$ *to* $\frac{|\mathcal{B}_2|}{|\mathcal{DB}|}$ *(at cost* $\frac{2}{45}$*). Thus,* $EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi)) = \frac{2}{45}$*. Accordingly, we also move* $\frac{5}{9}$ *earth from* $\mathcal{Q}_1$ *to* $\mathcal{Q}'_1$ *(i.e., from* q_1 *to* q'_1*, at cost* 0*), and* $\frac{2}{5}$ *earth from* $\mathcal{Q}_2$ *to* $\mathcal{Q}'_2$ *(i.e., from* q_4 *to* q'_4*), hence* $\overline{\mathcal{Q}'} = \left(\{\frac{5}{9}, 0, 0\}, \{\frac{2}{5}, 0, 0\}\right)$ *(at cost* 0*). When moving* $\frac{2}{45}$ *earth from* $\mathcal{Q}_1$ *to* $\mathcal{Q}'_2$*, there are multiple choices, we can move* $\frac{2}{45}$ *earth from* q_1 *to* q'_4*,* q'_5*, or* q'_6 *(at cost* $\frac{2}{45}$*). Assume that it is moved from* q_1 *to* q'_4*, then* $\overline{\mathcal{Q}'} = (\{\frac{5}{9}, 0, 0\}, \{\frac{4}{9}, 0, 0\})$*, and* $\frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{\forall q'_z \in \mathcal{Q}'_j} q'_z$*,* $j = 1, 2$*. The resultant* $\mathcal{SA}$ *distribution* $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{4}{9}, 0, 0\right)$ *follows the proportionality requirement with respect to* φ*. Alternatively, if* $\frac{2}{45}$ *earth is moved to* q'_5*, then* $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{2}{5}, \frac{2}{45}, 0\right)$*. If the earth is moved to* q'_6*, then* $\mathcal{Q}' = \left(\frac{5}{9}, 0, 0, \frac{2}{5}, 0, \frac{2}{45}\right)$*. In all three cases, the cost of transforming* $\mathcal{Q}$ *to* $\mathcal{Q}'$ *is* $\frac{2}{45} \leq EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$*.*

Theorem 3.4 (Triangle inequality)**.** *Let $\mathcal{G}$ be an EC from table $\mathcal{DB}$ with bucket partition φ. If $\mathcal{D} + \mathcal{U} \leq t$, where $\mathcal{D} = EMD(d(\mathcal{G}, \varphi), d(\mathcal{DB}, \varphi))$ and $\mathcal{U} = \sum_{\forall \mathcal{B} \in \varphi} CET^U_{\mathcal{B}}$, then the $\mathcal{SA}$ distribution $\mathcal{Q}$ in $\mathcal{G}$ is t-close to the $\mathcal{SA}$ distribution $\mathcal{P}$ in $\mathcal{DB}$.*

Proof. We transform $\mathcal{Q}$ to $\mathcal{P}$ in two steps. First, we transform $\mathcal{Q}$ to $\mathcal{Q}'$, a distribution that follows the proportionality requirement with respect to φ; then, we transform $\mathcal{Q}'$ to $\mathcal{P}$. By Lemma 3.1, the cost of transforming $\mathcal{Q}$ to $\mathcal{Q}'$ is upper-bounded by $\mathcal{D}$. Furthermore, by Theorem 3.3, the cost of transforming $\mathcal{Q}'$ to $\mathcal{P}$ is upper bounded by $\mathcal{U}$. Therefore, the EMD of transforming $\mathcal{Q}$ to $\mathcal{P}$ via $\mathcal{Q}'$ is upper bounded by $\mathcal{D} + \mathcal{U}$. Thus, if $\mathcal{D} + \mathcal{U} \leq t$, the EMD between $\mathcal{Q}$ and $\mathcal{P}$ is at most t. □

We now consider the process of dynamically determining the size of an EC, or deciding how many tuples to take out from each bucket to form an EC. First, we consider all tuples of $\mathcal{DB}$ (i.e., all the buckets in φ) as a single EC, r. Then we split r into two ECs by dichotomizing $\mathcal{B}_i$ into $\mathcal{B}^1_i$ and $\mathcal{B}^2_i$, where $i = 1, 2 \ldots, |\varphi|$. $\mathcal{B}^1_i$ and $\mathcal{B}^2_i$ have *approximately* the same size. The left child c_1 of r is composed of $\mathcal{B}^1_i$, and the right child c_2 of r is composed of $\mathcal{B}^2_i$, where $i = 1, 2, \ldots, |\varphi|$. Let $d(c_1, \varphi)$ and $d(c_2, \varphi)$ be the φ distributions in c_1 and c_2 respectively. By Theorem 3.4, the split is allowed only if both $EMD(d(c_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} \leq t$ and $EMD(d(c_2, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} \leq t$ hold. After r is split, we recursively split c_1 and c_2 in the same way. We illustrate this process with an example.

Example 3.7. *Re-consider Example 3.6, with $t = 0.45$, $\varphi = \{\mathcal{B}_1, \mathcal{B}_2\}$, and $d^u_{12} = 1$. If we strictly follow the proportionality requirement, then there are at most two ECs, each having 5 tuples from $\mathcal{B}_1$ and 4 tuples from $\mathcal{B}_2$. However, we can generate more ECs by dynamically determining their size as follows. By*

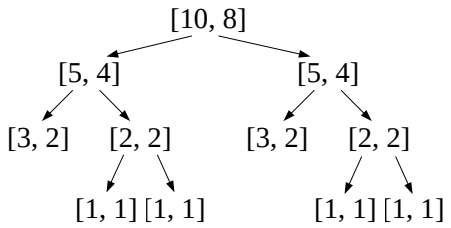


Figure 3.6: Example of dynamically determining EC size

Theorem 3.3, we have $\mathcal{U} = CET^U_{\mathcal{B}_1} + CET^U_{\mathcal{B}_2} = \frac{2}{9} + \frac{1}{6} = \frac{7}{18}$ *(see Example 3.3). The notation* $r = [10, 8]$ *in Figure 3.6 means that* r *contains* 10 *tuples from* $\mathcal{B}_1$ *and* 8 *tuples from* $\mathcal{B}_2$ *(i.e., all tuples in* $\mathcal{DB}$*). We dichotomize* r *into* $c_1 = [5, 4]$ *and* $c_2 = [5, 4]$*. Since both* c_1 *and* c_2 *follow the proportionality requirement with respect to* φ*, and* $\mathcal{U} = \frac{7}{18} < t$*, the split is allowed. We proceed to split* c_1 *into* $g_1 = [3, 2]$ *and* $g_2 = [2, 2]$*. Now* $d(\mathcal{DB}, \varphi) = (\frac{5}{9}, \frac{4}{9})$*,* $d(g_1, \varphi) = (\frac{3}{5}, \frac{2}{5})$*, and* $d(g_2, \varphi) = (\frac{1}{2}, \frac{1}{2})$*. Since* $EMD(d(g_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{2}{45} + \frac{7}{18} < 0.45$ *and* $EMD(d(g_2, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{1}{18} + \frac{7}{18} < 0.45$*, splitting* c_1 *is allowed by Theorem 3.4. If we further dichotomize* g_1 *into* $gg_1 = [2, 1]$ *and* $gg_2 = [1, 1]$*, then* $EMD(d(gg_1, \varphi), d(\mathcal{DB}, \varphi)) + \mathcal{U} = \frac{1}{9} + \frac{7}{18} > 0.45$*. Thus, splitting* g_1 *is not allowed. Still, further splitting* g_2 *into two ECs each having one tuple from* $\mathcal{B}_1$ *and one tuple from* $\mathcal{B}_2$ *is allowed. The process of splitting* c_2 *is similar to* c_1*. The recursive splitting process generates the tree shown in Figure 3.6. Each leaf node represents the size of a possible EC.*

The redistribution phase of SABRE uses the binary tree as illustrated in Figure 3.6 to effectively split ECs. The same tree structure has also been employed in Mondrian [49]. However, the trees for the two methods are generated in a very different manner due to the distinct requirements of their underlying privacy models. The Mondrian k-anonymi-zation algorithm, under local recoding, splits a node in the binary tree once it accommodates at least $2k$ tuples. Instead,

the redistribution phase of SABRE, tailored for t-closeness, allows such a splitting, only if the resultant ECs strictly satisfy the conditions specified in Theorem 3.4. Still, the redistribution phase only determines the size of each possible EC (i.e., the number of tuples from each bucket to compose it). How real tuples are retrieved from each bucket to create an EC is discussed in Section 3.4.3.

Function ECSize (φ)

1 Let $\varphi = \{\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_{|\varphi|}\}$ be the set of buckets generated from the bucketization phase;
2 Let a be an array, and a_i be the i^{th} element of a;
3 Create a tree ECT and its root r;
4 Initialize $r_{\cdot a}$ by $r_{\cdot a_i} = |\mathcal{B}_i|$, $i = 1, 2, \ldots, |\varphi|$;
5 dichotomize (r, φ);
6 Let S_a be a set of arrays initialized to be empty;
7 Traverse ECT, and for each leaf node n add $n_{\cdot a}$ to S_a;
8 Return S_a;

Function ECSize describes our algorithm that determines the number of tuples to be taken out from $\mathcal{B}_i$, $i = 1, 2, \ldots, |\varphi|$. Parameter φ is the set of buckets generated from the bucketization phase (step 1, see Section 3.4.1). Each array a represents an EC, and its i^{th} element a_i is the number of tuples from $\mathcal{B}_i$ when generating the EC (step 2). ECT is a tree with each node associated with an array a. The root of ECT is r, and $r_{\cdot a}$ represents an EC composed of the whole table (steps 3-4). We call function dichotomize to generate ECT (step 5). In the final ECT, each leaf node represents an EC that cannot be further split. Eventually, each leaf node n in ECT is scanned, and its $n_{\cdot a}$ is stored in S_a (steps 7-8).

Procedure dichotomize splits ECT recursively. Parameter n is a node in ECT, and φ is the computed bucket partition (see Section 3.4.1) of table $\mathcal{DB}$. Steps 3-5 split the EC denoted by node n into two, whose sizes are determined by a_i^1 and a_i^2 respectively, $i = 1, 2, \ldots, |\varphi|$. If both c_1 and c_2 follow Theorem

Procedure dichotomize (n, φ)

1 **if** $n_{.a_i} < 2$, *for all* $i \in \{1, \ldots, |\varphi|\}$ **then**
2 Return;
3 **foreach** $i \in \{1, \ldots, |\varphi|\}$ **do**
4 $a_i^1 = round(0.5 \times n_{.a_i})$;
5 $a_i^2 = n_{.a_i} - a_i^1$;
6 Set $c_{1.a}$ by $c_{1.a_i} = a_i^1, i = 1, \ldots, |\varphi|$;
7 $\mathcal{D}_1 = EMD(d(c_1, \varphi), d(\mathcal{DB}, \varphi))$;
8 Set $c_{2.a}$ by $c_{2.a_i} = a_i^2, i = 1, \ldots, |\varphi|$;
9 $\mathcal{D}_2 = EMD(d(c_2, \varphi), d(\mathcal{DB}, \varphi))$;
10 **if** $\mathcal{D}_1 + U \leq t$ *and* $\mathcal{D}_2 + U \leq t$ **then**
11 Set c_1 and c_2 to be the left and right child of n respectively;
12 dichotomize (c_1, φ);
13 dichotomize (c_2, φ);

3.4 (steps 6-10), then n will acquire c_1 and c_2 as its children (step 11). Steps 12-13 recursively examines whether the two newly generated children can be split. The process terminates if splitting a node cannot generate two smaller ECs (the evaluation of step 1 is true) or the evaluation of step 10 is false. When none of the nodes in ECT can be split, ECT is fully generated.

Now our presentation of the two phases is complete. In the following, we put the bucketization phase (Section 3.4.1) and the redistribution phase (section 3.4.2) together and summarize the framework of SABRE.

3.4.3 SABRE and its two instantiations

Algorithm SABRE provides a high level description of our framework. $\{v_1, v_2, \ldots, v_m\}$ is the domain of the sensitive attribute. The global $\mathcal{SA}$ distribution in the whole table $\mathcal{DB}$ is $(p_1, p_2, \ldots, p_m)$, where $p_i = \frac{\mathcal{N}_i}{|\mathcal{DB}|}$ and $\mathcal{N}_i$ is the number of tuples in $\mathcal{DB}$ with $\mathcal{SA}$ value of v_i (step 1). Steps 2-8 deal with the *bucketization phase*. Function bucketCat computes the bucketization tree for categorical sensitive attribute (step 5). If the $\mathcal{SA}$ is numerical, the tree is generated by function bucketNum (step 8). In the final bucketization tree, each leaf node is associated to a bucket, and all the tuples with the $\mathcal{SA}$ values included in the leaf node

will be pushed to the corresponding bucket. Thus, a bucket partition of $\mathcal{DB}$, φ, is formed. Step 9 is the *redistribution phase*; function ECSize dynamically determines the size of each possible EC. Steps 10-15 form all possible ECs and output them. S_a, returned by ECSize, is a list of arrays. Each array represents an EC. Given an array a (step 10), SABRE takes a_i tuples from bucket $\mathcal{B}_i \in \varphi$, where a_i is the i^{th} element of a and $i = 1, 2, \ldots, |\varphi|$ (step 13). Then SABRE forms an EC $\mathcal{G}$ out of them (step 14). Each generated EC is output (step 15).

Algorithm: SABRE ($\mathcal{DB}$, $\mathcal{SA}$, t)

1 Let $\{v_1, v_2, \ldots, v_m\}$ be all the $\mathcal{SA}$ values in $\mathcal{DB}$, and $\{p_1, p_2, \ldots, p_m\}$ be their distributions;
2 Let $\mathcal{VP}$ be the list of (v_i, p_i), $i = 1, 2, \ldots, m$;
3 **if** $\mathcal{SA}$ *is categorical* **then**
4 Let $\mathcal{H}$ be the domain hierarchy of $\mathcal{SA}$;
5 $\varphi =$ bucketCat $(\mathcal{H}, \mathcal{VP})$;
6 **else**
7 Sort $\mathcal{VP}$ in the ascending order of $\mathcal{SA}$ values;
8 $\varphi =$ bucketNum $(\mathcal{VP})$;
9 $S_a =$ ECSize (φ);
10 **foreach** *array* a *in* S_a **do**
11 Create an empty EC, say $\mathcal{G}$;
12 **foreach** a_i, i^{th} *element of* a **do**
13 $ec_i =$ takeOut $(\mathcal{B}_i, a_i)$;
14 add ec_i to $\mathcal{G}$;
15 output $\mathcal{G}$;

When taking out tuples from a bucket, SABRE does not distinguish their $\mathcal{SA}$ values. The t-closeness between the EC and the whole table is anyway guaranteed by Theorems 3.3 and 3.4. Which tuples to pick is to be determined by information loss considerations. As discussed, we adopt the General Loss Metric (GLM) (see Section 2.3), as we assume that the anonymized data are for multiple, not known a priori, uses. GLM requires the minimum bound boxes of ECs to be as small as possible. We achieve this by greedily picking tuples of similar QI values, to the extent that is possible.

We provide two instantiations of SABRE: SABRE-KNN and SABRE-AK.

They differ only in function takeOut $(\mathcal{B}_i, a_i)$ (step 13 of Algorithm SABRE), which determines the tuples that should be picked from each bucket. Both schemes utilize the notion of *nearest neighbors*. SABRE-KNN finds the *exact* neighbors, whereas SABRE-AK uses *approximate* neighbors.

We define a multidimensional space with each of the QI attributes serving as a dimension. The axis for a dimension defined by a numerical QI attribute is straightforward. For a categorical QI, the ordering of all leaves by a pre-order traversal of its domain hierarchy forms the axis. Thus, each tuple is represented as a point in this space. We use the Euclidean distance to measure the distance between two points.

SABRE-KNN selects a_i tuples from bucket $\mathcal{B}_i$ based on a kNN search. First, it forms an empty EC $\mathcal{G}$. Then it selects a random tuple x from a randomly selected bucket $\mathcal{B} \in \varphi$. Finally, in each bucket $\mathcal{B}_i$, $i = 1, 2, \ldots, |\varphi|$, it finds the nearest a_i neighbors of x and adds them into $\mathcal{G}$. x and all its selected nearest neighbors are deleted from their original buckets. Thus, to form an EC, all tuples in the buckets need to be scanned once. The time cost of this operation is $O(|S_{\mathcal{G}}| \cdot |\mathcal{DB}|)$, where $|S_{\mathcal{G}}|$ is the number of ECs and $|\mathcal{DB}|$ is the size of the dataset.

As an alternative to the computationally more demanding SABRE-KNN, we also suggest a more efficient scheme, SABRE-AK, that looks for approximate nearest neighbors of x. This is facilitated by the *Hilbert space-filling curve* [59], a continuous fractal that maps regions of the multidimensional QI space to one-dimensional Hilbert values. Each tuple has a Hilbert value corresponding to the region that contains it. If two tuples are close in the multi-dimensional space, their Hilbert values are also close with high probability. SABRE-AK first sorts all tuples in each bucket in ascending order of their Hilbert values. Then, when

looking for x's a_i nearest neighbors in bucket $\mathcal{B}_i$, it selects the a_i tuples that are closest to x in terms of their Hilbert values. In practice, we use binary search to find in $\mathcal{B}_i$ a tuple $\bar{x}$, whose Hilbert value is closest to that of x. Then we check the neighbors of $\bar{x}$ and select the closest a_i ones (including $\bar{x}$) to x. The average time cost of this search is $O\left(|S_\mathcal{G}| \cdot \left(\log \frac{|\mathcal{DB}|}{|\varphi|} + \frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}\right) \cdot |\varphi|\right)$, where $|\varphi|$ is the number of buckets, $\frac{|\mathcal{DB}|}{|\varphi|}$ the average size of a bucket, and $\frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}$ the average number of tuples taken out from a bucket to form an EC. Since $\left(\log \frac{|\mathcal{DB}|}{|\varphi|} + \frac{|\mathcal{DB}|}{|S_\mathcal{G}| \cdot |\varphi|}\right) \cdot |\varphi|) << |\mathcal{DB}|$, we expect SABRE-AK to be more efficient than SABRE-KNN.

3.5 Experimental study

In this section we evaluate the performance of our SABRE-based schemes: SABRE-KNN, SABRE-AK. We compare SABRE against tIncognito [52] and tMondrian [53], i.e., the t-closeness schemes extended from Incognito [48] and Mondrian [49], respectively. The prototype was implemented in Java and the experiments were run on a core-2 duo 2.33GHz CPU machine, with 4GB RAM, running windows XP. We use the CENSUS dataset [3], which contains 500,000 tuples, and has 8 attributes as shown in Table 3.4; the value following the type is the height of the corresponding attribute hierarchy. For instance, attribute *marital status* is categorical and has a hierarchy of height 2. The first 7 attributes are used by default as the QI, and the last one (i.e., *salary*) as the sensitive attribute divided into 50 classes. The least frequent salary class is 49 with frequency 0.2018%, while the most frequent one is 12 with frequency 4.8402%. We generate 5 microdata tables by randomly taking 100,000 to 500,000 tuples from the dataset; the one with 100,000 tuples is the default dataset. t-closeness provides protection against the disclosure of $\mathcal{SA}$ values. Yet it does not handle identity

disclosure. k-anonymity copes with identity disclosure by ensuring that each released tuple is indistinguishable from at least $k-1$ other tuples with respect to their QI values. Still, to create a level playing field, as tIncognito and tMondrian do, SABRE combines *t*-closeness and k-anonymity together. We set the default value of *k* to be 6 (i.e., the size of EC is at least 6). The closeness threshold *t* is a variable, in default it is set to 0.35.

Attribute	**Cardinality**	**Type**
age	79	numerical (4)
sex	2	categorical (1)
education	17	numerical (4)
marital status	6	categorical (2)
race	9	categorical (1)
work class	10	categorical (3)
birth place	83	categorical (2)
salary	50	numerical

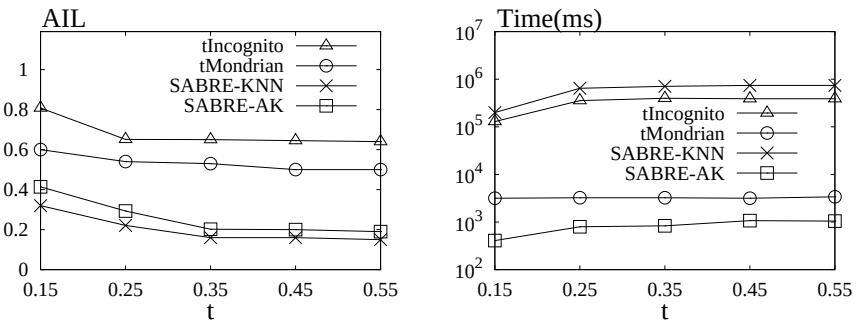
Table 3.4: The CENSUS dataset

We use several metrics to evaluate the quality of the anonymized dataset under the schemes we compare. First, we measure the average information loss (see section 2.3). Then we study the utility of the anonymized dataset using *median relative errors* [82] and *KL-divergence* [40]. We also compare the efficiency of the various schemes based on the elapsed time.

3.5.1 Basic results

We first study the effect of varying the closeness threshold t. Figure 3.7 shows the results. As expected, as t grows and the requirement for similarity between the *salary* distribution in each EC and that in the whole table is relaxed, the information quality for all schemes is improved (Figure 3.7(a)). The two SABRE-based schemes are about equally effective (with SABRE-KNN slightly better

than SABRE-AK) and provide superior information quality (i.e., lower average information loss) compared to both tIncognito and tMondrian. The benefit of a scheme tailored for t-closeness emerges. After all, SABRE selects tuples from buckets in a sophisticated manner, and forms ECs with tuples of as close as possible QI values. Thus, it competes successfully against the schemes of both tIncognito and tMondrian; these schemes were principally designed for the less complex problem of k-anonymization. Moreover, as Figure 3.7(b) shows, SABRE-AK is the most efficient. In all cases, it takes no more than 4 seconds to complete the processing. The time efficiency of tMondrian is comparable to SABRE-AK. They are two orders of magnitude faster than the other two methods.

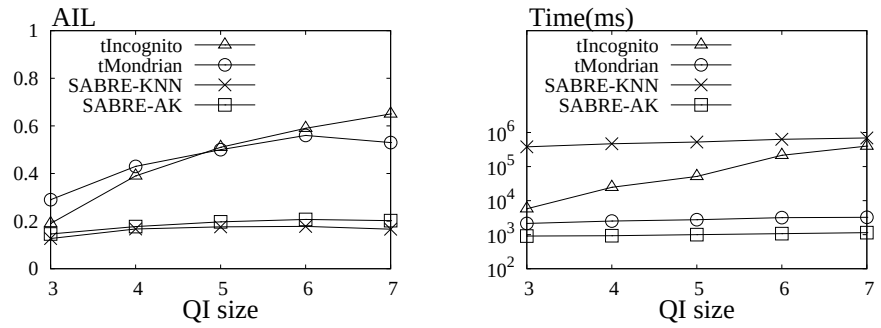


(a) (b)

Figure 3.7: Effect of varying closeness threshold

Next, we investigate the effect of the QI size. We vary the QI size from 3 to 7. When the QI dimensionality increases, data becomes more sparse in the QI space, due to the higher-dimensional degrees of freedom offered; thus, the formed ECs are more likely to have bigger minimum bounding boxes. Thus, we expect information quality to worsen as dimensionality grows for all methods. Still, Figure 3.8(a) shows that the average information loss of both tIncognito and tMondrian grows in a substantially steep manner as QI size grows, while the

SABRE-based schemes degrade only marginally. Thus, the information quality gap between tIncognito/tMondrian and the SABRE-based schemes *widens* as the QI size increases. This result clearly indicates that the SABRE-based methods are more scalable with respect to QI size. Moreover, Figure 3.8(b) shows that SABRE-AK is the fastest, followed by tMondrian, tIncognito, and SABRE-KNN.

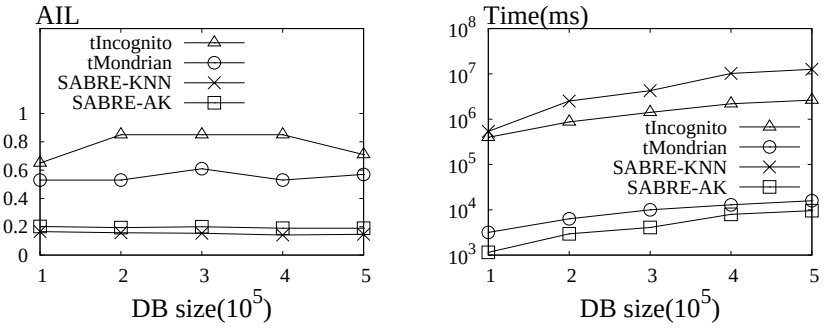


(a) (b)

Figure 3.8: Effect of varying QI size

Our next experiment studies the effect of database size. We vary the size of microdata table from 100,000 to 500,000 tuples. The results are reported in Figure 3.9. As Figure 3.9(a) shows, the data size has no much effect on the information quality of the schemes, except tIncognito. Still, as expected, the elapsed time increases as the table size grows for all schemes; SABRE-AK and tMondrian remain superior in this case.

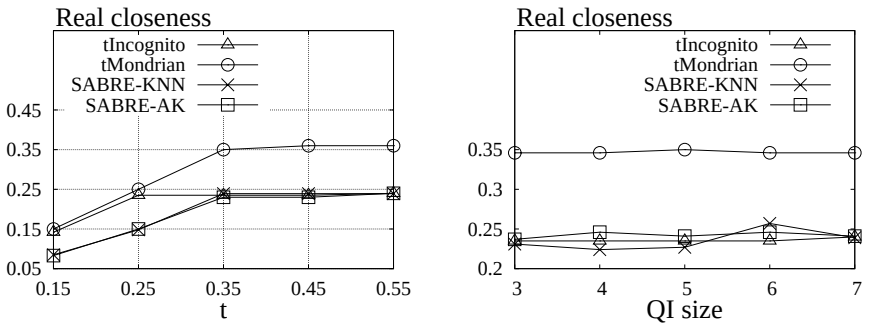
SABRE guarantees that the EMD difference between the distribution of a sensitive attribute in an EC and that in the original input table is at most t. However, the actual closeness of the anonymized data may be smaller than t. We check the real closeness of the anonymized data as follows. For each EC we calculate its closeness value, and we use the maximum one as the real closeness of the whole anonymized dataset. In Figure 3.10(a), we vary t from 0.15 to 0.55. In Figure 3.10(b), we set t to 0.35, and vary the QI size from 3 to 7. We observe



(a) (b)

Figure 3.9: Effect of varying $\mathcal{DB}$ dimensionality (size)

that the real EMD difference of all schemes is smaller than the given threshold t. This result indicates that the anonymized data achieves better privacy than the requirement. Furthermore, the SABRE-based schemes and tIncognito achieve consistently smaller real difference than tMondrian; this result indicates that SABRE and tIncognito offer better privacy than tMondrian.



(a) (b)

Figure 3.10: Real closeness

Figure 3.11 presents the results of the four approaches as we vary the k parameter of the k-anonymity guarantee that all methods also provide. As k increases, identity protection is improved. But a larger k implies that more tuples will be in an EC, thus the minimum bounding box to cover them becomes larger, resulting in higher information loss. The behavior of tIncognito is not uniform, due to its highly heuristic nature.

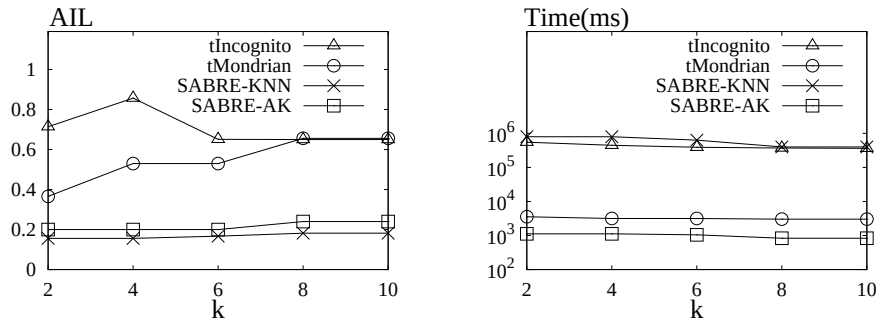


Figure 3.11: Effect of varying *k*

3.5.2 Accuracy of aggregation queries

Apart from information loss, we also study the utility of the anonymized data. In this section, we focus on aggregation queries as they are the basis of statistical analysis and many data mining applications (e.g., association rule mining and decision trees). We first consider the following type of aggregation queries with the *median relative error* as the metric [82]:

```
SELECT COUNT(*) FROM Anonymized-data
WHERE pred(A_1) AND ... AND pred(A_λ) AND pred(SA)
```

Each A_i is a QI attribute. $\mathcal{SA}$ is a sensitive attribute. The query has predicates on the λ randomly selected QI attributes and $\mathcal{SA}$. Let A be one of these $\lambda + 1$ attributes (λ QI attributes + $\mathcal{SA}$). $pred(A)$ has the form of $A \in R$. R is a random interval in the domain of A. R has the length of $|A| \cdot \theta^{\frac{1}{\lambda+1}}$, where $|A|$ is the domain length of A and θ is the *expected selectivity*. Given a query, the precise result $prec$ is computed from the original table, and the estimated result est is obtained from the anonymized table. To calculate est, we assume that tuples in each EC are uniformly distributed, and consider the intersection between the query and the EC. We define $\frac{|est-prec|}{prec} \times 100\%$ as the *relative error*. Our workload consists of 10,000 queries, and we measure the workload error as the *median relative error*. Relative error is undefined when $prec$ is 0. If $prec$ in

a given query is 0, we drop that query.

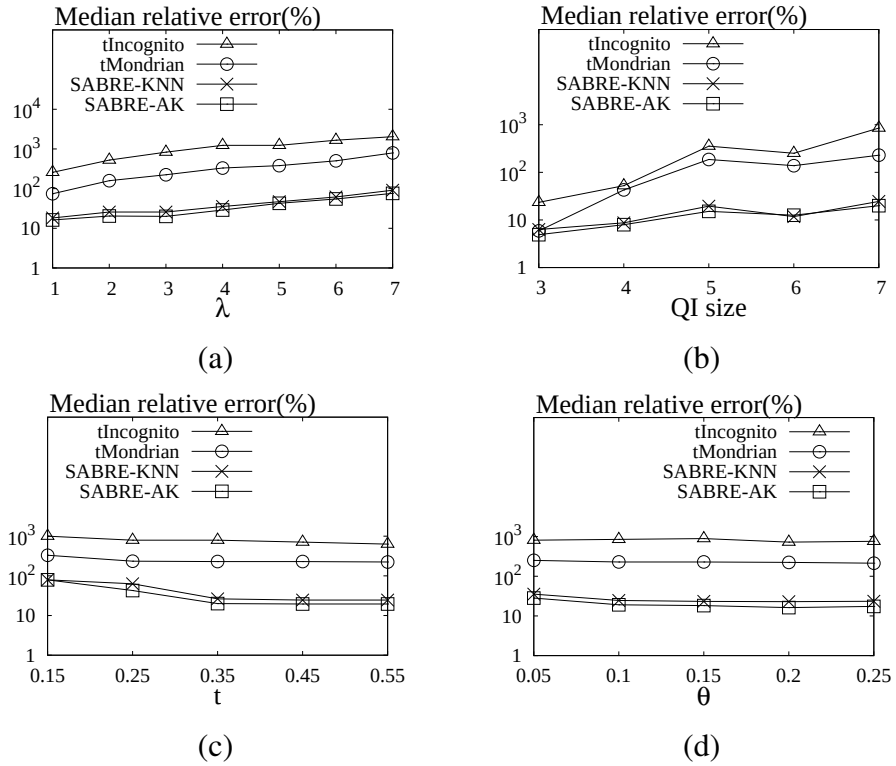


Figure 3.12: Median relative error

We first set θ to 0.1, and vary λ. Each predicate in the `WHERE` clause except the one on $\mathcal{SA}$ has some error. As λ increases, the number of predicates increases, hence the overall error of all the predicates is expected to increase as well. Not surprisingly, the error increases as λ grows, as shown in Figure 3.12(a). In the following experiments we fix λ to 3. Next we set θ to 0.1, and vary the QI size. As the QI size increases, data tend to be more sparse in the QI space, and it is more likely that ECs with bigger minimum bound boxes are created. Consequently, the information loss of the anonymized data grows, and the error we measure also increases. Expectedly, in Figure 3.12(b) the workload error increases with QI size. In Figure 3.12(c) we fix θ to 0.1 and vary t. As t grows, the requirement on the distribution of each EC is relaxed, hence the

information quality of anonymized data rises, and the error we measure drops. Finally, in Figure 3.12(d) we vary the selectivity θ. When θ increases, the range R for each attribute in a predicate increases. This makes the minimum bound box of an EC more likely to be entirely contained in the query region, therefore, the estimate becomes slightly more accurate and the error smaller. In all the above experiments, we find that SABRE-based schemes offer better utility. Remarkably, SABRE-based techniques outperform tIncognito by one order of magnitude in terms of median relative error.

In the following, we evaluate the utility based on typical OLAP queries using the *KL-divergence* metric (as in [40]):

```
SELECT A_1, A_2, ..., A_μ, COUNT(*)
FROM Anonymized-data
WHERE SA = val
GROUP BY A_1, A_2, ..., A_μ
```

All GROUP-BYs for all possible combinations of the QI attributes compose the OLAP datacube lattice. Level μ of the lattice corresponds to all GROUP-BYs over exactly μ attributes. We build two datacube lattices on the CENSUS dataset: α (on the original dataset), and β (on the anonymized dataset). For each cell of β, we consider the intersection between the cell and each EC, assuming a uniform distribution of tuples within the EC. Let α_c and β_c be values of a cell in α and β respectively. We use *KL-divergence* to measure the difference between the cells in α and those in β:

$$KL\text{-}divergence(\alpha, \beta) = \sum_{\forall cell\ c} \alpha_c \times \log \frac{\alpha_c}{\beta_c}$$

In Figure 3.13(a), the level of the lattice is set to 2, and we vary t. As t increases, *KL-divergence* decreases. This is so because, for larger t, the closeness requirement is relaxed, hence the information quality of the anonymized data is

improved. Ideally, the lower the *KL-divergence* is, the better the quality of the anonymized data. When all the cells in α and β are the same, *KL-divergence* is 0. In Figure 3.13(b) we vary the lattice level. When the level is higher, the granularity of the GROUP-BYs in the aggregation query becomes finer. On the other hand, when the level is lower, an aggregation query is more likely to include the whole range of an anonymized EC. This effect makes the four schemes perform better at lower levels. In Figure 3.13(c) we set the level of lattice to 2, and vary the QI size. SABRE-KNN and SABRE-AK are significantly more scalable than the other two methods with growing QI size. Eventually, in Figure 3.13(d) we set the level to 2, and vary the size of the dataset, to get a slightly improving trend of information quality, due to increasing data density. In all the above comparisons, SABRE-based schemes clearly outperform tIncognito and tMondrian.

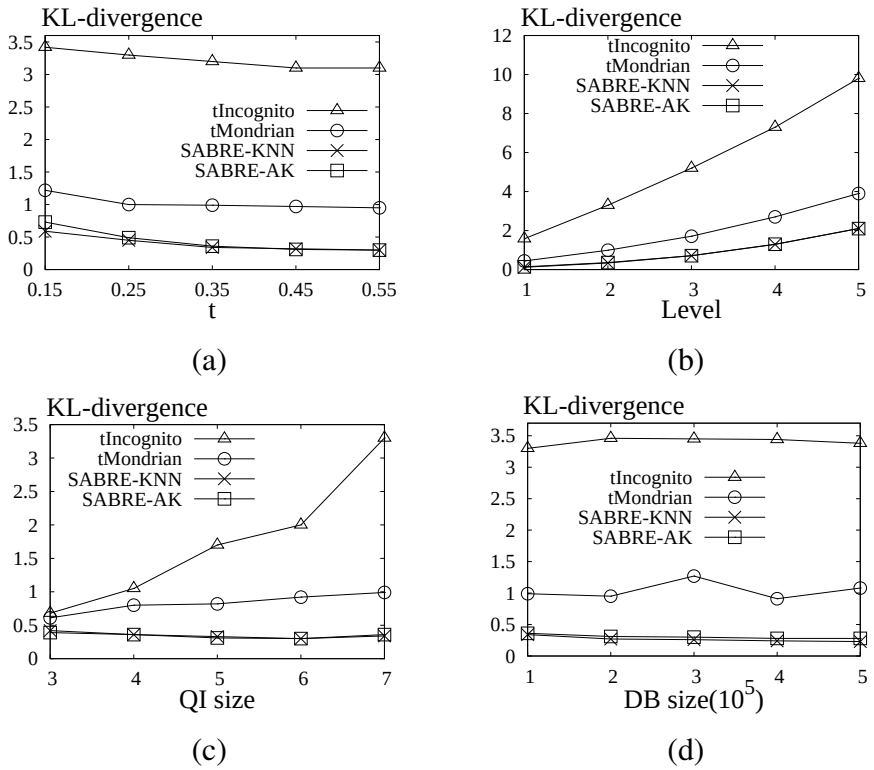


Figure 3.13: KL-divergence with OLAP queries

3.6 Discussion

In this section we make a discussion on SABRE. First, we examine the effect of the $\mathcal{SA}$ hierarchy on information quality. For a categorical $\mathcal{SA}$, the set of children of a split node in the bucketization tree depends on the pre-defined domain hierarchy of $\mathcal{SA}$. With Figure 3.1 as the hierarchy, the root of the bucketization tree is *respiratory and digestive diseases*. If it is split, its children will be *respiratory diseases* and *digestive diseases*, which are pre-defined in Figure 3.1. Thus, the bucketization process is affected by the $\mathcal{SA}$ hierarchy. Besides, the structure of a hierarchy is shaped by the number of its leaves and the fanouts of its root and internal nodes (the height is automatically decided thereby). We assume that the hierarchy is well defined by its domain expert.

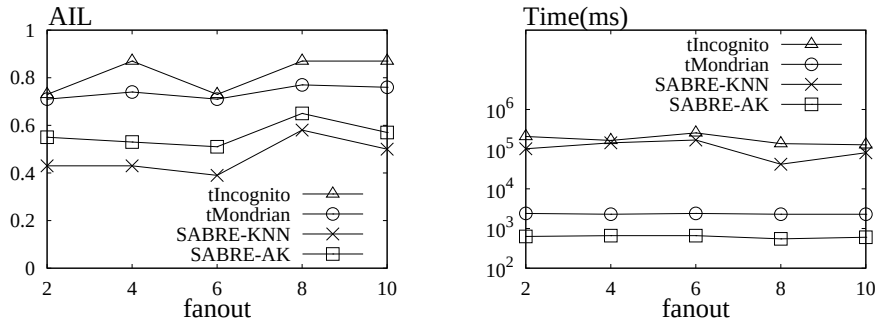


Figure 3.14: Effect of varying fanout

In Table 3.4, `salary` is a numerical sensitive attribute with 50 distinct values. Next, we consider them as 50 leaves, and build hierarchies over them to see the effects on anonymization process. To simplify the problem, we assume that the fanouts in a simulated hierarchy are uniform, while its leaves are sorted from left to right by the ascending order of their values. Figure 3.14 presents our experimental results. We observe that the information quality of the anonymized data does not change uniformly as a function of fanout. Still, overall the curves

suggest that a smaller value of fanout tends to preserve more information. After all, when the fanout is smaller, the hierarchy is deeper. Thus, the intermediate levels between the root and leaves are increased, hence the bucketization gains more flexibility in partitioning $\mathcal{SA}$ values to buckets at several levels. In effect, the redistribution phase can later redistribute tuples to ECs more effectively.

Apart from the preceding discussion, we also examine the potential for an extension of SABRE to settings with more than one sensitive attributes. Without loss of generality, assume that $\mathcal{DB}$ has two sensitive attributes $SA_1 = \{u_1, u_2, \ldots, u_n\}$ and $SA_2 = \{v_1, v_2, \ldots, v_m\}$. For the sake of simplicity we consider the case that SA_1 is independent of SA_2. SABRE is extended to attain t-closeness with respect to both SA_1 and SA_2 as follows. It first transforms $\mathcal{DB}$ to $\mathcal{DB}_1$, which satisfies t-closeness with respect to SA_1. Then it checks each EC in $\mathcal{DB}_1$ to determine whether it also observes t-closeness with respect to SA_2. If it does not, then it is merged with its nearest neighbor ECs, until t-closeness is attained. This effect is always achievable because an EC formed from the union of two ECs will not have an increased distance from $\mathcal{DB}$ with respect to SA_2 distribution. In particular, let $\mathcal{P}$ be the overall SA_2 distribution in $\mathcal{DB}$, and $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be the SA_2 distributions of two ECs in $\mathcal{DB}_1$, respectively. Let $\mathcal{Q}$ be the SA_2 distribution of the EC formed from the union of the two ECs; then, $EMD(\mathcal{P}, \mathcal{Q}) \leq \max\{EMD(\mathcal{P}, \mathcal{Q}_1), EMD(\mathcal{P}, \mathcal{Q}_2)\}$ [52]. In effect, after all the required merges of ECs in $\mathcal{DB}_1$, we can transform $\mathcal{DB}_1$ to $\mathcal{DB}_2$, which attains t-closeness with respect to both SA_1 and SA_2.

Lastly, we discuss the applicability of other distance metrics in SABRE. Let d be a distance measure, and $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ be any three sensitive attribute distributions. To be applicable in SABRE, d needs to satisfy the following properties: I. *non-negativity*: $d(\mathcal{P}_1, \mathcal{P}_2) \geq 0$; II. *identity of indiscernibles*:

$d(\mathcal{P}_1, \mathcal{P}_2) = 0$ if and only if $\mathcal{P}_1 = \mathcal{P}_2$; and III. *triangle inequality*: $d(\mathcal{P}_1, \mathcal{P}_3) \leq d(\mathcal{P}_1, \mathcal{P}_2) + d(\mathcal{P}_2, \mathcal{P}_3)$. If d also has the property of *symmetry*, then d is a metric. Therefore, besides EMD, other measures such as Euclidean metric and Hamming distance can also be applied to SABRE. However, neither the KL-divergence nor the Jensen-Shannon divergence has the triangle inequality property, hence they are inapplicable to SABRE. Still, the square root of the Jensen-Shannon divergence is a metric and therefore a possible candidate distance measure for SABRE. Nevertheless, when a new distance measure is applied to SABRE, the upper bound of the cost related with a bucket (Theorem 3.2) needs to be customized for that measure.

As a final note, t-closeness based on EMD has the drawback that it defines no clear intelligible relationship between *t* and the privacy it affords. However, EMD is still a meaningful distance measure for *t*-closeness, due to its following attractive properties [53]: 1. Awareness of semantic closeness; 2. Simplicity for understanding; 3. Subset property (i.e., if table $\mathcal{DB}$ satisfies t-closeness in QI, then it also satisfies t-closeness in any subset of QI.)

3.7 Summary

This chapter proposed SABRE, a novel framework for distribution-aware microdata anonymization based on the t-closeness principle. SABRE guarantees t-closeness in an elegant and efficient manner, without depending on techniques developed for other privacy models. We have shown the applicability of our scheme on both categorical and numerical attributes. Our extensive experimental study demonstrated that our two SABRE instantiations, SABRE-AK and SABRE-KNN, clearly outperform previous schemes in terms of information quality, while SABRE-AK also outperforms them in terms of efficiency. In

conclusion, SABRE provides the best known resolution of the tradeoff between privacy, information quality, and computational efficiency with a t-closeness guarantee in mind.

CHAPTER 4

β-LIKENESS: ROBUST MICRODATA ANONYMIZATION

The most recently proposed privacy model, t-closeness, raises the privacy guarantees above those of its predecessors, i.e., k-anonymity and ℓ-diversity. However, all the instantiations of t-closeness interpret the t threshold as a bound on the *cumulative* distance between two frequency distributions, failing to translate t to a comprehensible privacy guarantee. In this chapter we propose β-likeness, a new privacy principle, which requires that each equivalence class (EC) should satisfy a threshold β on the *positive relative* difference between each $\mathcal{SA}$ value's frequency in the EC and that in the whole table. Therefore, the β value corresponds to an intelligible privacy guarantee that limits an attacker's information gain.

4.1 Introduction

The t-closeness model requires that the frequency distribution of $\mathcal{SA}$ values in an EC *differs* from their overall distribution by at most a given threshold t; in effect, this threshold aims to constrain the information an adversary gains after seeing a single EC with respect to the information provided by the full released table.

Just like ℓ-diversity is open to many ways of measuring the number of "well-represented" values in an EC [57], t-closeness is open to diverse frequency distribution distance functions. To date, functions such as the Earth Mover's Distance (EMD) [52], Kullback-Leibler divergence [63], and Jensen-Shannon divergence [53, 54] have been adopted. Still, all these functions interpret the t threshold as a bound on the *cumulative* difference between two frequency distributions. Indeed, this interpretation emanates out of the t-closeness model itself [52, 53], not out of a particular manifestation. However, a privacy model should provide grounds for effective and human-understandable policy [60]. Yet a model that bounds a *cumulative* function of frequency differences between distributions fails to provide a meaningful privacy guarantee for the persons whose privacy is at stake.

To elaborate on this point, we first discuss EMD. Assume a data set $\mathcal{DB}$ with $\mathcal{SA}$ values HIV and Flu, having semantic distance 1 (for the purpose of measuring EMD between them). If the overall $\mathcal{SA}$ distribution between them is $\mathcal{P} = (0.4, 0.6)$, and their distribution in an EC is $\mathcal{Q} = (0.5, 0.5)$, then the EMD between $\mathcal{P}$ and $\mathcal{Q}$ is $EMD(\mathcal{P}, \mathcal{Q}) = 0.1$. Still, if their overall distribution is $\mathcal{P}' = (0.01, 0.99)$ and their distribution in an EC is $\mathcal{Q}' = (0.11, 0.89)$, then $EMD(\mathcal{P}', \mathcal{Q}') = 0.1$ again. Both cases satisfy 0.1-closeness. However, the information gain in the latter case is much larger than that in the former: the probability of HIV rises by only 25% from 0.4 to 0.5, but it rises by 1000% from 0.01 to 0.11. In effect, the two cases do *not* afford the same privacy in human terms. This example appears in [52], where it is noted that EMD does not provide a clear privacy guarantee. In fact, not only EMD, but any distance function that *aggregates absolute differences* would face a similar problem, since

such functions do not provide informative *maximum relative difference* guarantees [44] about individual $\mathcal{SA}$ values. In the current example, a small relative difference of Flu-frequency between $\mathcal{P}'$ and $\mathcal{Q}'$ evens up a large relative difference of HIV-frequency between them.

The recently proposed alternatives to EMD as frequency distribution distance functions, namely Kullback-Leibler divergence [63] and Jensen-Shannon divergence [53, 54], also fail to pay appropriate attention to less frequent $\mathcal{SA}$ values and their relative differences. In our running example, assume a dataset where the overall distribution of HIV and Flu is $\widetilde{\mathcal{P}} = (0.01, 0.99)$, and their distribution in an EC is $\widetilde{\mathcal{Q}} = (0.03, 0.97)$. Then the K-L divergence (J-S divergence) between $\mathcal{P}$ and $\mathcal{Q}$ (see above), is 0.0290 (0.0073), while that between $\widetilde{\mathcal{P}}$ and $\widetilde{\mathcal{Q}}$ is 0.0133 (0.0038). In effect, both these alternatives estimate the privacy afforded by $\widetilde{\mathcal{Q}}$ with respect to $\widetilde{\mathcal{P}}$ as *higher* than that afforded by $\mathcal{Q}$ with respect to $\mathcal{P}$. However, the confidence for HIV increases only by 25% in the latter case, while it rises by 200% in the former. Furthermore, the observed problem is aggravated when small frequency differences for a multitude of $\mathcal{SA}$ values balance out a large difference (i.e., privacy breach) for a single critical value.

From the above discussion, we derive a property that a more appropriate privacy model should have: it should require that each EC satisfy a bound on the *relative* frequency difference of each *single* $\mathcal{SA}$ value therein. Besides, in most real-world applications, it is information gain in the form of an *increase* in the adversary's confidence for a certain $\mathcal{SA}$ value (e.g., HIV-positive) that poses a privacy threat. A *decrease* in the adversary's confidence does not[1] constitute

[1] In case of a binary sensitive value (e.g., homosexual, heterosexual) a decrease in one implies an increase in the other.

a threat. We deduce that it suffices to pose a bound on *positive* frequency differences only. Besides, a novel privacy model requires an anonymization algorithm customized for the particular privacy-utility tradeoff posed by that model itself [27, 41].

Past work has attempted to propose a privacy model that has the first property outlined above: δ-disclosure [21]. However, this attempt fails in three fundamental respects: (1) it postulates that every $\mathcal{SA}$ value in the overall table should also occur in each EC; (2) it does not distinguish between an increase and a decrease in the adversary's confidence on an $\mathcal{SA}$ value; (3) it focuses on a *negative* result, namely the destruction of data-mining utility by existing algorithms adopted to δ-disclosure (and other models). Our work differs from [21] in all these three respects.

In this chapter, we propose β-likeness: a robust model for microdata anonymization that follows on the progression from k-anonymity to ℓ-diversity to t-closeness, while eschewing the drawbacks of δ-disclosure; this model imposes a threshold on the *positive relative* difference of *each* $\mathcal{SA}$ value frequency between an EC and the overall table. Thereby, it provides a human-understandable privacy guarantee that limits an adversary's information gain; such a comprehensible guarantee is not provided by aggregate measures. Furthermore, we accompany our privacy model with an anonymization algorithm tailored for its own particular requirements. Our scheme first partitions tuples into buckets by their $\mathcal{SA}$ value frequencies, and then redistributes those tuples to ECs. Our experimental study demonstrates that our scheme: (i) provides effective privacy guarantees in a way that state-of-the-art t-closeness schemes cannot, even when set to achieve the same information accuracy or privacy measured by the criterion of t-closeness; (ii) is more effective and efficient in its task than both the

δ-disclosure scheme used in [21] (which is an adaptation of Mondrian [49, 50]), and a similar benchmark scheme that we devise by adapting Mondrian to the new model.

The remaining of this chapter is organized as follows. In the next section, we introduce the modeling of β-likeness. Section 4.3 presents BUREL, an algorithm customized for β-likeness. We carry out a comprehensive performance evaluation in Section 4.4, and conclude our work in Section 4.5.

4.2 The privacy model

This section introduces our new privacy model, β-likeness. Table 4.1 gathers together the notations we use throughout this chapter.

$\mathcal{DB}$	Original microdata table
$\mathcal{SA}$	Sensitive attribute in $\mathcal{DB}$
$\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$	The domain of $\mathcal{SA}$
N_i	Number of tuples with v_i in $\mathcal{DB}$
$p_i = N_i/\lvert\mathcal{DB}\rvert$	Frequency of v_i in $\mathcal{DB}$
$\mathcal{P} = (p_1, p_2, \ldots, p_m)$	Overall $\mathcal{SA}$ distribution in $\mathcal{DB}$
$\mathcal{G}$	Equivalence class
$\mathcal{Q} = (q_1, q_2, \ldots, q_m)$	$\mathcal{SA}$ distribution in $\mathcal{G}$

Table 4.1: Notations

4.2.1 β-likeness

In our model, we do not perturb the $\mathcal{SA}$ value of any tuple in $\mathcal{DB}$. Therefore, once $\mathcal{DB}$ is anonymized and published, $\mathcal{P}$, the $\mathcal{SA}$ distribution in $\mathcal{DB}$, will be public knowledge. Thus, given any EC $\mathcal{G}$ with $\mathcal{SA}$ distribution $\mathcal{Q}$, we need to constrain the information gained from $\mathcal{Q}$ with respect to $\mathcal{P}$.

Definition 4.1 (information gain)**.** *Assume a table $\mathcal{DB}$ with a sensitive attribute $\mathcal{SA}$. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the $\mathcal{SA}$ domain, and $\mathcal{P} = (p_1, p_2, \ldots, p_m)$ be the overall $\mathcal{SA}$ distribution in $\mathcal{DB}$. Suppose that $\mathcal{Q} = (q_1, q_2, \ldots, q_m)$ is the $\mathcal{SA}$ distribution in an equivalence class $\mathcal{G}$, formed by tuples from $\mathcal{DB}$. The* information gain *on any $\mathcal{SA}$ value $v_i \in \mathcal{V}$ is $\mathsf{D}(p_i, q_i)$, where D is a distance function between p_i and q_i.*

We say that the information gain on v_i is *positive*, when $p_i < q_i$, and *negative*, when $p_i \geq q_i$. Negative information gain lowers the correlation between a personal record and v_i in EC $\mathcal{G}$ below that in the whole table. In most cases, such gain enhances privacy. However, there may exist $\mathcal{SA}$ values such as *heterosexual*, for which a reduced likelihood may inadvertently violate privacy. Nevertheless, we assume that the $\mathcal{SA}$ domain always includes the negation of such values. Thus, negative information gain on *heterosexual* always appears as positive gain for *homosexual*. Therefore, we can directly control the positive gain on the value (such as *homosexual*) that poses the privacy threat. Under this reasonable assumption, we are concerned with *positive* information gain only. Then we define basic β-likeness as follows.

Definition 4.2 (basic β-likeness)**.** *Given table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, let $\mathcal{V} = \{v_1, \ldots, v_m\}$ be the $\mathcal{SA}$ domain, and $\mathcal{P} = (p_1, \ldots, p_m)$ the overall $\mathcal{SA}$ distribution in $\mathcal{DB}$. An EC $\mathcal{G}$ with $\mathcal{SA}$ distribution $\mathcal{Q} = (q_1, \ldots, q_m)$ is said to satisfy basic β-likeness, if and only if $\max\{\mathsf{D}(p_i, q_i) | p_i \in \mathcal{P}, p_i < q_i\} \leq \beta$, where $\beta > 0$ is a threshold.*

For a table $\mathcal{DB}'$ anonymized from table $\mathcal{DB}$ to obey β-likeness, all equivalence classes $\mathcal{G} \subset \mathcal{DB}'$ have to conform to β-likeness. Contrary to previous

models [21, 52, 54, 63], basic β-likeness clearly quantifies the relationship between the β threshold and positive information gain. Thanks to the maximum-distance threshold it imposes, it inherently safeguards against *skewness attacks* and *semantic attacks* [52]. Lastly, as it clearly distinguishes between positive and negative information gain (and accepts $\mathcal{SA}$ values absent from an EC), it allows for more flexibility in anonymization, hence higher information quality than models like δ-disclosure [21].

Apart from specifying a *maximum*, instead of a cumulative, distance threshold, we should also define the distance function D in an appropriate manner. As we have argued, a measure of *absolute* difference does not serve our purposes, since it fails to protect less frequent $\mathcal{SA}$ values. We opt for *relative* difference instead, and define the distance function as $\mathsf{D}(p_i, q_i) = \frac{q_i - p_i}{p_i}$. This function obeys the *monotonicity property*.

Lemma 4.1 (Monotonicity Property)**.** *Assume that an $\mathcal{SA}$ value $v_i \in \mathcal{V}$ has frequency p_i in the overall table $\mathcal{DB}$, q_i^1 (q_i^2) in EC $\mathcal{G}_1$ ($\mathcal{G}_2$), generated from tuples in $\mathcal{DB}$, and q_i^3 in $\mathcal{G}_1 \cup \mathcal{G}_2$. Then $\mathsf{D}(p_i, q_i^3) \leq \max\{\mathsf{D}(p_i, q_i^1), \mathsf{D}(p_i, q_i^2)\}$.*

Proof. Assume there are n_1 (n_2) tuples with v_i in $\mathcal{G}_1$ ($\mathcal{G}_2$). Then $q_i^1 = \frac{n_1}{|\mathcal{G}_1|}$, $q_i^2 = \frac{n_2}{|\mathcal{G}_2|}$, and $q_i^3 = \frac{n_1+n_2}{|\mathcal{G}_1|+|\mathcal{G}_2|} = \frac{q_i^1|\mathcal{G}_1|+q_i^2|\mathcal{G}_2|}{|\mathcal{G}_1|+|\mathcal{G}_2|} \leq \max\{q_i^1, q_i^2\}$. Thus, $\mathsf{D}(p_i, q_i^3) \leq \max\{\mathsf{D}(p_i, q_i^1), \mathsf{D}(p_i, q_i^2)\}$. □

The monotonicity property ensures that a union of two ECs yields no larger distance between p_i and q_i than its united parts. Hence, ECs violating β-likeness can be transformed to follow β-likeness by merge operations. The relative distance function instantiates basic β-likeness by the constraint $\mathsf{D}(p_i, q_i) = \frac{q_i - p_i}{p_i} \leq \beta$, where p_i and q_i are the distributions of any $\mathcal{SA}$ value $v_i \in \mathcal{V}$ in the whole table and an EC, respectively. This constraint amounts to an upper bound for the frequency of v_i in any EC, q_i, namely $q_i \leq (1 + \beta) \cdot p_i$. Our *relative* distance

function pays due attention to less frequent $\mathcal{SA}$ values. However, this function provides a meaningful frequency bound only if $(1+\beta)\cdot p_i < 1$; it then caters for $\mathcal{SA}$ values whose frequency in $\mathcal{DB}$ is $p_i < \frac{1}{1+\beta}$. In our effort to pay due attention to such less frequent values, we have discriminated against $\mathcal{SA}$ values of frequency larger than $\frac{1}{1+\beta}$. Such values can assume frequency 1 in an EC. Thus, an adversary identifying that a person's record is within such an EC can infer the $\mathcal{SA}$ value of that person with 100% confidence. The disclosure of such frequent $\mathcal{SA}$ values may pose a privacy threat. To address this limitation, we provide an *enhanced* definition of β-likeness.

Definition 4.3 (enhanced β-likeness)**.** *For table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, let $\mathcal{V} = \{v_1, \ldots, v_m\}$ be the $\mathcal{SA}$ domain, and $\mathcal{P} = (p_1, \ldots, p_m)$ the overall $\mathcal{SA}$ distribution in $\mathcal{DB}$. An EC $\mathcal{G}$ with $\mathcal{SA}$ distribution $\mathcal{Q} = (q_1, \ldots, q_m)$ is said to satisfy enhanced β-likeness, if and only if $\forall q_i$, $\mathsf{D}(p_i, q_i) = \frac{q_i - p_i}{p_i} \leq \min\{\beta, -\ln(p_i)\}$, where $\beta > 0$ is a threshold and $\ln(p_i)$ is the natural logarithm of p_i.*

The inequality constraint in the above definition implies $q_i \leq p_i \cdot (1 + \min\{\beta, -\ln(p_i)\})$. We can then define the *upper bound* that enhanced β-likeness imposes on the frequency of v_i in an EC by function $f(p_i) = p_i \cdot (1 + \min\{\beta, -\ln(p_i)\})$, which can be decomposed as follows.

$$f(p_i) = \begin{cases} p_i\,(1+\beta)\,, & 0 < p_i \leq e^{-\beta} \\ p_i\,(1-\ln(p_i))\,, & e^{-\beta} \leq p_i \leq 1 \end{cases} \tag{4.1}$$

The first segment of $f(p_i)$ is a linear, monotonically increasing function of p_i. The second segment is a concave, also monotonically increasing function of p_i, with derivative $-\ln(p_i)$. The two segments meet at $p_i = e^{-\beta}$. In effect, $f(p_i)$ is a continuous, monotonically increasing function of p_i in $(0, 1]$ with $f(0) = 0$ and $f(1) = 1$. Intuitively, the second segment *bends* the function's slope so

as not to exceed the maximum value of 1. The monotonicity of $f(p_i)$ implies that an EC $\mathcal{G}$ following the enhanced β-likeness constraint obeys the following properties:

1. The maximum frequency of an $\mathcal{SA}$ value v_i in $\mathcal{G}$ is less than 1, i.e., $f(p_i) < 1$ for any $p_i < 1$.

2. For two $\mathcal{SA}$ values v_i and v_ℓ, such that $p_i < p_\ell$, the maximum allowed frequency of v_i in $\mathcal{G}$ is less than that of v_ℓ, i.e., $f(p_i) < f(p_\ell)$.

3. For an $\mathcal{SA}$ value v_i that is 'infrequent' in table $\mathcal{DB}$, with $p_i \leq e^{-\beta}$, its frequency in $\mathcal{G}$ is at most β times larger than p_i, i.e., $q_i \leq f(p_i) = (1 + \beta) \cdot p_i$.

4. For an $\mathcal{SA}$ value v_i that is 'frequent' in table $\mathcal{DB}$, with $p_i > e^{-\beta}$, its frequency in $\mathcal{G}$ is at most $-\ln(p_i)$ times larger than p_i, i.e., $q_i \leq f(p_i) = (1 - \ln(p_i)) \cdot p_i < (1 + \beta) \cdot p_i$.

These properties guarantee that privacy is protected for all $\mathcal{SA}$ values. Less frequent values receive due attention, with a concrete privacy guarantee, while more frequent values are disallowed from assuming frequency values of 1. The function of the β parameter is twofold: It defines the privacy constraint for less frequent values, as well as the frequency threshold $e^{-\beta}$ above which $\mathcal{SA}$ values are considered frequent enough for their privacy constraint to assume a default form independent of β. We emphasize that our framework for enhanced β-likeness is valid for any monotonic upper-bound function, and independent of the particular choice we make. Our choice of the natural logarithm of p_i is justified as a convenient choice that confers the desirable properties to the $f(p_i)$ function.

Since enhanced β-likeness provides even more robust privacy than basic β-likeness, in the following we focus on it. Unless specified otherwise, β-likeness henceforward always refers to enhanced β-likeness.

4.2.2 Extensions of β-likeness

Definition 4.3 limits the information gain on a single $\mathcal{SA}$ value. Still, it usually makes sense to treat groups of related $\mathcal{SA}$ values in unison. For a categorical $\mathcal{SA}$, an attacker may also be interested in the association between an individual and a group of *semantically related* $\mathcal{SA}$ values. Assuming a domain hierarchy having all values in $\mathcal{V}$ as leaf nodes, the set of leaves under a common ancestor are semantically related. Figure 4.1 shows an example hierarchy for $\mathcal{SA}$ *disease*. Values `SARS`, `pneumonia`, and `bronchitis` are semantically related, as they are all respiratory diseases. Likewise, in case of a *numerical* $\mathcal{SA}$, it may be sufficient for an adversary to detect a *range* within which an $\mathcal{SA}$ value resides [51, 86].

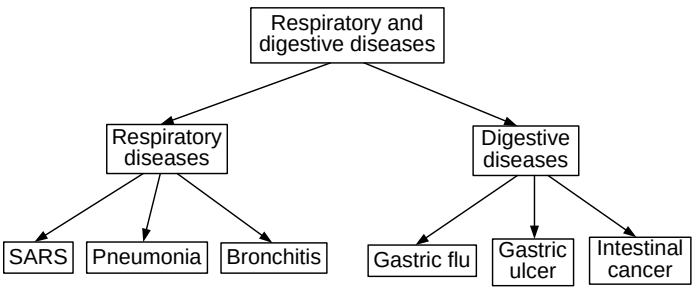


Figure 4.1: Domain hierarchy for diseases

Let $\mathcal{V}' \subseteq \mathcal{V}$ be a set of semantically related $\mathcal{SA}$ values in an EC $\mathcal{G}$. The *prior probability* of associating an individual with $\mathcal{V}'$ is $\rho_1 = \sum_{v_i \in \mathcal{V}'} p_i$; the respective posterior probability is $\rho_2 = \sum_{v_i \in \mathcal{V}'} q_i$. To effectively evaluate the information gain on $\mathcal{V}'$, we need to measure the distance between ρ_1 and ρ_2, as follows.

Definition 4.4 (Semantic β-likeness)**.** *Let $\mathcal{G}$ be an EC generated from tuples in table $\mathcal{DB}$ with a* categorical *$\mathcal{SA}$, $\mathcal{H}$ be the domain hierarchy of $\mathcal{SA}$. $\mathcal{G}$ obeys* semantic β-likeness *iff, for* any *internal node I below a certain level ℓ in $\mathcal{H}$,* $\mathsf{D}(\rho_1, \rho_2) \leq \min\{\beta, -\ln(\rho_1)\}$*, with* $\rho_1 = \sum_{v_i \in \mathsf{leaves}(I)} p_i$*,* $\rho_2 = \sum_{v_i \in \mathsf{leaves}(I)} q_i$*, where* $\mathsf{leaves}(I)$ *is the set of leaves under* I*.*

We develop an analogous definition for numerical $\mathcal{SA}$ as follows.

Definition 4.5 (Range-based β-likeness)**.** *Let $\mathcal{G}$ be an EC generated from tuples in table $\mathcal{DB}$ with a* numerical *$\mathcal{SA}$. We say that $\mathcal{G}$ conforms to* range-based β-likeness *iff, for* any *contiguous interval of $\mathcal{SA}$ values $\mathcal{V}' \subseteq \mathcal{V}$ such that* $\max_{v_i \in \mathcal{V}'}\{v_i\} - \min_{v_i \in \mathcal{V}'}\{v_i\} \leq \varepsilon$*,* $\mathsf{D}(\rho_1, \rho_2) \leq \min\{\beta, -\ln(\rho_1)\}$*, where* $\rho_1 = \sum_{v_i \in \mathcal{V}'} p_i$ *and* $\rho_2 = \sum_{v_i \in \mathcal{V}'} q_i$*.*

In definition 4.5, the range $\mathcal{V}'$ is specified by the absolute difference between the maximum and minimum value therein. Instead, we can also define the subset by relative difference as follows.

$$\frac{\max_{v_i \in \mathcal{V}'}\{v_i\} - \min_{v_i \in \mathcal{V}'}\{v_i\}}{\min_{v_i \in \mathcal{V}'}\{v_i\}} \leq \varepsilon$$

Lemma 4.1 proves the monotonicity of distance function $\mathsf{D}(p_i, q_i)$. Similarly, $\mathsf{D}(\rho_1, \rho_2)$ also has this property.

Lemma 4.2. *Let $(q_1^1, q_2^1, \ldots, q_m^1)$ and $(q_1^2, q_2^2, \ldots, q_m^2)$ be the SA distributions of two ECs $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, and $(q_1^3, q_2^3, \ldots, q_m^3)$ be the SA distribution in $\mathcal{G}_1 \cup \mathcal{G}_2$. Assume that $\mathcal{V}' \subset \mathcal{V}$ is any subset of related $\mathcal{SA}$ values. Then, we have* $\mathsf{D}(\rho_1, \rho_2^3) \leq \max\{\mathsf{D}(\rho_1, \rho_2^1), \mathsf{D}(\rho_1, \rho_2^2)\}$*, where* $\rho_1 = \sum_{v_i \in \mathcal{V}'} p_i$*,* $\rho_2^n = \sum_{v_i \in \mathcal{V}'} q_i^n$*,* $n = 1, 2, 3$*.*

The proof is analogous to that of lemma 4.1. The extensions of β-likeness further enhance the privacy it affords. However, a data set satisfying the regular

definition of enhanced β-likeness (Definition 4.3) does not automatically fulfill its semantic and range-based extensions (Definitions 4.4 and 4.5). The following Lemma provides a formal proof.

Lemma 4.3. *Given an EC $\mathcal{G}$ satisfying enhanced β-likeness, $\mathcal{G}$ does not automatically follow extended β-likeness.*

Proof. We offer a counter example. Given β and two related $\mathcal{SA}$ values, under an extension of β-likeness, v_1 and v_2, assume that their frequencies in $\mathcal{DB}$ and $\mathcal{G}$ are $0 < p_1 = p_2 = p \leq 1$, $q_1 = q_2 = q$, respectively, and that $\beta > -\ln(p)$. Since $\mathcal{G}$ follows enhanced β-likeness, it holds that $\mathsf{D}(p_1, q_1) = \mathsf{D}(p_2, q_2) \leq -\ln(p)$. Thus, $q \leq p - p\ln(p)$. Assume that $q = p - p\ln(p)$. In order for $\mathcal{G}$ to satisfy extended (i.e., semantic or range-based) β-likeness, it should hold that $q_1 + q_2 \leq (p_1 + p_2) - (p_1 + p_2)\ln(p_1 + p_2)$, or $q \leq p - p^{\frac{\ln(2p)}{2}}$. Substituting q, it should hold that $p - p\ln(p) \leq p - p^{\frac{\ln(2p)}{2}} \Leftrightarrow 2\ln(p) \geq \ln(2p) \Leftrightarrow p \geq 2$. However, $p < 1$, hence the last inequality cannot hold. Thus, neither semantic nor range-based β-likeness follows from regular enhanced β-likeness. □

4.3 The algorithm

In this section we design our algorithm customized for β-likeness. The fundamental intuition is based on the following observation. Assume $\mathcal{DB}$ is partitioned into a set of buckets by a 'group-by' on $\mathcal{SA}$. If we form an EC by taking from each bucket a number of tuples *proportional* to the bucket's size, then the $\mathcal{SA}$ distribution in the formed EC will be the same as the global $\mathcal{SA}$ distribution in $\mathcal{DB}$, preventing adversaries from gaining extra information on $\mathcal{SA}$ values.

However, we actually wish to allow for some variation in $\mathcal{SA}$ distributions,

observing the β-likeness constraint. Thus, we can follow a more relaxed approach, in which again all tuples of the same $\mathcal{SA}$ value still appear in a single bucket, but now more than one $\mathcal{SA}$ values may co-exist in one bucket. Again we form ECs choosing tuples proportionally to the sizes of the buckets they hail from, *without* discriminating between different $\mathcal{SA}$ values therein. This way, the $\mathcal{SA}$ distribution in the created EC is not guaranteed to be the same as that in $\mathcal{DB}$. Some privacy is sacrificed for the sake of flexibility and information quality. We do aim at such information quality, to the extent allowed by the β-likeness constraint. The following two definitions along with the example clarify the above intuition. Our analysis borrows some formulations from SABRE (see Chapter 3), our work where we develop an algorithm for the t-closeness model.

Definition 4.6 (bucket partition (Chapter 3))**.** *Let $\mathcal{DB}$ be a table with sensitive attribute $\mathcal{SA}$, we say that a set of buckets φ forms an* exact bucket partition *of $\mathcal{DB}$ if and only if $\bigcup_{\forall \mathcal{B} \in \varphi} \mathcal{B} = \mathcal{DB}$ and each $\mathcal{SA}$ value, and each tuple, appears in* exactly one *bucket.*

Definition 4.7 (proportionality requirement (Chapter 3))**.** *Let φ be a bucket partition of table $\mathcal{DB}$. Assume that an EC, $\mathcal{G}$, is formed with x_j tuples from bucket $\mathcal{B}_j \in \varphi$, $j = 1, 2, \ldots, |\varphi|$. $\mathcal{G}$ abides to the* proportionality requirement *with respect to φ, if and only if the values x_j are proportional to the sizes of buckets $|\mathcal{B}_j|$, i.e., $x_1 : x_2 : \ldots : x_{|\varphi|} = |\mathcal{B}_1| : |\mathcal{B}_2| : \ldots : |\mathcal{B}_{|\varphi|}|$.*

Name	Weight	Age	Disease
Mike	70	40	SARS
John	60	60	pneumonia
Bob	50	50	bronchitis
Alice	70	50	intestinal cancer
Beth	80	50	hepatitis
Carol	60	70	gastric ulcer

Table 4.2: Patient records

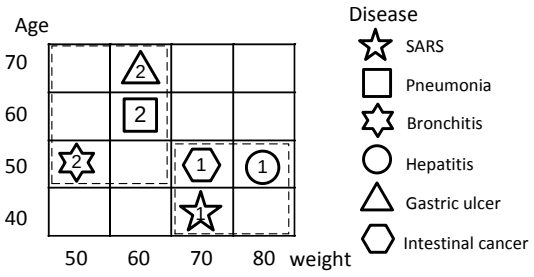


Figure 4.2: Better information quality

Example 4.1. *Consider Table 4.2, where* {`weight, age`} *is the QI, and* `disease` *is the* $\mathcal{SA}$*. Figure 4.2 shows the QI-space and the distribution of tuples, with each QI attribute corresponding to a dimension. A bucket partition* φ *of this table could consist of six buckets of one tuple each,* $b_1, b_2, \ldots, b_6$*, with* $\mathcal{SA}$ *values* SARS*,* pneumonia*,* bronchitis*,* hepatitis*,* gastric ulcer*, and* intestinal cancer*, respectively. Taking one tuple from each of these buckets, we could build a single EC satisfying* 0*-likeness. Still, such an EC covers the entire QI-space, incurring high information loss. An alternative bucket partition could consist of three buckets of two tuples each,* $\varphi = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$*, with* SARS *and* pneumonia *in bucket* $\mathcal{B}_1$*,* bronchitis *and* hepatitis *in* $\mathcal{B}_2$*, and the rest in* $\mathcal{B}_3$*. We can then build two ECs, by taking one tuple from each of these buckets, as shown in Figure 4.2. Tuples in the same EC are labeled by the same number in the figure. This EC partitioning achieves better information quality, as the minimum bounding boxes of ECs in QI-space are smaller.*

A relaxed bucket partition, as in the above example, enables higher information quality, but no longer abides by 0-likeness. Still, it suffices to create ECs in such a way that they obey β-likeness for a given β. For instance, the anonymization produced in Example 4.1 satisfies β-likeness (for $\beta \geq 1$) with respect to the distribution of `disease` in Table 4.2.

Following the above observations, we propose our β-likeness scheme, whose main body is composed of two phases—first, the *bucketization* phase partitions tuples into a set of buckets, so that ECs formed under the *proportionality requirement* obey β-likeness; then, the *redistribution* phase relaxes the proportionality requirement to further improve information quality, and dynamically

determines the size of each possible EC. After the EC sizes are calculated, tuples are retrieved from buckets in a way that aims to preserve information quality, and concrete ECs are generated over them.

4.3.1 Bucketization phase

Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the domain of $\mathcal{SA}$, and $\mathcal{P} = (p_1, p_2, \ldots, p_m)$ be the overall distribution of $\mathcal{SA}$ values in a table $\mathcal{DB}$. We partition $\mathcal{V}$ into subsets, and use them to divide $\mathcal{DB}$ into a bucket partition φ; that is, all tuples in $\mathcal{DB}$ with $\mathcal{SA}$ values in the same subset of $\mathcal{V}$ are pushed to a single bucket of φ. Suppose that EC $\mathcal{G}$ is formed by taking x_j tuples from bucket $\mathcal{B}_j \in \varphi$, $j = 1, 2, \ldots, |\varphi|$. Let V_j be the subset of $\mathcal{SA}$ values contained in bucket $\mathcal{B}_j$. In a worst-case scenario, *all* x_j tuples may have the *least frequent* $\mathcal{SA}$ value in V_j, v_{ℓ_j}, with $p_{\ell_j} = \min_{v_i \in V_j}\{p_i\}$, hence the frequency of v_{ℓ_j} in $\mathcal{G}$ will be $q_{\ell_j} = \frac{x_j}{|\mathcal{G}|}$. For β-likeness to hold, it should hold in this worst-case scenario, i.e., $\frac{x_j}{|\mathcal{G}|} \leq f(p_{\ell_j}) = (1 + \min\{\beta, -\ln(p_{\ell_j})\}) \cdot p_{\ell_j}$. We then arrive at the following theorem.

Theorem 4.1 (Eligibility Condition)**.** *Let φ be a bucket partition of table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, $\mathcal{G}$ an EC formed with x_j tuples from bucket $\mathcal{B}_j \in \varphi$, V_j the set of $\mathcal{SA}$ values in bucket $\mathcal{B}_j \in \varphi$, and $p_{\ell_j} = \min_{v_i \in V_j}\{p_i\}$, $j = 1, 2, \ldots, |\varphi|$. If $\forall j \in \{1, 2, \ldots, |\varphi|\}$, $\frac{x_j}{|\mathcal{G}|} \leq f(p_{\ell_j})$, then $\mathcal{G}$ follows β-likeness.*

Proof. For any $\mathcal{SA}$ value $v_k \in \mathcal{V}$, let $\mathcal{B}_j \in \varphi$ be the single bucket that contains tuples in $\mathcal{DB}$ with v_k as their $\mathcal{SA}$ value, hence $v_k \in V_j$. Since $\mathcal{G}$ draws x_j tuples from $\mathcal{B}_j$, the frequency of v_k in $\mathcal{G}$ is $q_k \leq \frac{x_j}{|G|} \leq f(p_{\ell_j}) \leq f(p_k)$. Expanding to all $v_k \in \mathcal{V}$, we conclude that $\mathcal{G}$ follows β-likeness. □

Theorem 4.1 defines the *eligibility condition* for an EC to follow β-likeness. However, it does not provide a way to specify a particular number of tuples

x_j to choose from a given bucket $\mathcal{B}_j$. In other words, although it effectively allows to *verify* β-likeness, it offers no guidance for *constructing* a β-likeness-complying anonymization. To overcome this lack of guidance, we start out by assuming that ECs are formed following the *proportionality requirement*. Under this assumption, it holds that $\frac{x_j}{|\mathcal{G}|} = \frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{v_i \in V_j} p_i$, and the next lemma can be easily deduced from Theorem 4.1.

Lemma 4.4. *Let $\mathcal{G}$ be an EC that follows the proportionality requirement with respect to a bucket partition φ of table $\mathcal{DB}$ with sensitive attribute $\mathcal{SA}$, V_j the set of $\mathcal{SA}$ values in bucket $\mathcal{B}_j \in \varphi$, and $p_{\ell_j} = \min_{v_i \in V_j}\{p_i\}$, $j = 1, 2, \ldots, |\varphi|$. If $\forall j \in \{1, 2, \ldots, |\varphi|\}$, $\sum_{v_i \in V_j} p_i \leq f(p_{\ell_j})$, then $\mathcal{G}$ follows β-likeness.*

In effect, Lemma 4.4 determines the condition that the frequencies of a subset of $\mathcal{SA}$ values, $V_j \subset \mathcal{V}$, should obey, so that, if the values in V_j are put in the same bucket B_j of a bucket partition φ, then ECs obeying the proportionality requirement over φ will satisfy β-likeness. This condition is trivially satisfied by a bucket partition with a *single* $\mathcal{SA}$ value per bucket. However, such a bucket partition would force every EC to follow a strict proportionality requirement over a large number of buckets, leading to anonymizations of poor information quality. We should better strive to achieve a *loose* bucket partition that satisfies the condition of Lemma 4.4 in a non-trivial manner, having *more than one* distinct $\mathcal{SA}$ values per bucket and as few buckets as possible (as in Example 4.1).

We develop a *bucketization* scheme for this task. We start out by representing, $\mathcal{P}$, the set of $\mathcal{SA}$ frequencies in $\mathcal{DB}$, in ascending order so that, $p_i \leq p_{i+1}$, $i = 1, \ldots, m-1$. By Lemma 4.4, a set of consecutive $\mathcal{SA}$ values in $\mathcal{V}$, $v_b, v_{b+1}, \ldots, v_e$, are allowed to be in the same bucket provided that $\sum_{i=b}^{e} p_i < f(p_\ell)$, where $p_\ell = \min\{p_b, p_{b+1}, \ldots, p_e\}$.

Our bucketization scheme, presented in Function DPpartition, partitions $\mathcal{V}$

by dynamic programming, aiming to minimize the number of buckets. Let $N[e]$ denote the minimum number of buckets to which we can partition the prefix of e elements in $\mathcal{V}$, i.e., $v_1, v_2, \ldots, v_e$. The value of $N[e]$ is calculated recursively as:

$$N[e] = \min_{\{b|\mathsf{Combinable}(b,e)=true\}} \{N[b-1]\} + 1 \tag{4.2}$$

Function $\mathsf{Combinable}(b, e)$ checks whether $\mathcal{SA}$ values $v_b, \ldots, v_e$, $b < e$ are allowed to be in the same bucket, i.e., whether the inequality $\sum_{i=b}^{e} p_i < f(p_\ell)$, holds, with $p_\ell = \min\{p_b, p_{b+1}, \ldots, p_e\}$. The base case is $N(0) = 0$.

Function DPpartition ($\mathcal{DB}$, $\mathcal{SA}$)

1 Let $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ be the domain of $\mathcal{SA}$, with overall distribution $\mathcal{P} = (p_1, p_2, \ldots, p_m)$ in $\mathcal{DB}$;
2 Assume that $p_n \leq p_{n+1}$, where $n = 1, 2, \ldots, m-1$;
3 $N[0] = 0$;
4 $S[0] = 0$;
5 **for** *e=1 to* m **do**
6 $\quad N[e] = N[e-1] + 1$;
7 $\quad S[e] = e$;
8 $\quad b = e - 1$;
9 $\quad$ **while** $b > 0$ *and* $\mathsf{Combinable}(b, e) = true$ **do**
10 $\quad\quad$ **if** $N[b-1] + 1 < N[e]$ **then**
11 $\quad\quad\quad N[e] = N[b-1] + 1$;
12 $\quad\quad\quad S[e] = b$;
13 $\quad\quad b = b - 1$;
14 Initialize φ to be empty;
15 $e = m$;
16 **while** $e > 0$ **do**
17 $\quad b = S[e]$;
18 $\quad$ Create a new bucket $\mathcal{B}$, containing tuples with $\mathcal{SA}$ values in $\{v_b, v_{b+1}, \ldots, v_e\}$;
19 $\quad \varphi = \varphi \cup \{\mathcal{B}\}$;
20 $\quad e = S[e] - 1$;
21 Return φ;

Function DPpartition consists of two parts. The first part (steps 3-13) runs the DP recursion of Equation 4.2 (recording the value of b that minimizes the number of segments for the first e elements - step 12) to evaluate the final minimum value $N[m]$ and split $\mathcal{V}$ into segments accordingly; thereby it needs to assess the combinability of m^2 possible buckets of consecutive values in $\mathcal{V}$. To

assess combinability, we maintain the running $\sum p_i$ within a bucket, updated in $O(1)$ at each step, while the $\min\{p_i\}$ within a bucket is simply its first element. The complexity of this part is $O(m^2)$. The second part (steps 14-20) uses the results of the first part to build the derived bucket partition. Tuples with the $\mathcal{SA}$ values in a segment are pushed into a corresponding bucket (step 18), in $O(|\mathcal{DB}|)$. The overall time complexity of the algorithm is $O(m^2 + |\mathcal{DB}|)$.

4.3.2 Redistribution phase

The bucketization phase of our scheme delivers a bucket partition φ of $\mathcal{DB}$. We have so far assumed, as a starting point, that ECs are formed from φ strictly following the proportionality requirement, so as to satisfy β-likeness. However, a strict adherence to the proportionality requirement may result in large ECs, incurring high information loss. For example, if the size of some bucket $\mathcal{B}_j \in \varphi$ is a prime number (other than 2), then, in order to strictly follow the proportionality requirement, we should form an EC out of the whole table. We should rather relax the proportionality requirement: it should suffice that the number of tuples x_j chosen from bucket $\mathcal{B}_j$ in EC $\mathcal{G}$ be *approximately* proportional to the size of $\mathcal{B}_j$, i.e., $\frac{x_j}{|\mathcal{G}|} \approx \frac{|\mathcal{B}_j|}{|\mathcal{DB}|} = \sum_{v_i \in V_j} p_i$. The rationale for this relaxation is as follows. The bucket partition φ returned by DPpartition obeys the inequality $\sum_{v_i \in V_j} p_i \leq f(p_{\ell_j})$ (Lemma 4.4), where V_j is the set of $\mathcal{SA}$ values in bucket $\mathcal{B}_j \in \varphi$ and v_{ℓ_j} is the least frequent value in V_j. It follows that, if $\frac{x_j}{|\mathcal{G}|} \approx \sum_{v_i \in V_j} p_i$ (i.e., if we draw tuples into ECs approximately proportionally to the size of the bucket in φ they hail from), then the *eligibility condition* $\frac{x_j}{|\mathcal{G}|} \leq f(p_{\ell_j})$ (Theorem 4.1), and hence β-likeness, will be easy to achieve.

To ensure β-likeness, we determine the EC sizes to use in the anonymization by constructing a binary tree, the ECTree, in a top-down fashion. We start with a

bucket partition $\varphi = \{\mathcal{B}_1, \ldots, \mathcal{B}_{|\varphi|}\}$. The root of the tree r represents a potential EC that contains all tuples in $\mathcal{DB}$, i.e., $|\mathcal{B}_j|$ tuples from bucket $\mathcal{B}_j$. We denote the contents of the root as $r = \left[|\mathcal{B}_1|, \ldots, |\mathcal{B}_{|\varphi|}|\right]$. This can be a valid EC, but we prefer to have smaller ones. Then, we proceed to split r into two children (each representing an EC) by dichotomizing each $\mathcal{B}_j$ into $\mathcal{B}_j^1$ and $\mathcal{B}_j^2$. The left child c_L contains $\mathcal{B}_j^1$ and the right child c_R contains $\mathcal{B}_j^2$, $j = 1, 2, \ldots, |\varphi|$. We ensure that $\mathcal{B}_j^1$ and $\mathcal{B}_j^2$ have *approximately* the same size by setting $|\mathcal{B}_j^1| = \mathsf{round}\left(\frac{|\mathcal{B}_j|}{2}\right)$ and $|\mathcal{B}_j^2| = |\mathcal{B}_j| - |\mathcal{B}_j^1|$. The split is allowed only if both c_L and c_R satisfy the eligibility condition (Theorem 4.1), i.e., they can form ECs satisfying β-likeness. Assume the left child of r is $c_L = \left[|\mathcal{B}_1^1|, \ldots, |\mathcal{B}_{|\varphi|}^1|\right]$. Then, for the eligibility condition to be satisfied, it should hold that $\frac{|\mathcal{B}_j^1|}{\sum_{n=1}^{|\varphi|} |\mathcal{B}_n^1|} \leq f(p_{\ell_j})$, where p_{ℓ_j} is the frequency of the *least frequent* (in $\mathcal{DB}$) $\mathcal{SA}$ value included in $\mathcal{B}_j$. An analogous condition applies for the right child c_R. If splitting r into c_L and c_R is allowed, we proceed to check whether we can split c_L and c_R themselves. The splitting terminates when no node can be split any further. Then we get a final ECTree, in which each leaf node configures the number of tuples an EC should get from each bucket. A function biSplit(φ) returns the list of leaf nodes. Example 4.2 illustrates this process.

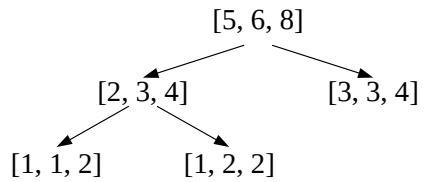


Figure 4.3: An example of dynamically determining EC sizes

Example 4.2. *Let* `disease` *be a categorical $\mathcal{SA}$ with the domain hierarchy of figure 4.1. Consider a table, containing 2 tuples with* `SARS`*, 3 with* `pneumonia`*, 3 with* `bronchitis`*, 3 with* `hepatitis`*, 4 with* `gastric ulcer`*, and 4 with* `intestinal cancer`*. Assume that $\beta = 2$. Therefore, the overall $\mathcal{SA}$*

distribution is $\mathcal{P} = (p_1, p_2, p_3, p_4, p_5, p_6) = (\frac{2}{19}, \frac{3}{19}, \frac{3}{19}, \frac{3}{19}, \frac{4}{19}, \frac{4}{19})$. $f(p_1) \approx 0.31$, $f(p_2) = f(p_3) = f(p_4) \approx 0.45$, *and* $f(p_5) = f(p_6) \approx 0.54$. ***By the bucketization phase (see algorithm*** DPpartition***) we have a bucket partition of the table,*** $\varphi = \{B_1, B_2, B_3\}$, *where* B_1 *accommodates tuples with* `SARS` *and* `pneumonia`, B_2 *has* `bronchitis` *and* `hepatitis`, *and* B_3 *contains the remaining two. The root node* $r = [5, 6, 8]$ *in Figure 4.3 represents an EC with 5 tuples from* B_1, *6 from* B_2, *and 8 from* B_3 *(i.e., all tuples in the table). We split* r *into* $c_1 = [2, 3, 4]$ *and* $c_2 = [3, 3, 4]$. ***Then EC*** c_1 ***has a size 9, and contains 2 tuples from*** B_1 ***with*** $\frac{2}{9} < min\{f(p_1), f(p_2)\}$, ***3 tuples from*** B_2 ***with*** $\frac{3}{9} < \min\{f(p_3), f(p_4)\}$, ***and 4 tuples from*** B_3 ***with*** $\frac{4}{9} < \min\{f(p_5), f(p_6)\}$. ***Thus,*** c_1 ***obeys the eligibility condition (Theorem 4.1). Similarly, we can verify that*** c_2 ***also satisfies the condition. Therefore, splitting*** r ***into*** c_1 ***and*** c_2 ***is allowed. Recursively, we can split*** c_1 ***into*** $[1, 1, 2]$ ***and*** $[1, 2, 2]$. ***When we try to split*** c_2 ***into*** $g_1 = [1, 1, 2]$ ***and*** $g_2 = [2, 2, 2]$, ***we find*** g_2 ***does not satisfy the eligibility condition (Theorem 4.1), because*** $\frac{2}{6} > \min\{f(p_1), f(p_2)\}$, ***hence this splitting is not allowed. Figure 4.3 is the final tree, with each leaf node showing the number of tuples a possible EC should draw from each bucket. In this case, no EC draws 0 tuples from a bucket, but that can be allowed in the general case, conferring flexibility to the algorithm.***

4.3.3 BUREL

We now put the above two phases together to devise BUREL, our algorithm that first *BU*cketizes tuples into buckets and then *RE*distributes them from buckets to ECs to attain β-*L*ikeness. BUREL and SABRE (Chapter 3) have two phases of common names. However, their internal workings, and the theoretical analysis that accompanies them, are different and particular to each scheme and model.

In its own bucketization phase, SABRE organizes $\mathcal{SA}$ values in a tree and hashes them into buckets by the tree hierarchy so that similar $\mathcal{SA}$ values stay in the same bucket. On the other hand, BUREL sorts the $\mathcal{SA}$ values by ascending order of frequencies and employs dynamic programming to partition them in segments of $\mathcal{SA}$ values with similar frequencies. In the redistribution phase, both SABRE and BUREL use a binary tree to split ECs. However, each algorithm caters to the requirements of the model it is designed for, extracted from the corresponding theoretical analysis, when checking the eligibility of a split.

Algorithm: BUREL ($\mathcal{DB}$, $\mathcal{SA}$, β)

1 Let $\{v_1, v_2, \ldots, v_m\}$ be all the $\mathcal{SA}$ values in $\mathcal{DB}$, and $\{p_1, p_2, \ldots, p_m\}$ be their distributions;
2 $\varphi = \mathsf{DPpartition}(\mathcal{DB}, \mathcal{SA})$;
3 $S_a = \mathsf{biSplit}(\varphi)$;
4 **foreach** *array* a *in* S_a **do**
5 Create an empty EC, say $\mathcal{G}$;
6 **foreach** a_j, j^{th} *element of* a **do**
7 $ec_j = \mathsf{Retrieve}(\mathcal{B}_j, a_j)$;
8 add ec_j to $\mathcal{G}$;
9 Output $\mathcal{G}$;

The *bucketization phase* of BUREL returns φ, a bucket partition of $\mathcal{DB}$ (step 2). Then, its *redistribution phase* (function biSplit) determines the *size* of each possible EC as a leaf in the ECTree and returns a list of arrays S_a (step 3). Each array contains the size values in a leaf of the ECTree. Then, specific ECs following the prescribed sizes are materialized and output (steps 4-9). Given an array $a \in S_a$, BUREL retrieves a_j tuples from bucket $\mathcal{B}_j \in \varphi$, where a_j is the j^{th} element of a and $j = 1, 2, \ldots, |\varphi|$, and forms an EC $\mathcal{G}$ out of the retrieved tuples (steps 6-8).

When retrieving tuples from a bucket, BUREL does not consider their SA values. The β-likeness between the constructed EC $\mathcal{G}$ and the whole table $\mathcal{DB}$ is guaranteed by Theorem 4.1. Instead, it determines which tuples to select

by information loss considerations, as prescribed by the General Loss Metric (GLM) (Section 2.3). GLM requires the Minimum Bounding Boxes of ECs to be as small as possible. Thus, function Retrieve$(\mathcal{B}_i, a_i)$ (step 7) greedily picks tuples of as similar as possible QI values. Our greedy strategy utilizes the notion of *nearest neighbors*. We define a multidimensional space with each of the QI attributes as a dimension. The mapping to such a QI-space for a numerical QI attribute NA is straightforward [41]. The axis of a categorical QI attribute CA is formed by the order provided by a pre-order traversal of the leaves in its domain hierarchy $\mathcal{H}_{CA}$. Each tuple is represented as a point in this QI-space.

In order to form an EC $\mathcal{G}$, BUREL first randomly picks a tuple x from a bucket of φ in $\mathcal{G}$. Then, the function Retrieve finds the a_j nearest neighbors (by Euclidean distance) of x in *each* bucket $\mathcal{B}_j$, $j = 1, 2, \ldots, |\varphi|$, and adds them into $\mathcal{G}$, until the size specifications are satisfied. Still, this process can be computationally demanding even with a sophisticated index structure [34]. Thus, we suggest a more efficient heuristic method using the *Hilbert space-filling curve* [59], a continuous fractal that can map regions of QI-space, hence tuples, to 1D Hilbert values, as in [41]. Tuples that are close in the QI-space are also likely to have nearby Hilbert values. Thus BUREL sorts all tuples in a bucket $\mathcal{B}_j$ by the ascending order of their Hilbert values. Thereafter it uses this order to select the a_j nearest neighbors of a tuple x within *each* bucket. In our implementation, we find the nearest Hilbert-neighbor $\bar{x}$ of x within a bucket $\mathcal{B}_j$ by binary search, and then expand to the next closest a_j neighbors to x. The average time complexity for this search operation is $O\left(|S_\mathcal{G}||\varphi| \cdot \left(\log \frac{|\mathcal{DB}|}{|\varphi|} + \frac{|\mathcal{DB}|}{|S_\mathcal{G}||\varphi|}\right)\right)$, where $|\varphi|$ is the number of buckets, $\frac{|\mathcal{DB}|}{|\varphi|}$ the average size of a bucket, and $\frac{|\mathcal{DB}|}{|S_\mathcal{G}||\varphi|}$ the average number of tuples taken out from a bucket to form an EC.

4.3.4 BUREL for extended β-likeness

We now extend BUREL to semantic and range-based β-likeness. Instead of accepting each generated EC as it is, we buffer it in an EC list $L_{\mathcal{G}}$. After the generation of all possible ECs, we check each EC in $L_{\mathcal{G}}$. Given an EC $\mathcal{G}_1$, its distance from another EC $\mathcal{G}_2$ is calculated as $\mathcal{IL}(\mathcal{G}_1 \cup \mathcal{G}_2) - \mathcal{IL}(\mathcal{G}_1)$, i.e., as the increased information loss of $\mathcal{G}_1$ after being merged with $\mathcal{G}_2$. If an EC does not satisfy extended β-likeness, we greedily merge it with its nearest neighbors, one by one, until extended β-likeness is satisfied. Such a result is always possible, since extended β-likeness obeys the monotonicity property (Lemma 4.2). To find the nearest neighbors of an EC, we scan the whole list $L_{\mathcal{G}}$. Therefore, the time complexity of the whole process is $O(|L_{\mathcal{G}}|^2)$. We emphasize that BUREL for range-based β-likeness provides a robust solution to the problem of anonymization under a numerical $\mathcal{SA}$, studied in [51, 86].

4.4 Experiments

In this section we evaluate the performance of BUREL. Our prototype was implemented in Java and the experiments were run on a Core2 Duo 2.33GHz CPU machine with 4GB RAM running Windows XP. In our experimental study, we have used the CENSUS dataset[2] [3], which contains $500,000$ tuples on 6 attributes as shown in Table 4.3. For categorical attributes, the value following the type is the height of the corresponding attribute hierarchy; for instance, attribute *marital status* is categorical and has a hierarchy of height 2. The first 5 attributes are potential QI-attributes, and the last one (i.e., *salary class*) is the sensitive attribute. By default, we take the first three attributes as QI. The least frequent

[2]This dataset is the same as that in the experiments of SABRE (Chapter 3). We select 6 out of the 8 attributes to avoid the curse of high dimensionality [8], thus better preserving information.

value of $\mathcal{SA}$ *salary class* is 49, with frequency 0.2018%; the most frequent $\mathcal{SA}$ value is 12, with frequency 4.8402%. The value $\beta = 1$ produces frequency threshold $e^{-\beta} \approx 37\%$, which marks all $\mathcal{SA}$ values as 'infrequent', and allows the frequency of any $\mathcal{SA}$ value in any EC to be at most $4.8402\% \times 2 = 9.7\%$. Thus, 1 is already a small threshold of β. In our experiments, we use $\beta \in \{1, 2, 3, 4, 5\}$. We generate 5 microdata tables by randomly taking $100,000$ to $500,000$ tuples from the dataset; the one with $500,000$ tuples is our default dataset.

Attribute	Cardinality	Type
Age	79	numerical
Gender	2	categorical (1)
Education Level	17	numerical
Marital Status	6	categorical (2)
Work Class	10	categorical (3)
Salary Class	50	sensitive attribute

Table 4.3: The CENSUS dataset

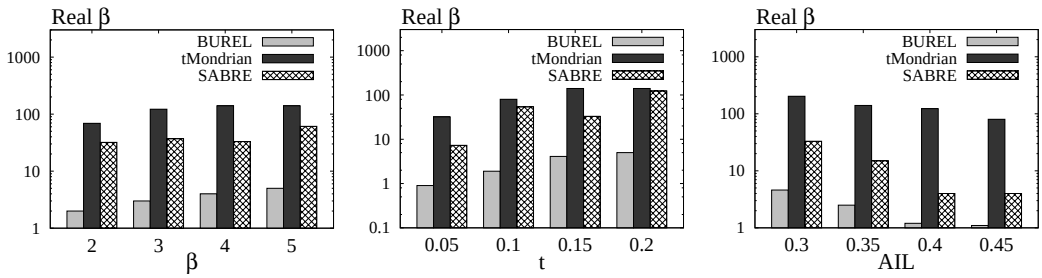
We treat the likeness threshold β as a variable, set by default to 4. Then, for any $\mathcal{SA}$ value v_i, if $p_i \leq e^{-4} = 0.018$, then its frequency q_i in any EC should not exceed $5p_i$; if $p_i > 1.8\%$, then $q_i \leq (1 - ln(p_i)) \cdot p_i$. The highest $\mathcal{SA}$ value frequency in our data set does not exceed 5%, so the frequency of any salary value in any EC will not exceed 20%.

As we have discussed, there are two predecessor distribution-based models that β-likeness can be compared to: t-closeness and δ-disclosure. The latter can be configured to achieve a privacy guarantee comparable, though not equivalent, to β-likeness. On the other hand, t-closeness cannot be configured in a similar fashion. Thus, in order to compare our β-likeness scheme to existing t-closeness schemes, we configure all compared schemes to achieve either the same privacy in terms of t-closeness, or the same information loss, on a given

data, and compare the privacy guarantee they achieve in terms of β-likeness; this approach creates an even playing field. Then, we evaluate BUREL on information loss, runtime, and accuracy on aggregate queries; for this evaluation, we compare it to both the δ-disclosure scheme used in [21] (which is an adaptation of Mondrian [49, 50]), as well as to a similar β-likeness scheme that we devise by adapting Mondrian to the new model. Finally, we also evaluate BUREL on range-based β-likeness.

4.4.1 Face-to-face with t-closeness

Our first task is to compare our new β-likeness privacy model to the predecessor distribution-based model of t-closeness. We argue that β-likeness provides a more informative and comprehensible privacy guarantee than t-closeness does. Still, in order to create an even playing field on which to compare β-likeness to t-closeness, we conducted three face-to-face comparisons as follows.



(a) vary beta (b) vary t (c) vary AIL

Figure 4.4: Comparison to t-closeness

In the first comparison, for a given dataset $\mathcal{DB}$ and β, we let BUREL transform $\mathcal{DB}$ to $\mathcal{DB}_\beta$, satisfying β-likeness. We then measure the closeness t_β, by the t-closeness model, between $\mathcal{DB}_\beta$ and $\mathcal{DB}$, i.e., the maximum EMD of the $\mathcal{SA}$ distribution in an EC of $\mathcal{DB}_\beta$ from its distribution in $\mathcal{DB}$. We then apply t-closeness schemes tMondrian [53] and SABRE on $\mathcal{DB}$ as well, with t_β

as the t-closeness threshold, to produce $\mathcal{DB}^M_{t_\beta}$ and $\mathcal{DB}^S_{t_\beta}$, respectively. Therefore, data sets $\mathcal{DB}_\beta$, $\mathcal{DB}^M_{t_\beta}$, and $\mathcal{DB}^S_{t_\beta}$ all achieve the same privacy under the criterion of t-closeness, as expressed by the value of t_β. Then we measure the β value achieved by $\mathcal{DB}^M_{t_\beta}$ and $\mathcal{DB}^S_{t_\beta}$ with respect to $\mathcal{DB}$. Given that all three schemes achieve the same privacy in terms of t-closeness, we are interested to compare the privacy they ahieve in terms of β-likeness. Figure 4.4(a) shows the results (in logarithmic y-axis), as a function of the β parameter. Remarkably, while all the three schemes are tuned to ensure the same t-closeness guarantee, BUREL provides consistently higher privacy by the criterion of β-likeness than SABRE and tMondrian. This result is expected, since t-closeness restricts only the cumulative difference between $\mathcal{SA}$ distributions, indifferent to the relative frequency difference of each individual $\mathcal{SA}$ value between an EC and the whole table.

In the second comparison, for a given dataset $\mathcal{DB}$ and a closeness constraint t, we let tMondrian (SABRE) transform $\mathcal{DB}$ to $\mathcal{DB}^M_t$ ($\mathcal{DB}^S_t$), abiding by t-closeness. We then let BUREL find, by binary search, a value β_t, such that, when β-likeness is enforced on $\mathcal{DB}$ by BUREL for β_t, it produces an anonymization $\mathcal{DB}_{\beta_t}$ characterized by the same (or smaller) closeness parameter t as $\mathcal{DB}^M_t$ ($\mathcal{DB}^S_t$). Thus, again we get three anonymized versions of $\mathcal{DB}$ that achieve the same degree of privacy under t-closeness. While in our first comparison we arrived at this state starting out with a β parameter, now we do so starting out with a t parameter. Therefore, our results are not biased against t-closeness schemes. As before, we compare the β-likeness achieved by $\mathcal{DB}^M_t$ ($\mathcal{DB}^S_t$) to that achieved by $\mathcal{DB}_{\beta_t}$, as a function of t. The results, shown in Figure 4.4(b), reaffirm our previous findings.

In our last experiment, given an AIL value l, we let BUREL determine, by

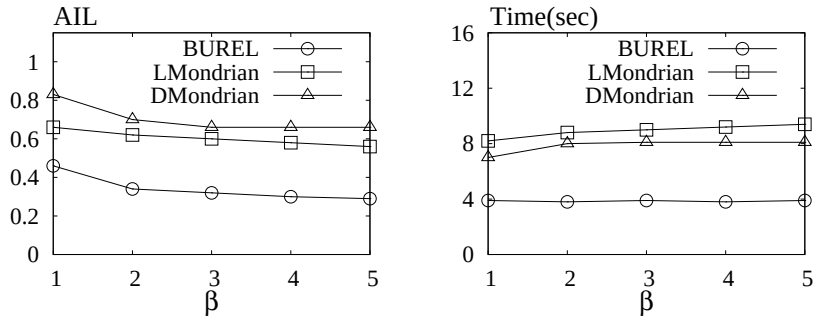
binary search on its β threshold, a value β_l, such that the data set $\mathcal{DB}_{\beta_l}$ generated by BUREL from $\mathcal{DB}$ with β_l as the likeness threshold achieves AIL equal to (or smaller than) l. Similarly, we determine, by binary search, a value t_l^M (t_l^S), which, taken as the closeness threshold in algorithm tMondrian (SABRE), generates data set $\mathcal{DB}_{t_l^M}$ ($\mathcal{DB}_{t_l^S}$) with AIL near l too, allowing for a small difference ϵ. Thus, we obtain three data sets $\mathcal{DB}_{\beta_l}$, $\mathcal{DB}_{t_l^M}$, and $\mathcal{DB}_{t_l^S}$, generated by BUREL, tMondrian, and SABRE, respectively, which all have information loss near l; to ensure the comparison is not biased in favor of BUREL, we ensure its AIL value is *not greater* than those of the other algorithms. We then compare the privacy they achieve in terms of β-likeness. Figure 4.4(c) shows the results. Not surprisingly, BUREL provides the highest privacy again, followed by SABRE and tMondrian.

The above experiments testify that, with other relevant factors being equal, state-of-the-art t-closeness schemes fail by a wide margin (as indicated by the logarithmic y-axes) to achieve privacy good in terms of β-likeness. Thus, they reaffirm that β-likeness is a privacy model raising substantially different requirements from those of t-closeness and requiring a different approach.

4.4.2 Performance evaluation

In this section we evaluate the performance of BUREL as a β-likeness algorithm in its own field. As there is no previous work on β-likeness, we employ two comparison benchmarks adopting some suggestions of related work. First, we devise an algorithm for β-likeness, following the conventional wisdom on designing algorithms for new privacy models: We adapt Mondrian

[49, 50], a k-anonymization algorithm, to the purposes of β-likeness, as previous works have done for other privacy models [21, 52, 54, 57]. Our adaptation, LMondrian, splits an EC only if both resultant ECs satisfy β-likeness. Second, in a similar way we extend Mondrian to a δ-disclosure [21] scheme, DMondrian. To render DMondrian comparable to BUREL and LMondrian, we set the value of δ so that the data anonymized by DMondrian obey β-likeness. As we have discussed, while β-likeness demands that an $\mathcal{SA}$ value's distribution in an EC be $q_i \leq (1+\min\{\beta,-\ln p_i\}) \cdot p_i$, for a given β, δ-disclosure requires that $e^{-\delta} \cdot p_i < q_i < e^{\delta} \cdot p_i$, where p_i is the overall distribution of v_i in the whole dataset. Thus, an algorithm for δ-disclosure achieves β-likeness for $\delta \leq \log(1+\min\{\beta,-\ln p_i\})$, for all p_i; in view of all $\mathcal{SA}$ values in $\mathcal{V}$, we set $\delta = \log\left(1+\min\left\{\beta,-\ln\left(\max_{v_i \in \mathcal{V}}\{p_i\}\right)\right\}\right)$. We first compare the three schemes with respect to average information loss and wall-clock time.

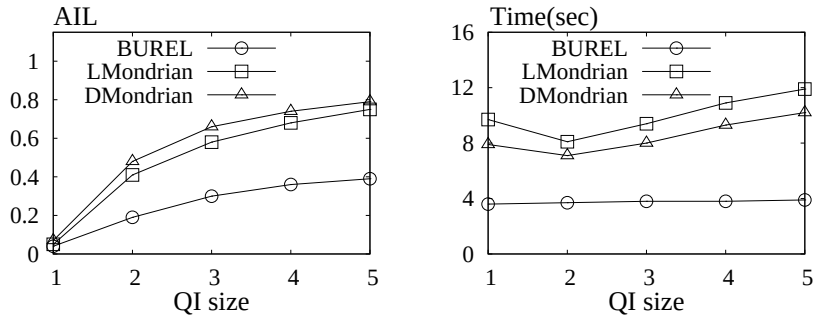


(a) information loss (b) time

Figure 4.5: Effect of varying β

First, we study performance as a function of the β threshold. Figure 4.5 shows the results. As β grows, the constraint on the relative difference of each $\mathcal{SA}$ (i.e., *salary*) value frequency between an EC and the overall table is relaxed, hence information quality rises (Figure 4.5(a)). BUREL outperforms both LMondrian and DMondrian in information quality, showing the benefit of a scheme tailored for β-likeness. This result reconfirms the finding of [21] that

a k-anonymization algorithm, adapted to δ-disclosure, yields unacceptably high information loss; as we discussed, we aim at a *positive* result and propose a *better* alternative. In addition, given that δ-disclosure overprotects data by imposing a constraint on *negative* information gain, LMondrian performs better than its stricter sibling, DMondrian. More remarkably, BUREL also outpaces both Mondrian-based schemes in time efficiency (Figure 4.5(b)). Overall, BUREL achieves almost half the information loss of its Mondrian-based competitors in about half the time.

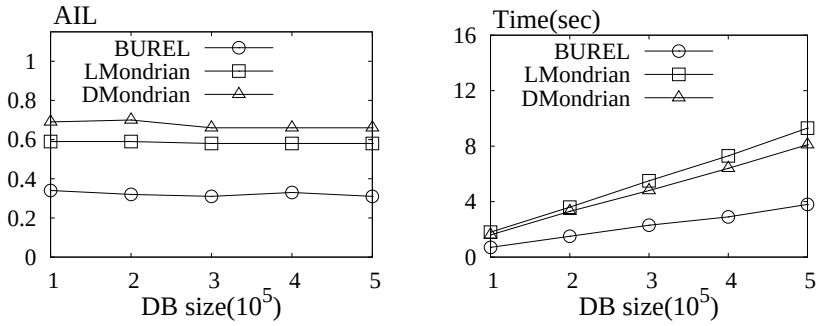


(a) information loss (b) time

Figure 4.6: Effect of varying QI

Next, we investigate the effect of QI dimensionality (size), varying it from 1 to 5. As QI dimensionality increases, the data become more sparse in QI space, as more high-dimensional degrees of freedom are offered; thus, the formed ECs are more likely to have large minimum bounding boxes, and information quality degrades, as Figure 4.6(a) shows. The information loss of BUREL is again lower than that of the Mondrian-based methods. In addition, BUREL is again the fastest of the three (Figure 4.6(b)).

Our next experiment studies the effect of database size, varying the size of the microdata table from 100K to 500K tuples. Figure 4.7 presents our results. Interestingly, data size has no clear effect on information quality. This is due to the fact that, as the amount of tuples grows, more sensitive values are revealed,



(a) information loss (b) time

Figure 4.7: Effect of varying dataset

imposing their own requirements. The mere increase of data density does not help, as it would with simpler models like k-anonymity. Still, the elapsed time increases as the table size grows; BUREL is again found to be superior in both respects.

We now study the utility of the anonymized data on aggregation queries, which are the basis of real data analysis tasks such as statistical analysis and data mining applications. We consider the following type of queries, as in [82]:

```
SELECT COUNT(*) FROM Anonymized-data
WHERE pred(A_1) AND ... AND pred(A_λ)
AND pred(SA)
```

Each A_i is a QI attribute. The query has predicates on λ randomly selected QI attributes and the $\mathcal{SA}$. For each of these $\lambda+1$ attributes A, $pred(A)$ has the form of $A \in R_A$, where R_A is an arbitrary interval in the domain of A. The *expected selectivity* over the table is $0 < \theta < 1$. Assuming data are uniformly distributed, θ can be achieved if each attribute A selects records within a range of length $|A| \cdot \theta_A$ of its domain, such that $(\theta_A)^{\lambda+1} = \theta$. In effect, the length of R_A should be $|A| \cdot \theta^{\frac{1}{\lambda+1}}$, where $|A|$ is the domain length of attribute A. Given a query, the precise result $prec$ is computed from the original table, and an estimated

result est is obtained from the anonymized table. To calculate est, we assume that tuples in each EC are uniformly distributed, and consider the intersection between the query and the EC. We define $\frac{|est-prec|}{prec} \times 100\%$ as the *relative error*. Our workload consists of 10K queries, and we measure the workload error as the *median relative error*. Relative error is undefined when $prec$ is 0. If $prec$ in a given query is 0, we drop that query.

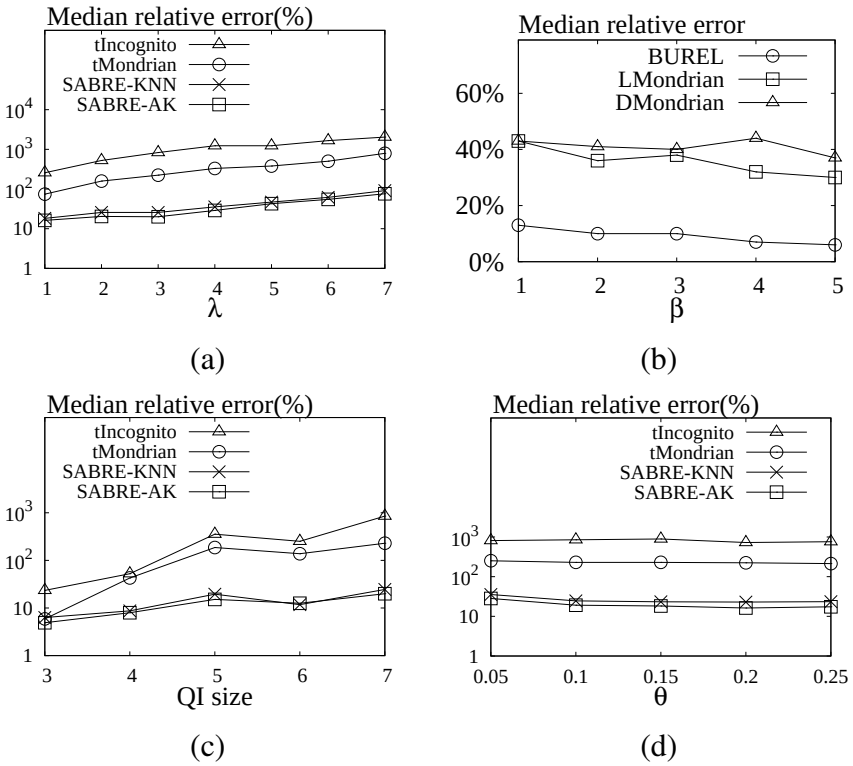


Figure 4.8: Median relative error

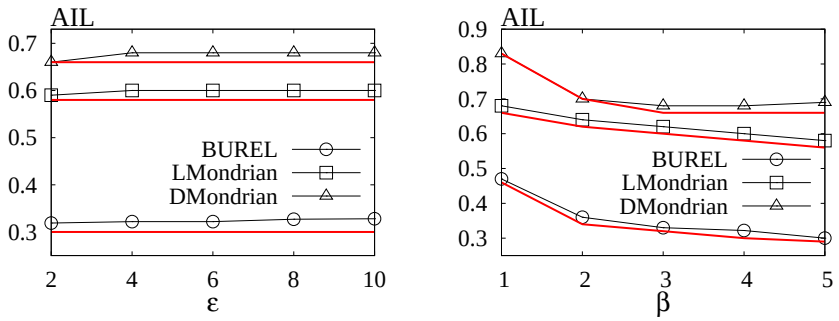
In our first experiment, we use the first 5 attributes in Table 4.3 as the QI, with expected selectivity $\theta = 0.1$, and vary the dimensionality of the query, i.e. the number of QI attributes λ on which predicates are defined. These attributes contribute to the error. Thus, the increase of λ exercises a negative effect on error. However, as λ grows, the length of the query range R_A in the domain of each queried attribute also grows (for constant θ); thereby, the minimum

bounding box of an EC becomes more likely to be entirely contained in the query region. In effect, the error does not depend monotonically on λ (Figure 4.8(a)); it does not matter much how many attributes a given selectivity θ is shared among. In the next experiment, we fix λ to 3, θ to 0.1, and vary β. Figure 4.8(b) shows the results. As β grows, the privacy requirement is relaxed, hence information quality rises and the error drops. Next, we set θ to 0.1, and vary the QI size. As the QI size increases, the data tend to be more sparse in QI-space, hence it is more likely that ECs with bigger bounding boxes are created. Thus, in Figure 4.8(c) the workload error increases with QI size. Finally, Figure 4.8(d) presents the results as a function of selectivity θ. As θ grows, the length of the range R_A for each attribute in a predicate increases. This makes the minimum bounding box of an EC more likely to be entirely contained in the query region, so the estimate becomes more accurate and the error smaller. In all experiments, BUREL achieves consistently better utility.

4.4.3 Extension to range-based β-likeness

Now we evaluate the extension of BUREL to range-based β-likeness. For the comparison, we adapted LMondrian (similarly for DMondrian), so that the split of ECs it performs is allowed only when both resultant ECs satisfy range-based β-likeness.

Figure 4.9 presents our results. As $\mathcal{SA}$ *Salary* is a numerical sensitive attribute, we employ a parameter ε that bounds the range of values by Definition 4.5. A bigger ε indicates that there are more salary values in a range, hence the privacy constraint becomes stronger and information quality degrades. Figure 4.9(a) shows the result with varying ε, with QI size 3 and $\beta = 4$. To show the extra information loss due to the constraint on range-based β-likeness, we



(a) varying ε (b) varying β

Figure 4.9: Range-based β-likeness

add three horizontal lines in the figure, which represent the information loss of the same schemes when only enhanced β-likeness is applied. Information loss grows slightly with ε; the additional requirement of range-based β-likeness incurs only marginally more information loss. This result reconfirms the robustness of β-likeness: an EC that satisfies β-likeness is usually likely to also fulfill its range-based variant. Figure 4.9(b) shows our results with ε set to 6 and varying β. Information quality grows with β, with three comparison lines added again. The results reconfirm that range-based β-likeness imposes a minimal premium on information quality.

4.5 Summary

This chapter has revisited the microdata anonymization problem, and made two contributions. First, we propose β-likeness, a robust privacy model that provides a comprehensible privacy guarantee, expressed as a limit on the relative positive information gain on each single sensitive attribute value. Second, we devise BUREL, an algorithm customized for that model itself. Our experimental results testify that the problem raised by the β-likeness concept cannot be treated by algorithms developed for other privacy models, and verify the effectiveness and

efficiency of BUREL in the task it aims to accomplish, as opposed to approaches adapting other algorithms to the new model.

CHAPTER 5

CASTLE: CONTINUOUSLY ANONYMIZING DATA STREAMS

So far, we have focused on static data sets. In this and the next chapters we will concentrate on the privacy preservation of data streams. Compared with traditional data sets, data streams are continuous and unbounded. Due to the constraints on performance and storage, algorithms designed for streaming data are permitted to make only one pass over the data; backtracking is not allowed. These unique requirements arising in the context of data streams make the task of anonymizing data streams a more challenging issue. Therefore, our work to protect the privacy of streaming data will start with simple privacy model, i.e., k-anonymity, then go on with more sophisticated ones, such as ℓ-diversity and t-closeness.

In this chapter, we will first extend the k-anonymity model to the context of data streams. Then we will present CASTLE (*Continuously k-Anonymizes STreaming data via adaptive cLustEring*), a cluster-based scheme that anonymizes data on-the-fly and, at the same time, ensures the freshness of output data by satisfying specified delay constraints. We further show how CASTLE can be extended to support ℓ-diversity in a straightforward way.

5.1 Introduction

Data streams are common to many application environments, such as, telecommunication, market-basket analysis, network monitoring, and sensor networks. Mining these continuous data streams [36, 56, 85] helps companies (the owner of data streams) to learn the behavior of their customers, thus bringing unique opportunities. Many companies do not have the in-house expertise of data mining, so it is beneficial to outsource the mining to a professional third party [62]. However, data streams may contain much private information that must be carefully protected. Consider Amazon.com. In a single day, it records hundreds of thousands of online sales transactions, which are received in the form of streaming data. Suppose that the sales transaction stream has the schema $\mathcal{S}$(tid, cid, $goods$), where tid is transaction identifier, cid is customer identifier, and $goods$ is a list of items bought by the corresponding customer. Suppose that a relation $\mathcal{R}$ containing the information about Amazon customers is stored on disk, with schema $\mathcal{R}$(cid, $name$, sex, age, $zipcode$, $address$, $telephone$). Let $\mathcal{SR}$ be the stream generated by joining $\mathcal{S}$ with $\mathcal{R}$ on cid. Suppose moreover that, to analyze customers' buying behavior (e.g., building a decision tree), the mining is on $\mathcal{SR}$[1], and Amazon.com outsources it to a professional third-party. To protect the privacy of customers, attributes that explicitly identify customers (such as $name$, $address$ and $telephone$) are projected out of $\mathcal{SR}$. However, the remaining data in $\mathcal{SR}$ may still be vulnerable to *linking attacks*, as illustrated in Figure 5.1, by joining QI attributes (e.g., sex, $zipcode$, and age) with external public databases (e.g., a voter registration table). Therefore, the streaming transactions

[1]In real stream systems, typically customer information does not appear in the stream to reduce redundancy. Mining, which needs customer information, requires joining the data stream with local customer databases. In what follows, we consider mining and anonymization on joint streams.

in $\mathcal{SR}$ need to be carefully anonymized before they are passed to the third-party.

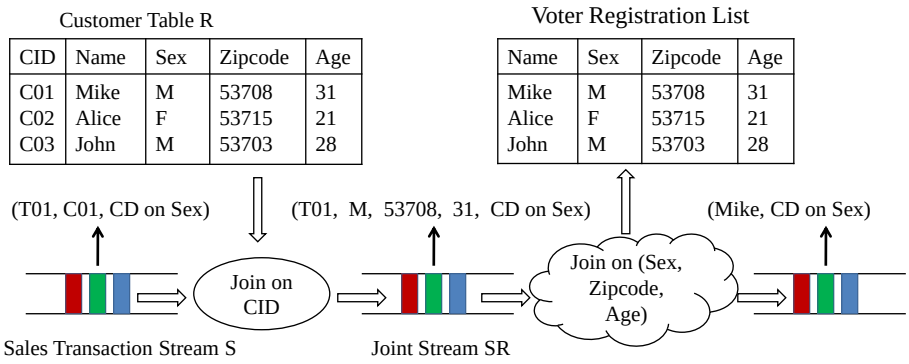


Figure 5.1: Linking attack on transactional data streams

A well-known technique to anonymize data is k-anonymity [67]. However, most of previous k-anonymization algorithms are designed specifically for static data sets. They cannot be directly applied on streaming data for the following reasons. First, these techniques typically assume that an individual only has one record for publishing. Under this assumption, once an equivalence class (EC) contains k tuples, it will hide the released information of one person among a crowd of $k-1$ other persons. However, this assumption is not realistic for streaming data and incurs the following attack. Consider that the QI size is big enough and the QI values of two distinct persons are different with a high probability. Consider again the example of Amazon.com and assume that one customer has purchased $n \geq k$ items. After joining these transactions with customer relation $\mathcal{R}$, n tuples with the same QI value will appear in the resultant stream. According to traditional k-anonymity schemes these n tuples can form an EC and be output immediately without any generalization on their QI values. However, the precise QI value can be linked to a single person in the voting list with a high probability. Thus, the customer is re-identified and his/her privacy is violated.

A second reason that traditional k-anonymity schemes cannot be applied to streaming data is the one-pass requirement imposed on algorithms designed for data streams. Due to the constraints of performance and storage, an algorithm is allowed to make only one-pass over the streaming data (backtracking over data is not allowed). However, traditional anonymization schemes scan a data set multiple times, contrary to the one-pass requirement. Furthermore, data streams have a *temporal dimension*. Streaming tuples arrive at a certain rate, they are dynamically processed, and the result is output with a certain delay. In some applications, the output data are immediately used to trigger appropriate procedures. For example, in a sensor network application the output stream can be used to real-time react to some anomalous situations, thus the time to react is very crucial. Therefore, a data stream anonymization scheme should ensure strong guarantees on the *maximum delay* between the input of data and their output.

Apart from traditional k-anonymity schemes, other alternative techniques, such as *weak k-anonymity* [16], privacy preservation for table republication [23, 61, 73, 81], and data stream clustering [9, 42], cannot be directly applied to anonymize streaming data either. As we will further explain in Section 5.2, these techniques either guarantee weaker privacy than k-anonymity, or target goals very different from that of anonymizing streaming tuples.

To cope with all the above-discussed requirements, in this chapter, we present *CASTLE* (*Continuously Anonymizing STreaming data via adaptive cLustEring*), a cluster-based scheme that k-anonymizes streams on the fly and, at the same time, ensures the freshness (i.e., the maximum delay between the arrival of a tuple and its release to the third party) of anonymized data by satisfying specified delay constraints. Moreover, we propose an extension of CASTLE to support

ℓ-diversity [57] on data streams. To the best of our knowledge, this is the first reported work that considers k-anonymity and ℓ-diversity on data streams.

The basic idea of the proposed approach is to exploit quasi-identifier attributes to define a metric space: tuples are modeled as points in this space. CASTLE groups incoming tuples into clusters and releases all tuples belonging to the same cluster with the same generalization. CASTLE supports the anonymization of both numerical and categorical attributes, by generalizing the latter through domain generalization hierarchies, and the first through intervals. Clustering of tuples is further constrained by the need to have fresh anonymized data. To cope with this requirement, CASTLE ensures that the delay between a tuple's input and its output is at most equal to a given parameter δ. We refer to this constraint as *delay constraint*. When a tuple is going to expire (i.e., its delay is equal to $\delta - 1$), CASTLE immediately releases it. Obviously, it could be the case that an expiring tuple does not belong to a cluster with size at least k. To manage this case, CASTLE implements a merge and split technique to obtain a cluster with size at least k and whose generalization minimizes the information loss. Additionally, to reduce information loss, CASTLE exploits a strategy that allows the reuse of clusters. When a cluster is anonymized and all its tuples have been given in output, CASTLE still keeps it (a.k.a. the corresponding generalization) in memory to anonymize newly arriving tuples, if necessary. However, we found that adopting a naive reuse strategy is flawed *even if it strictly follows the definition of k-anonymity on static data sets*, since it is vulnerable to inference attacks that exploit the sequence of anonymized tuples returned as output. In the chapter, we present the reuse strategy employed by CASTLE to avoid such privacy breaches.

The rest of the chapter is organized as follows. In the next section, we

discuss why the alternative techniques cannot be adopted to anonymize data streams. Section 5.3 proposes our privacy model customized for the anonymization of streaming data. We present a general overview of CASTLE framework in Section 5.4. The detailed algorithms including an extension to support ℓ-diversity, and the security analysis, are presented in Section 5.5. After that, Section 5.6 makes a formal analysis of the time/space complexity of CASTLE. Finally, we report the experimental results on the effectiveness of CASTLE in Section 5.7 and conclude this chapter in Section 5.8.

5.2 Alternative strategies

In this section we investigate related techniques. They are inadequate for data streams, because they either offer weaker protection on data streams than k-anonymity, or target goals different from that of anonymizing data streams.

The first strategy we will study is an extension of *weak k-anonymity* [16]. It achieves stream anonymity by joining data stream $\mathcal{S}$ with a k-anonymized version of customer relation $\mathcal{R}$. The first problem with this solution is that the dimensions to be anonymized (i.e., QI attributes) may come from both $\mathcal{R}$ and $\mathcal{S}$. Thus, anonymizing only $\mathcal{R}$ is inadequate, since the resultant joined stream need be further anonymized. Furthermore, such a solution opens inference channels for an attacker. Weak k-anonymity does not require that each equivalence class has a size of at least k. It only requires that the QI value of each released tuple can be linked to at least k individuals in a public table. It is proven that once k-anonymity is achieved, weak k-anonymity is automatically satisfied. However, the reverse is not true. Therefore, weak k-anonymity is less secure than k-anonymity. Furthermore, weak k-anonymity provides privacy only under the assumption that an adversary does not know a priori whether an individual is in

the released data or not, which appears too strong for most of the application environments related to data streams.

As an example, assume Table 5.1 is a portion of the customer table, in which quasi-identifier attributes are *Sex, Zipcode* and *Age*. Table 5.2 is a 3-anonymized version of Table 5.1, where CID and Name are put outside of the table only for row referencing. Suppose that Mike has bought something from a store and there is an anonymized record in the output stream published by this store, i.e., $\bar{x}$*(M, [53703-53708], [26-31], sex video)*. In addition, no other record with the same generalized QI value is found. Suppose that Beth is an attacker, and that she knows the detailed QI information of Mike. If Beth does not know that Mike appears in the stream (the fact that Mike has made a purchase), by joining $\bar{x}$ with Table 5.2 she will find that $\bar{x}$ is linkable to 3 persons: Mike, John, and Mike. This is *weak 3-anonymity*. The identity of the buyer of *sex video* is hidden. However, once Beth knows that Mike has made a purchase, she is sure that $\bar{x}$ refers to Mike, not to John or Mike, and knows that Mike bought sex video. Mike's privacy is therefore violated.

CID	Name	Sex	Zipcode	Age
C01	Mike	M	53708	31
C02	Alice	F	53715	21
C03	John	M	53703	28
C04	Bob	M	53706	26
C05	Beth	F	53703	24
C06	Carol	F	53706	22

Table 5.1: Customer table

CID	Name	Sex	Zipcode	Age
C01	Mike	M	[53703-53708]	[26-31]
C03	John	M	[53703-53708]	[26-31]
C04	Bob	M	[53703-53708]	[26-31]
C02	Alice	F	[53703-53715]	[21-24]
C05	Beth	F	[53703-53715]	[21-24]
C06	Carol	F	[53703-53715]	[21-24]

Table 5.2: 3-anonymized customer table

In addition, applying the above method requires a further anonymization step if we want to support ℓ-diversity [57] principle. To satisfy this principle, the tuples with the same generalized value should have at least ℓ distinct values for each different sensitive attribute. Since the resultant joined stream only k-anonymizes QI attributes from $\mathcal{R}$, it should be further processed to make the

sensitive attributes from $\mathcal{S}$ ℓ-diverse.

Recently, the problem of anonymizing dynamic datasets, which evolve with tuple insertions and deletions, has started to be investigated [23,61,73,81]. Still, such methods cannot be applied on data streams, since the inferences arising in dynamic datasets are different from those existing in the context of data streams, as the following discussion clarifies. Anonymizing dynamic datasets requires to publish multiple k-anonymized releases of a table. As a consequence, an attacker, by analyzing the multiple generalizations of a same tuple x in the different releases, is able to calculate the intersection of all these generalizations, thus having a better guess of x's real QI value. In some cases, the attacker may even find the exact QI value of x, and link it with a specific victim, as shown in [23, 61, 81]. Anyway, such an inference does not occur when anonymizing data stream, since each streaming tuple is anonymized only once, instead of multiple times. Actually, the possible inferences arising from the anonymization of data stream are due to the fact that an attacker is able to inspect the sequence of anonymized tuples in the output stream (as discussed in Section 5.5.3 below).

Lastly, we discuss the schemes of clustering data streams [9,42]. They focus on finding φ centers in the streaming data so that the sum of distances from data points to their closest centers is minimized. As a consequence, the principle regulating clustering generation is to minimize the distance while the number of total clusters is at most φ. In contrast, in order to k-anonymize streaming data, the proposed approaches have to follow another principle (i.e., the number of tuples in each cluster has to be at least k), which requires to devise new clustering algorithms.

Based on all the above discussion, we strongly believe that a solution for the anonymization of data streams need to be specifically designed.

5.3 The privacy model

We model data stream as an infinite append-only sequence of tuples. Each streaming tuple contains, together with application-specific attributes, also a time stamp that indicates its time of origin. Time stamp is usually modeled as the time when the tuple is generated, or the position of the tuple inside the stream. Without loss of generality, we will consider the tuple position throughout the chapter. Thus, given a tuple x in a stream $\mathcal{S}$, we denote with $x.p$ the attribute of x storing the position of tuple x.

Definition 5.1 (k-anonymity of data streams)**.** *Let $\mathcal{S}$(p, pid, a_1, ..., a_j, A_1, ..., A_d) be a stream, where $\{A_1,\ldots,A_d\}$ is the QI, pid is the person's identity, p is the tuple's position, and $a_1, \ldots, a_j$ are the remaining attributes. Let $\mathcal{S}_{out}$ be the anonymized stream generated from $\mathcal{S}$ where p and pid have been pruned. We say that $\mathcal{S}_{out}$ is k-anonymized, if both the following conditions hold:*

- *For each tuple $x \in \mathcal{S}$, if it is not suppressed, there exists in $\mathcal{S}_{out}$ the corresponding anonymized tuple $\bar{x}$.*
- *Given a tuple $\bar{x} \in \mathcal{S}_{out}$, we define $\mathcal{G}_{\bar{x}} = \{\bar{x}' \in \mathcal{S}_{out} | \bar{x}'.A_i = \bar{x}.A_i, i \in [1, d]\}$ as the EC containing the anonymized tuples with the same generalized QI value as $\bar{x}$. Let $\mathsf{DP}(\mathcal{G}_{\bar{x}})$ be the set of distinct persons which tuples in $\mathcal{G}_{\bar{x}}$ refer to. Then for each possible $\mathcal{G}_{\bar{x}} \subset \mathcal{S}_{out}$, $|\mathsf{DP}(\mathcal{G}_{\bar{x}})| \geq k$.*

Traditional k-anonymity schemes [19, 39, 43, 48, 67] assume that each individual has only one tuple appearing in the data set, so they form ECs, each with at least k tuples only. However, such an assumption does not hold for streaming data as discussed in the introduction. To address this, Definition 5.1 explicitly requires that each EC in the context of data streams refer to at least k distinct persons (Example 5.1 in Section 5.4.1 illustrates a case when $k = 3$).

A relevant feature that closely related to the anonymization of data streams is the *freshness* of the output data. This can be considered as the maximum allowed time of a tuple staying in the memory before it is output, which is formally defined as follows.

Definition 5.2 (Delay constraint)**.** *Let $\mathcal{A}$ be a k-anonymization scheme that takes as input a data stream $\mathcal{S}$ and generates in output a data stream $\mathcal{S}_{out}$, and let δ be a positive integer. We say that $\mathcal{A}$ satisfies the* delay constraint *δ if and only if for each new arriving tuple $x \in \mathcal{S}$ with position $x.p$, all tuples with position less than $x.p - \delta$ have already been output by $\mathcal{A}$.*

According to the above definition, when a new tuple x arrives, the tuple x' with position $x'.p = x.p - \delta$ can still stay in the memory. However, when the next new tuple with position $x.p + 1$ comes, x' should already have been output. Therefore, once x has arrived, x' is *expiring* and needs to be output. Note that the δ parameter can be tuned on the basis of the application domain, the temporal requirements, and the desired information quality. Indeed, when δ increases, the maximum delay between the arrival of a tuple and its release to the third party for data ming is increased. However, this allows CASTLE to buffer more tuples, and the defined metric space (each QI attribute is one dimension) becomes denser. So CASTLE is more likely to group similar tuples together and reduces information loss. Therefore, it is possible to trade off between the allowed delay and the obtained information quality. The experiments in Section 5.7 verify this relationship.

In the introduction, we have discussed the prior knowledge (capability) of an attacker. We formalize it as follows.

Definition 5.3 (Adversary model)**.** *At any instant $\imath$, an attacker's knowledge includes:*

- *The deployed generalization principle (that is, the algorithms according to which tuples are generalized by CASTLE).*
- *The values of all the anonymized tuples released at instant i, and the values of all the others released earlier.*
- *The QI value of any victim, and whether a tuple related to the victim appears in the output stream.*

The above modelling shows the unique characteristics of the anonymization of data streams. Therefore, in the following we denote k-anonymity of data streams by k_s-anonymity to distinguish it from k-anonymity for static data sets. The aim of CASTLE is continuously k_s-anonymizing data streams under the delay constraint, so that the output streams are immune from adversarial attacks, and at the same time, preserving as much information as possible (by GLM in Section 2.3).

5.4 The CASTLE framework

We present the CASTLE framework in this section. Before that, we first introduce the notion of clusters over data streams.

5.4.1 Clusters over data streams

The basic idea of CASTLE is to exploit QI attributes to define a metric space, so that streaming tuples are considered as points in this space. According to this strategy, a cluster can be seen as d-dimensional intervals, where d is the number of QI-attributes. The formal definition of cluster is given below.

Definition 5.4 (Cluster over a data stream)**.** *Let $\mathcal{S}(p, pid, a_1, \ldots, a_j, A_1, \ldots, A_d)$ be a stream where $\{A_1, \ldots, A_d\}$ are the QI attributes. Let $\mathcal{S}' \subset \mathcal{S}$ be a subset of*

tuples. A cluster $\mathcal{C}$ over $\mathcal{S}'$ is defined as a set of intervals, called range intervals*, in the QI attribute domains. For each QI attribute A_i, the corresponding range interval r_i is defined as follows:*

- *If A_i is a continuous attribute with domain $[\mathcal{L}_{A_i}, \mathcal{U}_{A_i}]$, $r_i \subset [\mathcal{L}_{A_i}, \mathcal{U}_{A_i}]$ is the minimal sub-interval that contains all values of A_i in $\mathcal{S}'$.*
- *If A_i is a categorical attribute with domain hierarchy $\mathcal{H}_i$, let $\mathsf{Leaves}(\mathcal{H}_i)$ be the set of leaves of $\mathcal{H}_i$ generated by a left-most traversal of all the leaves in $\mathcal{H}_i$. Let $\mathcal{SL}_{A_i}$ be the smallest subset of $\mathsf{Leaves}(\mathcal{H}_i)$ containing all values of A_i in $\mathcal{S}'$, r_i's bounds are the left-most and the right-most values of $\mathcal{SL}_{A_i}$, respectively.*

Given a cluster $\mathcal{C}$ we denote with $\underline{\mathcal{C}.size}$ the number of distinct persons which tuples in $\mathcal{C}$ refer to. This can be easily calculated by considering the number of distinct values of the pid attribute. Moreover, we denote with $\mathcal{C}.r_i$ the i-th range interval of $\mathcal{C}$, and with $\mathcal{C}(r_1,\ldots,r_n)$ the cluster together with its range intervals.

Example 5.1. *Consider the stream $\mathcal{S}$(p, pid, $a_1, \ldots, a_j$,* Age, Edu*), where $\{$`Age`, `Edu`$\}$ is the QI, and the domain hierarchy $\mathcal{H}_{Edu}$ of categorical attribute* Edu *is shown in Figure 5.2. Assume the following tuples*[2]*: (pid_1,* 25, Bachelor*), (pid_2,* 26, Master*), and (pid_3,* 30, Ph.D*). According to Definition 5.4, cluster $\mathcal{C}$ defined over these three tuples has [*25, 30*] as* Age *range interval and [*Bachelor, Ph.D*] as* Edu *range interval. Therefore, it is denoted as $\mathcal{C}$([*25, 30*], [*Bachelor, Ph.D*])* *with size equal to* 3*. If we further add the tuple (pid_1,* 25, Bachelor*) to $\mathcal{C}$, it does not change its range intervals nor its size (which is still equal to* 3*).*

[2] *For simplicity here and in the following we only consider the pid and QI attributes. Moreover, where not relevant, pid attribute is omitted.*

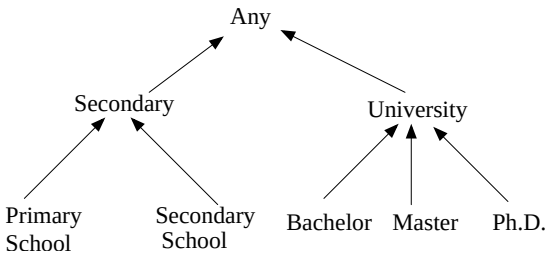


Figure 5.2: Domain generalization hierarchy of education

Given a cluster $\mathcal{C}$ with the size of at least k, if all its tuples are output by sharing a same generalized QI value, then the properties of k_s-anonymity can be satisfied (see Definition 5.1). The following defines the generalization related to such an output.

Definition 5.5 (Cluster generalization)**.** *Let $\mathcal{C}(r_1,\ldots,r_n)$ be a cluster. The corresponding cluster generalization (or simply generalization), denoted as $\mathcal{CG} = (g_1, \ldots, g_n)$, is so defined that, for each $r_i, i \in [1, n]$, g_i is computed as follows:*

- *if r_i is defined on a continuous attribute A_i, $g_i = r_i$;*
- *if r_i is defined on a categorical attribute A_i with domain hierarchy $\mathcal{H}_i$, then $g_i \in \mathcal{H}_i$ is the lowest common ancestor of the bounds in r_i.*

Moreover, we say that a tuple x is output with $\mathcal{C}$'s generalization, if each QI attribute A_i, $i \in [1, n]$, of x is replaced by the corresponding value g_i in the generalization associated with $\mathcal{C}$.

Example 5.2. *Consider the cluster of Example 5.1, that is, $\mathcal{C}$([*25, 30*], [*Bachelor, Ph.D*]). According to the domain generalization hierarchy presented in Figure 5.2, the generalization associated with $\mathcal{C}$ is ([*25, 30*],* University*), since* University *is the lowest common ancestor of* Bachelor *and* Ph.D.

After a cluster reaches the size of k and outputs all its tuples, we may buffer its generalization. Later when a tuple falling in this cluster is expiring, we can

release the tuple by reusing buffered generalization (see Section 5.4.3 below for details). Clearly, this output tuple together with other released tuples by this cluster share a same QI value (i.e., the generalization), and form an EC complying with k_s-anonymity. We call such a reused cluster k_s-anonymized cluster, which is defined as follows.

Definition 5.6 (k_s-anonymized cluster)**.** *Let $\mathcal{C}(r_1,\ldots,r_n)$ be a cluster, and $(g_1,\ldots,g_n)$ be the corresponding generalization. If at a given time instant $\imath$, $\mathcal{C}.size$ is greater than or equal to k and all tuples in $\mathcal{C}$ are output with $\mathcal{C}$'s generalization $(g_1,\ldots,g_n)$, we say that, starting from $\imath$, $\mathcal{C}$ is a k_s-anonymized cluster.*

At the end of this subsection, we define the information loss metrics[3] for the clusters over data stream. Let $\{A_1, A_2, \ldots, A_d\}$ be the QI, and $\mathcal{C}(g_1, g_2, \ldots, g_d)$ be a cluster and its generalization. We define the information loss with respect to QI attribute A_i in cluster $\mathcal{C}$ as $\mathcal{IL}_{A_i}(g_i, \mathcal{C})$. If A_i is numerical with domain range $[\mathcal{L}_{A_i}, \mathcal{U}_{A_i}]$, and g_i is an interval $[l^{\mathcal{C}}_{A_i}, u^{\mathcal{C}}_{A_i}]$, then $\mathcal{IL}_{A_i}(g_i, \mathcal{C}) = \frac{u^{\mathcal{C}}_{A_i} - l^{\mathcal{C}}_{A_i}}{\mathcal{U}_{A_i} - \mathcal{L}_{A_i}}$. If A_i is categorical with the domain hierarchy $\mathcal{H}_{A_i}$, and g_i is a node in $\mathcal{H}_{A_i}$, then

$$\mathcal{IL}_{A_i}(g_i, \mathcal{C}) = \begin{cases} 0, & |\mathsf{Leaves}(g_i)| = 1 \\ \frac{|\mathsf{Leaves}(g_i)|}{|\mathsf{Leaves}(\mathcal{H}_{A_i})|}, & \text{otherwise} \end{cases}$$

where $\mathsf{Leaves}(g_i)$ is the set of leaves under the subtree of $\mathcal{H}_{A_i}$ rooted at g_i, and $\mathsf{Leaves}(\mathcal{H}_{A_i})$ is the whole set of leaves in $\mathcal{H}_{A_i}$. Therefore, the information loss of cluster $\mathcal{C}$ is $\mathcal{IL}(\mathcal{C}) = \frac{1}{d} \times \sum_{i=1}^{d} \mathcal{IL}_{A_i}(g_i, \mathcal{C})$.

5.4.2 Scheme overview

Initially, no clusters are in memory. When CASTLE receives the first tuple, it generates a cluster over it. Then, for every newly arriving tuple x, CASTLE

[3]Once we regard a cluster as an EC, these metrics will be the same as those in Section 2.3.

selects, among all the existing clusters, the one that is *closest* to x. We introduce a concept, *cluster enlargement* (defined as below), to evaluate the distance between a tuple and a cluster. Given the selected cluster $\mathcal{C}$ and the new tuple x, $\mathcal{C}$ may need to enlarge its range intervals to accommodate x. As a consequence, the enlargement implies an increase of information loss. To minimize information loss, when selecting the cluster, CASTLE chooses the one that needs the smallest enlargement.

Definition 5.7 (Cluster enlargement)**.** *Consider a data stream* $\mathcal{S}(p, pid, a_1, \ldots, a_j, A_1, \ldots, A_d)$ *with* $\{A_1, \ldots, A_d\}$ *as QI. Let* $\mathcal{C}$ *be a cluster defined over tuples in* $\mathcal{S}' \subset \mathcal{S}$*, and* $(g_1, g_2, ..., g_n)$ *be its generalization. Let* x *be a tuple in* $\mathcal{S} \setminus \mathcal{S}'$*, and* $\mathcal{C}^{+x}(\tilde{g}_1, \tilde{g}_2, ..., \tilde{g}_n)$ *be another cluster and its generalization defined over tuples in* $\mathcal{S}' \cup \{x\}$*. Then, the enlargement of* $\mathcal{C}$ *with respect to* x *is defined as follows:*

$$\mathsf{Enlargement}(C, x) = \frac{1}{d} \sum_{i=1}^{d} \left(\mathcal{IL}_{A_i}(\tilde{g}_i, \mathcal{C}^{+x}) - \mathcal{IL}_{A_i}(g_i, \mathcal{C})\right)$$

Note that in the case $\mathcal{C}$ contains only one tuple $\bar{x}$, we say that $\mathsf{Enlargement}(\mathcal{C}, x)$ returns the distance between $\bar{x}$ and x.

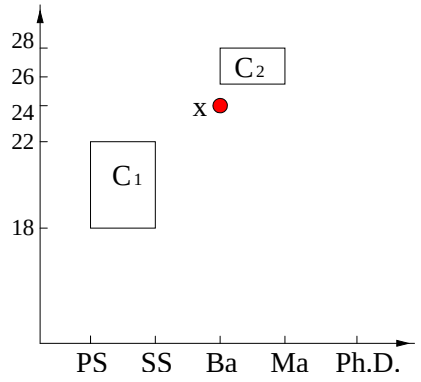


Figure 5.3: Cluster selection

Example 5.3. *Consider clusters* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *in Figure 5.3. To enclose tuple* x*(*24*,* Bachelor*) into cluster* $\mathcal{C}_1$ *its range intervals should be enlarged to [*18*,* 24*] and*

*[*Primary School*,* Bachelor*], respectively, which implies that the new generalization associated with $\mathcal{C}_1$ is ([*18*,* 24*],* Any*). Assume that [*18*,* 120*] is the domain of the* Age *attribute. Thus, the enlargement of $\mathcal{C}_1$ due to absorbing x is* $\mathsf{Enlargement}(\mathcal{C}_1, x) = 1/2 \times (6/102 + 5/5) - 1/2 \times (4/102 + 2/5) = 0.309$. *In contrast, the range intervals of $\mathcal{C}_2$, enlarged to enclose x, are [*24*,* 28*] and [*Bachelor*,* Master*], respectively, which correspond to the new generalization associated with $\mathcal{C}_2$: ([*24*,* 28*],* University*). This implies that the enlargement of $\mathcal{C}_2$ due to absorbing x is* $\mathsf{Enlargement}(\mathcal{C}_2, x) = 1/2 \times (4/102 + 3/5) - 1/2 \times (2/102 + 3/5) = 0.01$. *Thus, x is pushed into $\mathcal{C}_2$.*

However, always pushing a new tuple into an existing cluster may result in clusters with big generalizations, thus implying poor information quality. Therefore, to prevent clusters from becoming too big, if pushing a new tuple to any existing cluster makes the information loss of that cluster greater than a threshold τ, CASTLE generates a new cluster over the new tuple (see Section 5.4.2 for more details).

To satisfy delay constraints, when a new tuple arrives, CASTLE checks whether a tuple in some cluster is going to expire. If this is true, the corresponding tuple must be immediately output. Here, there are two main cases. The first is when cluster $\mathcal{C}$, hosting the expiring tuple, has already a size greater than or equal to k. In this case CASTLE simply outputs all the tuples in $\mathcal{C}$ with its generalization, and starting from the instant of outputting, it considers $\mathcal{C}$ as a k_s-anonymized cluster. The second case is when the cluster $\mathcal{C}$ hosting the expiring tuple has size less than k. To immediately output the expiring tuple, CASTLE merges $\mathcal{C}$ with some of its neighboring clusters such that the size of the resultant cluster is greater than or equal to k. More precisely, CASTLE selects those that result in minimum enlargement to $\mathcal{C}$. Then, all the tuples contained in the new

cluster can be output with its generalization (see Section 5.5 for more details on the merge operation).

In both cases, before outputting the cluster's tuples, CASTLE verifies whether the cluster can be split into smaller sub-clusters. Indeed, according to the adopted information loss metric, the smaller the cluster is (i.e., its range intervals), the smaller its information loss will be. Therefore, if $\mathcal{C}$'s size is at least $2k$, before outputting the tuples, CASTLE splits it into two or more sub-clusters, each with size at least k (see Section 5.5 for more details on the split operation).

5.4.3 Reuse of k_s-anonymized clusters

To increase the information quality of anonymized data, we have enhanced CASTLE with a strategy that enables to *reuse* k_s-anonymized clusters (i.e., their generalizations). According to this approach, after a cluster becomes k_s-anonymized, it is not deleted from memory. Instead, its generalization is kept, and is used later on to output expiring tuples contained in it.

A k_s-anonymized cluster will be no more k_s-anonymized if its range intervals are enlarged due to the insertion of new tuples. Therefore, when a new tuple x arrives, CASTLE selects the cluster to absorb x only among the set of non-k_s-anonymized clusters. This not only avoids the enlargement of k_s-anonymized clusters, but also gives to the non-k_s-anonymized clusters more possibilities to become k_s-anonymized. It is relevant to note that this strategy leads to overlaps between k_s-anonymized and non-k_s-anonymized clusters. The main advantage of overlaps is that the generalizations of k_s-anonymized clusters can be used for anonymizing expiring tuples that have been absorbed by non-k_s-anonymized clusters but are also enclosed into k_s-anonymized clusters. Therefore, every time a tuple x inside a non-k_s-anonymized cluster is going to expire, CASTLE

verifies whether x also falls in a k_s-anonymized cluster $\mathcal{KC}$. If this happens, x is immediately output with $\mathcal{KC}$'s generalization. Thus, the reuse strategy avoids some cluster merges and consequently improves information quality.

Example 5.4. *Let us suppose that, at a given instant, cluster $\mathcal{C}_2$([26, 28], [*Bachelor, Master*]) becomes k_s-anonymized. Moreover, suppose that, after some time, a new cluster $\mathcal{C}_3$([24, 27], [*Bachelor, Master*]) is generated (see Figure 5.4). In case a tuple x(*26, Bachelor*) of $\mathcal{C}_3$ is going to expire and $\mathcal{C}_3$ still has size less than k, x can be given in output with $\mathcal{C}_2$'s generalization, that is, ([*26, 28*],* University*).*

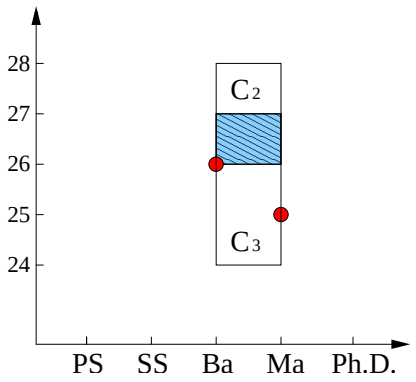


Figure 5.4: Overlapping clusters

When a tuple x is expiring, the best way to preserve information is to select from all the k_s-anonymized clusters the one which contains x and has the minimal information loss. However, this method allows an attacker to infer additional knowledge about a tuple's value or even guess its exact value, as the following example shows.

Example 5.5. *With reference to Figure 5.4, suppose that at a given instant $\imath$ both clusters $\mathcal{C}_2$([26, 28], [*Bachelor, Master*]) and $\mathcal{C}_3$([24, 27], [*Bachelor, Master*]) are k_s-anonymized. Suppose moreover that a tuple $\tilde{x}$ (*25, Master*) arrives after*

*instant $\imath$ and that, after some time, it is expiring. CASTLE outputs this tuple with $\mathcal{C}_3$'s generalization, i.e., ([*24, 27*],* University*). However, by tracing the output stream, an attacker is able to infer that $\tilde{x}.Age$ is not [*26, 27*] (the age overlap between $\mathcal{C}_2$ and $\mathcal{C}_3$), otherwise, x would be in $\mathcal{C}_2$ and would have been output with $\mathcal{C}_2$'s generalization, which preserves more information comparing with $\mathcal{C}_3$'s generalization. Thus, an attacker can infer that $\tilde{x}.Age$ belongs to $\mathcal{C}_3.Age \setminus (C_2.Age \cap C_3.Age) = [24, 25]$.*

To overcome this attack, we employ the following reuse strategy: if an expiring tuple falls into the overlap of two or more k_s-anonymized clusters, CASTLE randomly selects one of them and anonymizes the tuple with its generalization. This avoids the security flaw previously discussed (see Section 5.5.3 for a formal proof).

Example 5.6. *Consider again Example 5.5 and assume that the strategy discussed above is adopted. When an attacker sees the generalization of $\tilde{x}$ (the generalization of $\mathcal{C}_3$), s/he knows that $\tilde{x}$ could be in $\mathcal{C}_2 \cap \mathcal{C}_3$ (the overlap between $\mathcal{C}_2$ and $\mathcal{C}_3$) or $\mathcal{C}_3 \setminus (\mathcal{C}_2 \cap \mathcal{C}_3)$. This inference tells that $\tilde{x}$ could be in any place of $(\mathcal{C}_2 \cap \mathcal{C}_3) \cup (\mathcal{C}_3 \setminus (\mathcal{C}_2 \cap \mathcal{C}_3))$, which is exactly the generalization of $\mathcal{C}_3$.*

5.4.4 Adaptability to data stream distribution

In order to adapt CASTLE to data stream distribution we introduce several strategies. The first is related to the value of τ, i.e., the threshold determining whether an arriving tuple should be pushed to an existing cluster or a new cluster should be created over it. To make CASTLE adaptive to the data stream distribution, we do not consider a predefined and fixed τ. Instead, τ is set to the average information loss of the μ most recent k_s-anonymized clusters (see Section 5.7 for a description on how to set the number μ). Let us see the benefits

of this dynamically calculated τ. When a data stream contains well clustered tuples, it is possible to generate over them small clusters with low information loss. As a consequence, τ will assume a small value, and a new tuple will be pushed into a cluster only if they are very close. This ensures that if tuples in a stream are well clustered, only clusters with small range intervals are formed. On the contrary, if a data stream contains sparsely distributed tuples, clusters with big range intervals are generated. This implies a high value of τ, and also allows an increased enlargement. Thus, a new tuple is more likely to be pushed into an existing cluster. Accordingly, each cluster has improved possibility of reaching the size of k, and the number of cluster merging operations is reduced.

Another adaptivity is obtained by constraining the maximum number of non-k_s-anonymized clusters that stay in memory, delimited by a parameter η. This parameter co-operates with τ to handle the variation of data distribution. More precisely, suppose that, at the beginning all tuples are well clustered, so τ has a small value. If the newly arriving tuples are sparsely distributed, many small-size clusters will be formed since the small τ only allows a small cluster enlargement. The large number of clusters increases the overhead of searching for the best cluster into which a new tuple should be pushed, and it also increases the overhead of merging clusters when tuples expire. The introduction of η prevents the generation of a possibly large number of clusters and thus limit the overhead discussed above. The value of η can be set regarding the available computational and storage resources. Hence, a new cluster is generated only if the number of non-k_s-anonymized clusters is fewer than η. Otherwise, a tuple is pushed to the existing cluster, which still requires the minimum enlargement to enclose it.

5.5 CASTLE algorithms and security analysis

In the following, we present the algorithms implementing the techniques illustrated in the previous section. Then, we show their extension to achieve ℓ-diversity. Moreover, we analyze their security.

5.5.1 Algorithms

The main algorithm of our work is CASTLE, which continuously processes the incoming data stream by producing in output a flow of k_s-anonymized tuples.

Algorithm: CASTLE ($\mathcal{S}$, k, δ, η)

```
1  Let Γ be the set of non-k_s-anonymized clusters, initialized to be empty;
2  Let Ω be the list of k_s-anonymized clusters, set to be empty;
3  τ = 0;
4  while new tuples still come from S do
5      Get the arriving tuple x from S;
6      C = bestSelection(x);
7      if C = NULL then
8          Create a new cluster over x and insert the cluster into Γ;
9      else
10         Push x into C;
11     Suppose that x' is the tuple so that x'.p = x.p − δ;
12     if x' has not yet been output then
13         delayConstraint(x');
```

Algorithm CASTLE takes as input the original data stream $\mathcal{S}$ together with other parameters: k, δ and η. At the beginning, the set of non-k_s-anonymized clusters (i.e., Γ) as well as the set of k_s-anonymized clusters (i.e., Ω) are empty. Then, every time a tuple x arrives (step 5), it calls function bestSelection (step 6), to select from Γ the best cluster into which x is pushed (step 10). If such a cluster does not exist, a new cluster over x is created (steps 7-8). In addition, CASTLE verifies whether the arrival of the new tuple x forces another tuple x' with position $x.p - \delta$ to expire (step 11). If this is the case, procedure delayConstraint will be called (step 13). In the following, we illustrate the

procedures/functions used by Algorithm CASTLE.

Function bestSelection (x)

1 Initialize E to be an empty set;
2 **foreach** $\mathcal{C} \in \Gamma$ **do**
3 Calculate $e = \mathsf{Enlargement}(\mathcal{C}, x)$;
4 Insert $(e, \mathcal{C})$ into E;
5 $e_m = \min_{\forall e}\{e \mid (e, \mathcal{C}) \in E\}$;
6 $S_{e_m} = \{\widehat{\mathcal{C}} \mid \widehat{\mathcal{C}} \in \Gamma \; and \; \mathsf{Enlargement}(\widehat{\mathcal{C}}, x) = e_m\}$;
7 **foreach** $\widehat{\mathcal{C}} \in S_{e_m}$ **do**
8 Compute $\mathcal{IL}(\widehat{\mathcal{C}} \cup \{x\})$, i.e., the information loss of $\widehat{\mathcal{C}}$ after absorbing x;
9 **if** $\mathcal{IL}(\widehat{\mathcal{C}} \cup \{x\}) \leq \tau$ **then**
10 Insert $\widehat{\mathcal{C}}$ into SC_x;
11 **if** SC_x *is empty* **then**
12 **if** $|\Gamma| \geq \eta$ **then**
13 Return any cluster in S_{e_m} with minimum size;
14 **else**
15 Return NULL;
16 **else**
17 Return any cluster in SC_x with minimum size;

Function bestSelection. According to the reuse strategy, a new tuple is always pushed into a close non-k_s-anonymized cluster. bestSelection first calculates the set of non-k_s-anonymized clusters S_{e_m}, which requires the minimum enlargement to absorb incoming tuple x (steps 1-6). To be adaptive to the data stream distribution described in Section 5.4.4, we refine S_{e_m}, thus only clusters whose information loss after absorbing x will not exceed τ are kept (steps 7-10). These remaining clusters are stored in SC_x (step 10). If SC_x is not empty (the evaluation of step 11 is false), bestSelection returns the cluster with the minimum size (step 17). Otherwise, it implies that a new cluster should be created over x. Still, to avoid a large number of non-k_s-anonymized clusters, thus better adaptive to data stream distribution, the function verifies whether the constraint on the maximum number of non-k_s-anonymized clusters is satisfied. When the number of non-k_s-anonymized clusters is greater than or equal to η (step 12), it is impossible to create a new cluster. In this case, among clusters

requiring the minimum enlargement, i.e., S_{e_m}, the one with the minimum size is returned (step 13). Otherwise, the function returns a NULL value (step 15), which triggers in Algorithm CASTLE the generation of a new cluster over x.

Procedure delayConstraint (x)

```
1  Find C the non-k_s-anonymized cluster holding the expiring tuple x;
2  if C.size ≥ k then
3      outputCluster(C);
4  else
5      Let S_KC be the set of k_s-anonymized clusters in Ω containing x;
6      if |S_KC| > 0 then
7          Randomly select a cluster Ĉ from S_KC;
8          Output x with the generalization of Ĉ;
9          Return;
10     if Σ_{i=1}^{|Γ|} C_i.size < k then
11         Suppress x;
12         Return;
13     MC = mergeClusters(C, Γ \ C);
14     outputCluster(MC);
```

Procedure delayConstraint. When a tuple x is expiring, delayConstraint takes the responsibility of outputting it. The anonymization of x is achieved case by case. First, delayConstraint checks the size of the non-k_s-anonymized cluster $\mathcal{C}$, which contains x. If $\mathcal{C}$ reaches the size of k, x together with other tuples in $\mathcal{C}$ are output by $\mathcal{C}$'s generalization (steps 1-3). Otherwise, delayConstraint verifies whether it is possible to apply on the expiring tuple the reuse strategy illustrated in Section 5.4.3 (steps 5-9). This is possible, if there exist one or more k_s-anonymized clusters, i.e., $S_{\mathcal{KC}}$, containing x. In addition, to overcome the inference problem described in Section 5.4.3, a cluster is randomly selected from $S_{\mathcal{KC}}$, and x is output by its generalization (steps 7-8). As the last alternative, procedure delayConstraint examines whether a merge of $\mathcal{C}$ with some other non-k_s-anonymized clusters is possible. Note that, if the total size of all clusters in Γ is fewer than k (step 10), a merge operation would not produce a cluster with the size at least k. Therefore, the only way to output the expiring tuple

is suppressing it (step 11). Otherwise, the merge takes place (step 13), and the resultant cluster is output (step 14).

The mergeClusters function receives as input the cluster to be merged, i.e., $\mathcal{C}$, and the set of non-k_s-anonymized clusters excluding $\mathcal{C}$ itself. For every non-k_s-anonymized cluster $\mathcal{C}_i$, it calculates the enlargement of $\mathcal{C}$ due to the potential merge with $\mathcal{C}_i$. Then, it selects the one, which incurs the minimum enlargement to $\mathcal{C}$, and merges $\mathcal{C}$ with it. This process continues until $\mathcal{C}$'s size is at least k. Finally, the resultant cluster is returned.

Procedure outputCluster ($\mathcal{C}$)

```
1  if C.size ≥ 2k then
2      S_c = split^k(C);
3  else
4      S_c = {C};
5  foreach C_i ∈ S_c do
6      Output all tuples in C_i with its generalization;
7      Update τ according to IL(C_i);
8      if IL(C_i) < τ then
9          Insert C_i into Ω;
10     else
11         Delete C_i;
12     Delete C_i from Γ;
```

Procedure outputCluster. At the moment of outputting, a cluster with size no less than $2k$ will be split into sub-clusters, each with size at least k (steps 1-2). Then all the tuples staying in a cluster are output by the corresponding cluster generalization (steps 5-6). Meanwhile, τ is updated to be the average information loss of the most recent k_s-anonymized clusters including the new ones (step 7). Furthermore, we insert the newly created k_s-anonymized clusters into Ω for later reuse. To minimize information loss, only those with good information quality are preserved (steps 8-11).

Function split^k. CASTLE employs *KNN search* [10] to split a cluster $\mathcal{C}$ of size at least $2k$. It works as follows. First, a tuple x is randomly selected from

$\mathcal{C}$ and a sub-cluster $\mathcal{C}_{new}$ is created over it. Then, x's k-1 nearest neighbors in $\mathcal{C}$ (x itself excluded) are inserted into $\mathcal{C}_{new}$. To comply with the constraints of k_s-anonymity in Definition 5.1, no pair of tuples in $\mathcal{C}_{new}$ share a pid value. x together with the selected neighbors are deleted from $\mathcal{C}$. This process repeats, each time generating a new sub-cluster, until the size of $\mathcal{C}$ is less than k. Finally, the remaining tuples in $\mathcal{C}$ are pushed into the above newly created sub-clusters.

Function split$^{\mathsf{k}}$ ($\mathcal{C}$)

1 Initialize S_c to be an empty set;
2 Let φ be the set of buckets created by grouping tuples in $\mathcal{C}$ by pid attribute;
3 **while** $|\varphi| \geq k$ **do**
4 Randomly select a bucket $\mathcal{B} \in \varphi$, and pick one of its tuples x;
5 Delete x from $\mathcal{B}$ and create a new subcluster $\mathcal{C}_{new}$ over x;
6 **if** $\mathcal{B}$ *is empty* **then**
7 delete $\mathcal{B}$ from φ;
8 Let H_{k-1} be a heap with $k-1$ nodes, each set to be infinitely faraway from x;
9 **foreach** *bucket in* $\varphi \setminus \mathcal{B}$ **do**
10 Pick one of its tuples $\hat{x}$, and calculate the distance between $\hat{x}$ and x;
11 **if** $\hat{x}$ *is closer to* x *than the root of* H_{k-1} **then**
12 $\hat{x}$ replaces the root, and H_{k-1} is adjusted accordingly;
13 **foreach** *node in* H_{k-1} **do**
14 Assume that x' is the tuple in the node;
15 Insert x' into $\mathcal{C}_{new}$;
16 Find $\mathcal{B}'$ the bucket containing x';
17 Delete x' from $\mathcal{B}'$;
18 **if** $\mathcal{B}'$ *is empty* **then**
19 delete $\mathcal{B}'$ from φ;
20 Add $\mathcal{C}_{new}$ to S_c;
21 **foreach** *bucket* $\tilde{\mathcal{B}} \in \varphi$ **do**
22 Pick a tuple $\tilde{x} \in \tilde{\mathcal{B}}$;
23 Find $\tilde{x}$'s nearest cluster in S_c, and add all the tuples from $\tilde{\mathcal{B}}$ to the cluster;
24 Delete $\tilde{\mathcal{B}}$;
25 return S_c;

Let us see in more details how split$^{\mathsf{k}}$ works. First, the function groups all tuples in $\mathcal{C}$ by their pid values, creating a set of buckets φ (step 2), each one containing only tuples with the same value for attribute pid. Then, it randomly selects a tuple x and creates a new cluster $\mathcal{C}_{new}$ over it (steps 4-5). To speed up the process of finding the k-1 nearest neighbors of x, we adopt a heap with

k-1 nodes (step 8). From each bucket, we select a representative tuple and calculate its distance from x (step 10). Note that all the tuples in a bucket have the same distance from x, because they all share a single QI value. The heap is so maintained that only the k-1 nearest representatives are stored in it (steps 11-12). All the nodes in the heap are pushed to the new sub-cluster $\mathcal{C}_{new}$ (steps 13-19). Since each node in the heap is from a different bucket, it is sure that the k tuples in $\mathcal{C}_{new}$ refer to k distinct persons. Thus, $\mathcal{C}_{new}$ strictly follows k_s-anonymity. All the tuples in the new sub-cluster are deleted from the buckets they hail from (steps 5, 17). The above process continues to create new sub-clusters, until the number of buckets is less than k (steps 3-19). Finally, each of the remaining tuples is added to its nearest sub-cluster created above (steps 21-24).

5.5.2 Extension to ℓ-diversity

Recently, Machanavajjhala et al. [57] proposed the ℓ-diversity principle to provide defence against the attacks based on the distribution of non-QI sensitive attribute ($\mathcal{SA}$). In this section we illustrate how CASTLE can be straightforwardly extended to support ℓ-diversity on data streams. ℓ-diversity postulates that each EC contain at least ℓ "well represented" $\mathcal{SA}$ values. It can be instantiated in multiple ways, such as *distinct ℓ-diversity*, *entropy ℓ-diversity*, and *recursive (c, ℓ)-diversity*. Here we consider distinct ℓ-diversity, which requires that each EC contains ℓ distinct $\mathcal{SA}$ values. Given a cluster $\mathcal{C}$, we denote with $\mathcal{C}.diversity$ the number of distinct $\mathcal{SA}$ values for tuples in $\mathcal{C}$. In order to ensure that the output data by CASTLE comply with also ℓ-diversity besides k_s-anonymity, we slightly modify the definition of k_s-anonymized cluster as follows.

Definition 5.8 (k_s-anonymized and ℓ-diversified cluster)**.** *Let $\mathcal{C}(r_1,\ldots,r_n)$ be a*

cluster, and (g_1,...,g_n) be the corresponding generalization. If, at a given time instant $\imath$: (1) $\mathcal{C}.size$ *is greater than or equal to* k, (2) $\mathcal{C}.diversity$ *is greater than or equal to* ℓ, *and* (3) *all tuples in* $\mathcal{C}$ *are output with* $\mathcal{C}$*'s generalization (g_1,...,g_n), we say that, starting from* $\imath$, $\mathcal{C}$ *is a* k_s*-anonymized and* ℓ*-diversified cluster.*

After the alteration of the k_s-anonymized cluster, accordingly we need to adjust the algorithms of CASTLE. The overall CASTLE framework is the same; it is only necessary to customize algorithms delayConstraint and split$^{\mathsf{k}}$ according to the extra requirements enforced by ℓ-diversity. Now, Procedure delayConstraint outputs a whole non-k_s-anonymized cluster $\mathcal{C}$ (line 2 of delayConstraint), only if the following two conditions hold at the same time: 1) $\mathcal{C}.size \geq k$, and 2) $\mathcal{C}.diversity \geq \ell$. Moreover, it gives up the merge operation and chooses to suppress a tuple (lines 10-11 of delayConstraint), if $\sum_{i=1}^{\Gamma} C_i.size < k$ or there does not exist at least ℓ distinct $\mathcal{SA}$ values among all clusters in Γ. Function split$^{\mathsf{k}}$ conforms to the cluster size constraint only. In the following, we redefine it so that it will also follow the diversity requirement.

Function split$^{\ell}$. The basic idea of splitting a cluster $\mathcal{C}$ works as follows. It first partitions the set of tuples from $\mathcal{C}$ into buckets by grouping by their $\mathcal{SA}$ values. Then, it selects a portion of tuples from each bucket, altogether k tuples, and creates a sub-cluster $\mathcal{C}_{sub}$ over them. Since each bucket contributes tuples to C_{sub} and the number of buckets is at least ℓ, the constraint of ℓ is satisfied automatically. However, one person may have multiple tuples in the stream, so the buckets grouped by $\mathcal{SA}$ may overlap on pid values. Likewise, the set of k tuples selected from the buckets may also overlap on pid, thus they refer to less than k distinct persons and violate the property of k-anonymity. Therefore, to assist the process of selecting k tuples each with a different pid value, we

consider buckets without overlap on pid and turn to the following strategy. It is done by selecting only one tuple from $\mathcal{C}$ for each distinct pid value, and grouping only these selected tuples into buckets according to their $\mathcal{SA}$ values. Avoiding overlaps ensures that the k tuples selected from buckets have different pids, thus ensuring the size of the new generated subcluster $\mathcal{C}_{sub}$ to be k exactly. After creating all the possible sub-clusters, each tuple x previously not selected from $\mathcal{C}$ is inserted into one of the new sub-clusters. In particular, tuple x is inserted into the unique sub-cluster that contains a tuple with the same pid of x.

Function split$^{\ell}$ ($\mathcal{C}$, $\mathcal{SA}$)

1 Let φ be the set of disjoint buckets generated by generateBuckets$(C, \mathcal{SA})$;
2 Initialize S_c to be an empty set;
3 **while** $|\varphi| \geq \ell$ *and* $sum = \sum_{\mathcal{B}_i \in \varphi} \mathcal{B}_i.size \geq k$ **do**
4 Select a tuple x from a randomly selected bucket $\mathcal{B} \in \varphi$;
5 Delete x from $\mathcal{B}$ and generate a subcluster $\mathcal{C}_{sub}$ over it;
6 **foreach** *bucket* $\widehat{\mathcal{B}} \in \varphi$ **do**
7 **foreach** *tuple* $\hat{x} \in \widehat{\mathcal{B}}$ **do**
8 Calculate $\hat{e} =$ Enlargement$(C_{sub}, \hat{x})$;
9 Sort tuples in $\widehat{\mathcal{B}}$ by the ascending order of their enlargement $\hat{e}$;
10 Assume that $\widehat{T}$ is the set of the first $k \times \frac{\widehat{\mathcal{B}}.size}{sum}$ tuples in $\widehat{\mathcal{B}}$;
11 Insert $\widehat{T}$ into $\mathcal{C}_{sub}$ and delete it from $\widehat{\mathcal{B}}$;
12 **if** $\widehat{\mathcal{B}}.size = 0$ **then**
13 Delete $\widehat{\mathcal{B}}$ from φ;
14 Add $\mathcal{C}_{sub}$ to S_c;
15 **foreach** $\mathcal{B}' \in \varphi$ **do**
16 **foreach** *tuple* $x' \in \mathcal{B}'$ **do**
17 Find the nearest subcluster $\mathcal{C}_{near} \in S_c$ to x';
18 Insert x' into $\mathcal{C}_{near}$;
19 Delete $\mathcal{B}'$;
20 **foreach** *subcluster* $\mathcal{C}_{sub} \in S_c$ **do**
21 **foreach** *tuple* $\tilde{x} \in C_{sub}$ **do**
22 Let $E_{\tilde{x}}$ be the set of tuples in $\mathcal{C}$, such that $E_{\tilde{x}} = \{x \in \mathcal{C} | x.pid = \tilde{x}.pid\}$;
23 Insert $E_{\tilde{x}}$ into $\mathcal{C}_{sub}$;
24 Return S_c;

Let us see now in more details how split$^{\ell}$ works. In the first step, it calls Function generateBuckets to create the set of buckets φ without overlaps. Once the number of buckets is greater than ℓ and the number of tuples in φ is greater

than k, a new k_s-anonymized and ℓ-diversified subcluster $\mathcal{C}_{sub}$ is formed (lines 3-13). In particular, the function selects, from each bucket $\widehat{\mathcal{B}} \in \varphi$, a subset of tuples $\widehat{T}$ proportional to the bucket size (lines 6-11). To reduce the information loss, $\widehat{T}$ are only tuples from $\widehat{\mathcal{B}}$ which require the minimum enlargement to $\mathcal{C}_{sub}$ (lines 7-10). These tuples are then inserted into $\mathcal{C}_{sub}$ (line 11). Finally, when one or both the conditions of the while loop are no more satisfied (the evaluation of line 3 is false), remaining tuples are accommodated into the new created subclusters (lines 15-23).

5.5.3 Formal results

In this sub-section, we analyze the security of CASTLE. We start by proving that CASTLE generates k_s-anonymized data streams.

Theorem 5.1. *Let $\mathcal{S}$(p, pid, a_1, ..., a_j, A_1, ..., A_d) be the input stream with $\{A_1, \ldots, A_d\}$ as QI, and $\mathcal{S}_{out}$ be the stream output by CASTLE. Then $\mathcal{S}_{out}$ is k_s-anonymized.*

Proof. According to the definition of k_s-anonymity (Definition 5.1), the theorem holds if the following two conditions are satisfied:

1. For each tuple x in the input stream $\mathcal{S}$, if it is not suppressed, there exists in output stream $\mathcal{S}_{out}$ the corresponding anonymized tuple $\bar{x}$.

2. For any possible EC $\mathcal{G}_{\bar{x}} \subset \mathcal{S}_{out}$ that contains all the tuples with the same generalized QI value as $\bar{x}$, the number of distinct individuals which tuples in $\mathcal{G}_{\bar{x}}$ refer to is greater than or equal to k, that is, $|\mathsf{DP}(\mathcal{G}_{\bar{x}})| \geq k$.

Let us first consider point 1. When tuple x is expiring, it is passed to procedure delayConstraint. We prove that this procedure always outputs an anonymized tuple corresponding to x, if x is not suppressed. Different cases

may arise. The first is when the non-k_s-anonymized cluster hosting x has the size greater than or equal to k. In this case, the procedure calls outputCluster (step 3 of delayConstraint), which gives in output a generalized tuple for x, thus the thesis holds. If the non-k_s-anonymized cluster hosting x has size less than k, procedure delayConstraint verifies whether the reuse strategy can be applied. If this is the case, x is given in output with the generalization of one of the k_s-anonymized clusters enclosing it, thus the thesis also holds (step 8 of delayConstraint). Finally, delayConstraint tests if it is possible to merge clusters. If this is impossible, x is suppressed (step 11 of delayConstraint). Otherwise, all the tuples in the cluster resulting from the merge operation are given in output (step 14 of delayConstraint). In both the cases, the thesis holds.

Let us consider now point 2. To prove it, we discuss all the three possible ways of outputting a tuple one by one. *Case 1:* Tuple x is contained in a cluster $\mathcal{C}$ with the size at least k; x is output together with all the other tuples in $\mathcal{C}$ by the generalization of $\mathcal{C}$. Obviously, the output tuples from $\mathcal{C}$ belong to a same EC $\mathcal{G}_{\bar{x}}$. Since $\mathcal{C}$ has the size at least k, the number of distinct individuals which tuples in $\mathcal{G}_{\bar{x}}$ refer to is greater or equal to k, that is, $|\mathsf{DP}(\mathcal{G}_{\bar{x}})| \geq k$. *Case 2:* Tuple x is output individually by the reuse strategy. Let $\mathcal{KC}$ be the k_s-anonymized cluster selected by the reuse strategy to output x. Since $\mathcal{KC}$ is k_s-anonymized, at some instant before the output of x, $\mathcal{KC}$ must have the size at least k and all its tuples are output with its generalization. Let $\mathcal{G}_{\bar{x}}$ be the EC that contains all the output tuples from $\mathcal{KC}$. Then the anonymized tuple $\bar{x}$ of tuple x belongs to $\mathcal{G}_{\bar{x}}$, and $|\mathsf{DP}(\mathcal{G}_{\bar{x}})| \geq k$. *Case 3:* Furthermore, a tuple can be also suppressed. In this case, the k$_s$-anonymity definition does not require that $|\mathsf{DP}(\mathcal{G}_{\bar{x}})| \geq k$. □

To show that CASTLE generates k_s-anonymized data streams is not enough. As shown by Example 5.5, reuse of k_s-anonymized clusters should be carefully

managed, otherwise, a potential attacker may infer additional knowledge about a tuple's value, or even guess its exact value. We have therefore to prove that the reuse strategy adopted by CASTLE is secure. Before the formal proof, we first introduce some notions, and investigate the underlying factors that enable this inference attack.

Given a cluster $\mathcal{C}$ and its generalization $(g_1, \ldots, g_d)$, the generalization g_j of the QI attribute A_j shows the range in which the real values of A_j for tuples in $\mathcal{C}$ falls. For instance, given QI attribute *Edu* and its domain hierarchy shown in Figure 5.2, if the generalized value for Edu is *University*, we can infer that the corresponding original tuple can have as value for this attribute one of the elements in the set {Bachelor,Master, Ph.D.}. Similarly, if the value given in output by CASTLE for QI attribute *Age* is [25-30], then the *Age* attribute of the corresponding non-anonymized tuple has a value in the interval [25,30]. In what follows, given a cluster and its generalization $\mathcal{C}(g_1, \ldots, g_d)$, we denote with $values(\mathcal{C}.g_i)$ the set of values implied by the generalization g_i for attribute A_i. More precisely, if A_i is a continuous attribute, $values(\mathcal{C}.g_i)$ contains all the values in the interval corresponding to $\mathcal{C}.g_i$. In contrast, if A_i is a categorical attribute, $values(\mathcal{C}.g_i)$ contains all the leaves of the subtree in $\mathcal{H}_i$ rooted at $\mathcal{C}.g_i$, where $\mathcal{H}_i$ denotes the domain hierarchy for attribute A_i.

Now let us consider Example 5.5 once more. Without a careful reuse management, given a tuple $\bar{x}$ output by cluster generalization $\mathcal{C}(g_1, \ldots, g_d)$, an attacker may, by tracing the sequence of anonymized tuples, may infer that the possible real values of $\bar{x}$ on attribute A_i ($i = 1, \ldots, d$) stay in a subset of $values(\mathcal{C}.g_i)$, instead of $values(\mathcal{C}.g_i)$ itself. Such an inference enables an attacker to link tuple $\bar{x}$ to less than k individuals, thus violating k-anonymity. The next example better clarifies this.

Example 5.7. *Recall Example 5.5. Suppose that $k = 4$ and QI is* {`Age, Edu`}. *Assume* 4 *persons* P_1, P_2, P_3, *and* P_4, *with QI values (*24, Bachelor*), (*25, Master*), (*26, Bachelor*), and (*27, Master*), respectively. Suppose that at instant* $\imath$, *each of the 4 persons has at least one tuple in* $\mathcal{C}_3$, *and* $\mathcal{C}_3$ *becomes* k_s*-anonymized after outputting all its tuples by its generalization. Moreover, suppose that after some time,* $\tilde{x}$*=(*25, Master, Video on sex*) is expiring, and is output by reusing* $\mathcal{C}_3$*'s generalization ([*24, 27*],* University*). As shown in Example 5.5, if the reuse strategy always tries to minimize information loss, then an attacker can learn that the original values of* $\tilde{x}$ *on* Age *is [*24, 25*], instead of* $\mathcal{C}_3.Age = [24, 27]$. *By linking ([*24, 25*],* University*) to the QI values of the discussed 4 persons, we know only* P_1 *or* P_2 *(excluding* P_3 *and* P_4*) can be the owner of* $\tilde{x}$. *Consequently, the privacy of* P_1 *and* P_2 *is weaken.*

To avoid this possible inference attack, CASTLE adopts the cluster reuse strategy introduced in Section 5.4.3. As the following theorem states, this strategy ensures that under the adversary model presented in Definition 5.3 an attacker cannot infer any additional knowledge on a tuple x other than the set of real values associated with the generalization according to which it has been given in output.

Theorem 5.2. *Let* $\mathcal{S}$*(*p, pid, a_1, …, a_j, A_1, …, A_d*) be a stream with* {A_1, …, A_d} *as QI, and* $\mathcal{S}_{out}$ *be the stream output by CASTLE. Suppose that* $\mathcal{C}_1, \ldots, \mathcal{C}_m$ *is the sequence of* k_s*-anonymized clusters appearing in* $\mathcal{S}_{out}$, *and that* x *is a tuple output by the cluster generalization of* $\mathcal{C}_r$, $1 \leq r \leq m$. *Then, for each QI attribute* A_i, $i = 1, \ldots, d$, *an attacker is unable to infer that the real value of* $x.A_i$ *belongs to a subset of* $values(\mathcal{C}_r.A_i)$.

Proof. For the simplicity of discussion, let us consider a data stream $\mathcal{S}$ with a single QI attribute q. Consider a tuple x k_s-anonymized with generalization $\mathcal{C}_r$.

In this case, an attacker could infer a set of possible values $\chi \subset values(\mathcal{C}_r.q)$ for attribute q, only if there exists a set of tuples $T \subseteq \mathcal{S}_{out} \setminus \{x\}$ k_s-anonymized with a generalization $\mathcal{C}_s$ ($1 \leq s \leq m$) that *overlaps* $\mathcal{C}_r$. Two generalizations $\mathcal{C}_r$ and $\mathcal{C}_s$ overlap if $values(\mathcal{C}_r.q) \cap values(\mathcal{C}_s.q) \neq \emptyset$. Let us assume that an attacker infers a set of possible values $\chi \subset values(\mathcal{C}_r.q)$ for attribute q, and prove that a contradiction arises. We first consider the case that q is a continuous attribute. χ can be inferred by analyzing the overlap between $\mathcal{C}_r$ and $\mathcal{C}_s$, and can be formalized as follows: χ= $values(\mathcal{C}_r.q) \setminus values(\mathcal{C}_s.q)$. Thus, χ consists of the values in $values(\mathcal{C}_r.q)$ which are not in $values(\mathcal{C}_s.q)$. However, if there exists an overlap between $values(\mathcal{C}_r.q)$ and $values(\mathcal{C}_s.q)$, it implies that CASTLE anonymizes $x.q$ by randomly selecting between $\mathcal{C}_r.q$ and $\mathcal{C}_s.q$ (see line 7, procedure delayConstraint). Hence, the possible real values of $x.q$ that can be inferred are, in addition to those in χ, also those in $values(\mathcal{C}_r.q) \cap values(\mathcal{C}_s.q)$. Therefore, χ= $(values(\mathcal{C}_r.q) \setminus values(\mathcal{C}_s.q)) \cup (values(\mathcal{C}_r.q) \cap values(\mathcal{C}_s.q))$, that is, χ = $values(\mathcal{C}_r.q)$, and a contradiction arises. We omit the proof when q is a categorical attribute, since it is very similar to the proof given for continuous attributes. □

It is important to note that above theorem holds only under the setting (Definition 5.3), where the adversary does not have any extra background knowledge (e.g., temporal background knowledge: the attacker may know that Mike buys an item at time 10). Considering further background knowledge is a challenging issue, that we plan to investigate in the future, and that may require the definition of alternative techniques to achieve k_s-anonymity.

Furthermore, CASTLE may output some tuples before their expiration. Consequenlty, the ordering of the output stream may be different from that of the input. Formally, we say that input stream $\mathcal{S}$ and output stream $\mathcal{S}_{out}$ share the same

ordering, if the following condition holds: Given any two tuples $x_1, x_2 \in \mathcal{S}$, and their corresponding output versions $\bar{x}_1, \bar{x}_2 \in \mathcal{S}_{out}$, if $x_1.p < x_2.p$, then $\bar{x}_1$ is output earlier than $\bar{x}_2$. Some data mining applications may be sensitive to the tuple ordering. If this is the case, we reorder output streams with regard to input streams, by releasing a tuple $\bar{x}$ anonymized from x only after all the tuples with position less than $x.p$ have already been output. The reordering strategy ensures the same ordering, but also incurs extra time delay. Now the total buffer time of a tuple x before its output is composed of two parts: 1) the time interval between its arrival and its anonymization, and 2) the buffer time for reordering. Anyway, the next theorem proves that even with the reordering the total buffer time is still upper bounded by δ, hence the *delay constraint* (Definition 5.2) still holds.

Theorem 5.3. *Let $\mathcal{S}$ be an input stream, and $\mathcal{S}_{out}$ be its output stream generated and reordered by CASTLE. Assume that δ is an integer representing the maximum allowed delay between a tuple's input and its output. Then, the total buffer time of any tuple $x \in \mathcal{S}$ before its output is upper bounded by δ, and the delay constraint is still guaranteed.*

Proof. Let x and $\bar{x}$ be an input tuple and its anonymized one, respectively. When tuple x_1 with position $x_1.p = x.p+\delta$ comes, all tuples (denote them by ET) with position less than $x.p = x_1.p - \delta$ have already expired and been anonymized. At the instant of $x_1.p$, x is expiring and is already anonymized to $\bar{x}$ (x can be anonymized at any time in [$x.p$, $x_1.p$] and buffered). We release $\bar{x}$ following the output of all the tuples in ET. According to the above buffer and release procedure, it is obvious that for any tuple $x' \in \mathcal{S}$ with the position $x'.p < x.p$, x' is output earlier than x. Therefore, the input and output streams have the same ordering. Furthermore, when x_1 comes, x and all the tuples with position less

than $x.p = x_1.p - \delta$ are output, so the total buffering time of x is upper bounded by δ, and *delay constraint* is satisfied. □

Note that, if an attacker has the knowledge of the ordering of the input stream, the reordering will enable him/her to link released tuples to their owners. For the simplicity of discussion, we assume that tuple positions start from 1 and are consecutive. Suppose that Mike buys an item. Let x be the record for Mike's purchase. If an attacker knows the ordering of input tuples, that is to say, s/he knows that Mike buys the item at time $x.p$, then s/he can infer that $x.p$-th tuple in the output stream belongs to Mike. Mike's privacy is violated. Therefore, if an attacker has the ordering knowledge, to protect the privacy, we need to add some randomness to the output ordering.

5.6 CASTLE complexity

We study the efficiency of the proposed approach by a formal analysis of the time/space complexity of the CASTLE algorithms. The parameters involved in the analysis have been summarized in Table 5.3. The third column in the table shows the typical values for the corresponding parameter; these values are obtained through the tuning process in the experiments presented in Section 5.7 below.

5.6.1 Time complexity

The time complexity of CASTLE is up to the main operations it carries out, that is: (1) Function bestSelection, selecting the best cluster where pushing a new tuple; (2) Function mergeClusters, merging clusters to generate a new k_s-anonymized cluster; (3)Function split$^{\mathsf{k}}$, splitting a cluster to into k_s-anonymized

Parameter	Description	Typical values
k	value of k in k_s-anonymity	100∼1,000
d	number of quasi-identifier attributes	6∼10
δ	delay constraint	5,000∼30,000
η	maximum number of non-k_s-anonymized clusters	50
N_{k_s}	maximum number of k_s-anonymized clusters	
C_e	time required to enlarge a single dimension of a cluster	
C_h	time required to replace the root of a heap	log(k)
S_g	space required to store the generalization of a QI attribute	
S_x	space required to store in memory a tuple	

Table 5.3: Parameters used in the complexity analysis

sub-clusters. Next, let us see them one by one in more details.

Function bestSelection. Given an arriving tuple x, Function bestSelection places it into its nearest non-k_s-anonymized cluster. The distance between x and a non-k_s-anonymized cluster $\mathcal{C} \in \Gamma$ is computed by Enlargement$(\mathcal{C}, x)$. In the worst case, cluster $\mathcal{C}$ need to enlarge over all its dimensions to enclose x. Thus, assuming the cost of calculating the enlargement over a dimension is C_e, then the time of Enlargement$(\mathcal{C}, x)$ is $C_e \cdot d$. Moreover, since Γ contains at maximum η non-k_s-anonymized clusters, we can estimate the worst case time complexity of bestSelection as:

$$C^w_{\mathsf{bestSelection}} = C_e \cdot d \cdot \eta \tag{5.1}$$

Function mergeClusters. Given a cluster $\mathcal{C}$, the merge operation is done repeatedly, each time combing $\mathcal{C}$ with its selected nearest neighbor. In each round, Function mergeClusters scans the whole list of non-k_s-anonymized clusters in Γ (excluding $\mathcal{C}$ itself), the one that incurs the minimum enlargement to $\mathcal{C}$ is chosen as the nearest neighbor. This alone requires time complexity of $(|\Gamma|-1)\cdot(C_e \cdot d)$. The merge runs continuously, until $\mathcal{C}$ reaches the size of k. In the worst case, Γ contains η clusters, and all of them are merged into $\mathcal{C}$. Moreover, at each iteration the size of Γ is decremented by one. Therefore, the upper bound time

complexity of mergeClusters is:

$$C^w_{merge} = \sum_{i=1}^{\eta-1}(C_e \cdot d)(\eta - i) = \frac{\eta \cdot (\eta - 1)}{2} \cdot (C_e \cdot d) \tag{5.2}$$

Function split^k. Recall lines 4-19 of Function split^k, we know that separating a sub-cluster from cluster $\mathcal{C}$ is done by randomly selecting a tuple $x \in \mathcal{C}$ and finding its k-1 nearest neighbors. Let φ be the set of buckets grouped by pid. In each bucket, a representative tuple is selected, and its distance from x is calculated. If the representative is closer to x than the root in the heap, it will replace the root and the heap will be adjusted accordingly. Therefore, to create a sub-cluster, the required time is $|\varphi| \cdot (C_e \cdot d + C_h)$, where $C_e \cdot d$ is the time to calculate the distance between two tuples, and C_h is the time to adjust the heap. This cost reaches the maximum when each tuple in $\mathcal{C}$ assumes a different pid value. Let n be the number of tuples in $\mathcal{C}$. Then the time to generate one sub-cluster is at most $n \cdot (C_e \cdot d + C_h)$. The above process repeatedly runs until all the possible sub-clusters are created. Since the number of sub-clusters is at most $\lfloor n/k \rfloor$ and in each iteration the size of $\mathcal{C}$ is decreased by k, the total cost of generating sub-clusters is no more than $\sum_{i=0}^{\lfloor n/k \rfloor - 1}(n - i \cdot k)(C_e \cdot d + C_h)$. Finally, the remaining tuples in $\mathcal{C}$ are inserted into the new sub-clusters generated above. For each of the remaining $n - k \cdot \lfloor n/k \rfloor$ tuples, it is calculated the enlargement of each sub-cluster (the cost is $\lfloor n/k \rfloor \cdot (C_e \cdot d)$). Therefore, the upper bound of the time complexity of the split^k is:

$$\begin{aligned} C^w_{\text{split}^k} &= \left[\sum_{i=0}^{\lfloor n/k \rfloor - 1}(n - i \cdot k)(C_e \cdot d + C_h)\right] + \\ &\quad (n - k \cdot \lfloor n/k \rfloor)(\lfloor n/k \rfloor \cdot (C_e \cdot d)) \\ &\approx \frac{n^2}{2k}(C_e \cdot d) \end{aligned} \tag{5.3}$$

Note that C_h is upper bounded by log(k) (the hight of heap) comparisons, so it is much smaller than $C_e \cdot d$, and the total cost of split^k is roughly $\frac{n^2}{2k}(C_e \cdot$

d). If tuples are uniformly distributed among non-k_s-anonymized clusters, then cluster $\mathcal{C}$ assumes $n = \delta/k$ tuples and the cost will be $\frac{\delta^2}{2k^3}(C_e \cdot d)$. However, with highly skew distribution, cluster $\mathcal{C}$ may contain almost all the δ tuples. In this case the cost is $\frac{\delta^2}{2k}(C_e \cdot d)$.

Let us now discuss the overall time complexity of CASTLE. In doing that, we estimate how many times bestSelection, mergeClusters, and split$^{\mathsf{k}}$ are called by CASTLE. Let $\mathcal{S}$ be the stream given in input to CASTLE, where with $|\mathcal{S}|$ we denote the number of tuples in $\mathcal{S}$.

bestSelection is called for each of the $|\mathcal{S}|$ tuples, thus its overall time complexity is:

$$C_{ToT\_\mathsf{bestSelection}} = |\mathcal{S}| \cdot C^w_{\mathsf{bestSelection}} = |\mathcal{S}| \cdot C_e \cdot d \cdot \eta \tag{5.4}$$

Moreover, since η is a predefined value, and will not exceed 100 in the experiments according to Table 5.3, we can estimate this cost as $O(|\mathcal{S}|)$.

Recall that a merge is performed when a tuple is expiring and no other way to output it is possible. Thus, we can say that in the worst case mergeClusters is called for each one of the $|\mathcal{S}|$ tuples. However, every time a merge is performed the number of tuples in $\mathcal{S}$ that still have to be evaluated by CASTLE decreases by at least k. Therefore, the upper bound of the total number of possible merge operations is $\frac{|\mathcal{S}|}{k}$, which implies that the overall time complexity of mergeClusters is:

$$\begin{aligned} C_{ToT\_\mathsf{mergeClusters}} &= \frac{|\mathcal{S}|}{k} \cdot C^w_{merge} \\ &= \frac{|\mathcal{S}|}{k} \cdot \frac{\eta \cdot (\eta - 1)}{2} \cdot (C_e \cdot d) \end{aligned} \tag{5.5}$$

Moreover, using the same reasoning we have made before regarding the η parameter, we can estimate this cost as O($\frac{|\mathcal{S}|}{k}$).

Finally, the split$^{\mathsf{k}}$ function is called when an expiring tuple is contained in a non-k_s-anonymized with size at least $2k$. In the worst case, we can say that this happens for all tuples in $\mathcal{S}$. However, similarly to the merge operation, every time a split is performed the number of tuples decreases by at least *2k*. Therefore, the number of possible split over $|\mathcal{S}|$ tuples is at most $\frac{|\mathcal{S}|}{2k}$, which implies that the overall time complexity of split$^{\mathsf{k}}$ is:

$$\begin{aligned} C_{ToT\_\mathsf{split}^{\mathsf{k}}} &= \frac{|\mathcal{S}|}{2k} \cdot C^{w}_{\mathsf{split}^{\mathsf{k}}} \\ &= \frac{|\mathcal{S}|}{2k} \cdot \frac{n^2}{2k} \cdot (C_e \cdot d) \\ &\leq \frac{|\mathcal{S}|}{2k} \cdot \frac{\delta^2}{2k} \cdot (C_e \cdot d) \end{aligned} \tag{5.6}$$

Therefore, the overall time complexity of CASTLE is bounded by the time complexity of split$^{\mathsf{k}}$, i.e., $O(\frac{|\mathcal{S}| \cdot \delta^2}{k^2})$. However, since $|\mathcal{S}|$ is much more greater than both k and δ, we can conclude that the overall time complexity of CASTLE is $O(\mathcal{S})$.

5.6.2 Space complexity

To estimate the space complexity we need to consider the information stored in memory when executing CASTLE. This consists of two main components: (1) the data stream tuples (i.e., tuples in the non-k_s-anonymized clusters), (2) the information about non-k_s-anonymized and k_s-anonymized clusters.

Regarding the first component, it is important to note that delay constraint ensures that, at any instant, there are at most δ tuples in memory. Thus, let S_x be the space required to store a single tuple, the first component requires $\delta \cdot S_x$ space.

Let us consider the second component. For each cluster, either non-k_s-anonymized or k_s-anonymized cluster, CASTLE stores the corresponding generalization. Thus, let S_g be the space required to store the generalization for a QI attribute, storing the generalization of a cluster requires $d \cdot S_g$ space. Since the number of non-k_s-anonymized clusters is upper bounded by η, and that of k_s-anonymized clusters is assumed no more than N_{k_s}[4], the space required by the second component will be at most $(\eta + N_{k_s}) \cdot d \cdot S_g$.

Put it all together, the total space required by CASTLE is:

$$S_{cost} = \delta \cdot S_x + (\eta + N_{k_s})(d \cdot S_g). \tag{5.7}$$

5.7 Performance evaluation

We have implemented CASTLE and have conducted several experiments. Our experiments have been designed with two objectives in mind. First we would like to verify that the proposed method is able to continuously anonymize a data stream while keeping the data useful. Second, to illustrate the effectiveness of CASTLE, we compare it with the approach presented in [10], which is the one comparable to our approach since it k-anonymizes the data set by a single pass on them. For these experiments, we used both synthetic and real world data. In particular, we have adopted the Adult data set from UC Irvine Machine Learning Repository [4], *UCI-Adult*, which has become a standard for studying k-anonymity. Moreover, in order to have a better simulation of a data stream, we have also considered the data set used in [39]. We refer to this as *SFU-Adult* [1]. We configure *UCI-Adult*[5] by removing tuples with missing values. Thus it contains 30,162 tuples. *SFU-Adult* contains 45,222 tuples after adding

[4] N_{k_s} is a threshold that can be determined based on the available memory.

[5] Both UCI-Adult and SFU-Adult contain only one tuple for each person. In Section 5.7.1 we will generate a stream with multiple tuples belonging to a same individual.

15,060 extra tuples to *UCI-Adult*. Of the 14 attributes in the two datasets, we choose 10 as the potential QI-attributes. Their characteristics are shown in Table 5.4.

Attribute	**Type**	**Numerical range**	
		# Leaves	**# Levels**
Age	Numerical	17 - 90	
Final-weight	Numerical	13492 - 1490400	
Education-num	Numerical	1 - 16	
Capital-gain	Numerical	0 - 99999	
Capital-loss	Numerical	0 - 4356	
Hours-per-week	Numerical	1 - 99	
Education	Categorical	16	5
Martial-status	Categorical	7	4
Occupation	Categorical	14	3
Native-country	Categorical	40	5

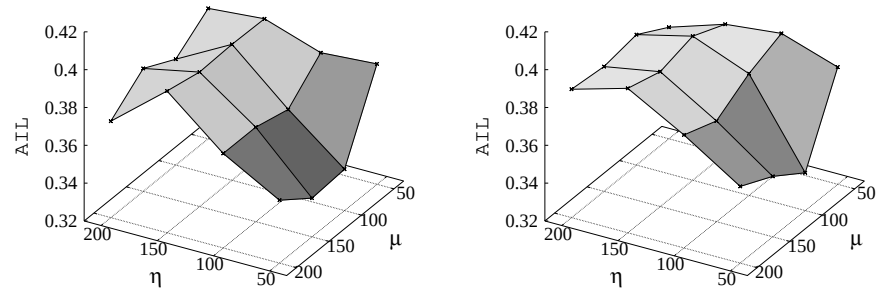
Table 5.4: Characteristics of the attributes

The algorithms were implemented by Microsoft visual C++ and the experiments were conducted on an Intel Pentium IV 2.4GHz with 1 GB RAM, running windows XP. In the following, we evaluate CASTLE using the metric described in Section 2.3.

5.7.1 Tuning CASTLE

The parameters that affect the performance of CASTLE are: δ, k, the number of QI attributes, μ, η, and the data distribution.

Effects of η and μ. CASTLE's adaptability to data distribution is controlled by two parameters: μ, the number of most recent k_s-anonymized clusters on which τ is calculated, and η, the threshold for controlling the maximum number of non-k_s-anonymized clusters in the memory. Figure 5.5 presents the average information loss of k_s-anonymized tuples by simultaneously varying η and μ.



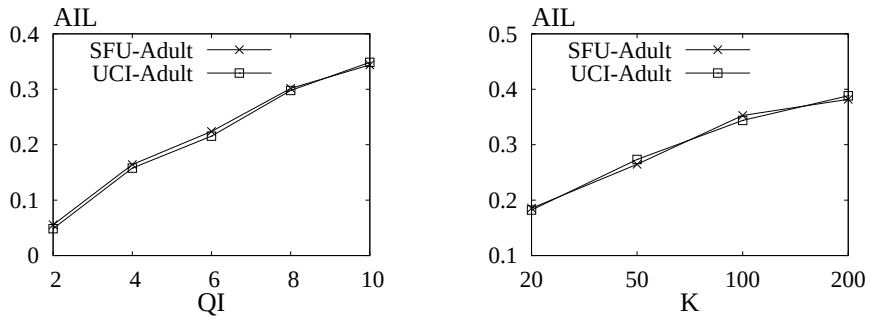
(a) SFU-Adult (b) UCI-Adult

Figure 5.5: Varying η and μ

We have set $k = 100$, $\delta = 10000$ and have used 10 QI attributes. We run the experiments on both the *SFU-Adult* (Figure 5.5(a)) and *UCI-Adult* (Figure 5.5(b)) data sets. In both data sets $\eta = 50$ minimizes the information loss. In the experiment of *SFU-Adult*, $\mu = 100$ and $\mu = 150$ yield the best information quality; in the experiment of *UCI-Adult*, $\mu = 100$ outperforms $\mu = 150$. In the following experiments, we shall use $\eta = 50$ and $\mu = 100$ as the default values.

Effects of *quasi-identifiers*. A further experiment measures how the size of QI affects the average information loss. The experiment has been conducted on the *UCI-Adult* and *SFU-Adult* data sets with $k = 100$ and $\delta = 10000$. Figure 5.6(a) reports how the average information loss varies by increasing the size of QI. The exploited metric space is defined based on QI attributes (i.e., each QI attribute is one dimension). When the size of QI increases, the data become more sparse in the defined space, and clusters are more likely to have 'big' minimum bounding boxes. This is related to *the curse of dimensionality* [8]. Therefore, as expected, the information loss increases when increasing the number of QI attributes.

Effects of k. Figure 5.6(b) shows how the average information loss increases by increasing the value of k. In this experiment we have considered 10 QI attributes, $\delta = 10000$, and both *UCI-Adult* and *SFU-Adult* data sets. The results are expected as a larger k implies that we need a larger cluster to anonymize



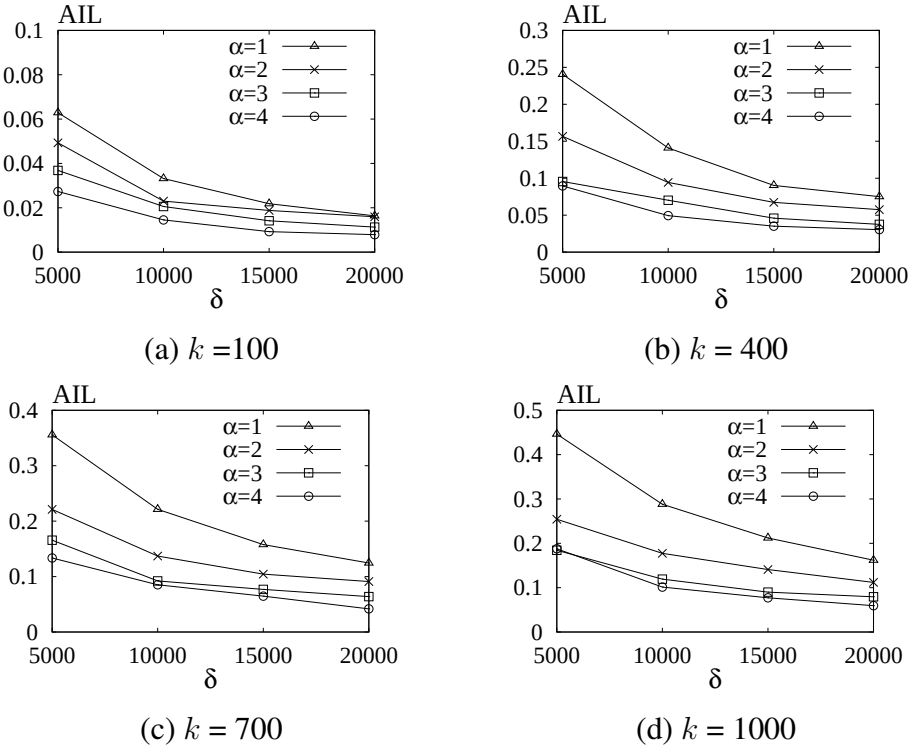
(a) varying QI (b) varying k

Figure 5.6: Varying QI and k

data — this translates to bigger loss in information.

Effects of data distribution. We have conducted several experiments to investigate how CASTLE scales with different data distributions. In order to do that, we have evaluated CASTLE on synthetic data sets following power law distribution generated by means of genzipf [2]. More precisely, a value v generated by genzipf has the following probability property: $p(v) = \frac{c}{v^{\alpha}}, v \in \{1, \ldots, N\}$ and $\sum_{v=1}^{N} p(v) = 1$, where c is the normalization constant automatically initialized and N is the biggest possible integer value in generation. Figure 5.7 reports the average information loss with different α values. The experiments have been conducted with fixed k and by varying δ. We have evaluated the behavior of CASTLE with respect to data distribution with different k values (cfr. Figures 5.7(a), . . . 5.7(d)). From the results, it is clear that CASTLE is very effective for clustered data. This is promising as real data are typically clustered.

Effect of multiple tuples referring to the same person. In the previous experiments we have considered the UCI-Adult and SFU-Adult datasets, by assuming that each tuple refers to a distinct person, that is, each tuple has a different pid. However, in a data stream multiple tuples may refer to a same person, so we have run further experiments to test how this duplication of pids may



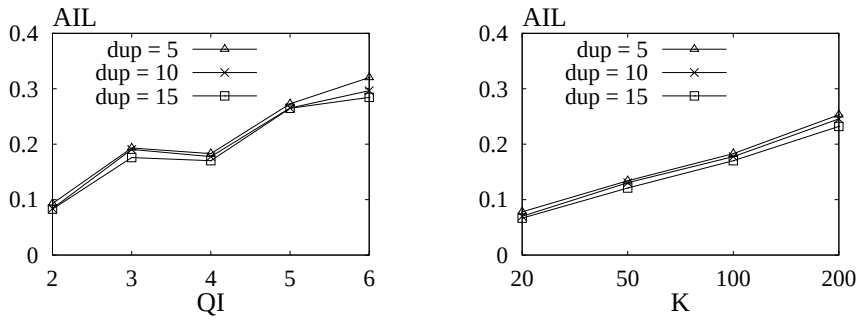
(a) k =100 (b) k = 400

(c) k = 700 (d) k = 1000

Figure 5.7: Information loss on power-law synthetic data

affect the information quality. In particular, we exploit the *IBM Quest Synthetic Data Generation Code* [14] to generate a set of tuples T_{tran} with schema $(pid, tid, list\ of\ items)$, where pid is a customer's id, tid represents transaction id, and *list of items* are the items a customer has bought. Then, we join T_{tran} with UCI-Adult on their pid to produce the data stream. Since T_{tran} is bigger than UCI-Adult, which contains 30,162 tuples only, we join tuples in T_{tran} with $pid > 30,162$ with randomly selected tuples in UCI-Adult. As a result, the streaming tuples have the schema $(pid, tid, QI, list\ of\ items)$. Moreover, to simulate the presence of more transactions referring to the same person, when a streaming tuple x is pushed into a cluster, CASTLE splits it into multiple tuples with schema $(pid, tid, QI, item)$, one for each distinct item contained in x. According to this approach, we have generated three different synthetic datasets,

by varying the length of $list\ of\ items$.



(a) varying QI (b) varying K

Figure 5.8: Information loss on transaction stream

The experiments are carried out by setting $\delta = 10000$, and varying k and the size of QI. In particular, in Figure 5.8(a) we set $k = 100$ and vary the size of QI, where $dup = 5$, $dup = 10$, and $dup = 15$ indicates the synthetic dataset on average with 5, 10, and 15 items in the $list\ of\ items$, respectively. In general, when the size of QI increases, the information quality degrades. In Figure 5.8(b) we set the size of QI to be 4, and vary k. We find that information loss uniformly increases as a function of k.

5.7.2 Utility

In this section, we study the utility of the anonymized stream. We consider aggregated queries as they are the basis of statistical analysis and many mining applications (e.g., decision tree and association rule mining), and exploit the metric [82] to evaluate the query accuracy:

`SELECT COUNT(*) FROM Anonymized-stream`

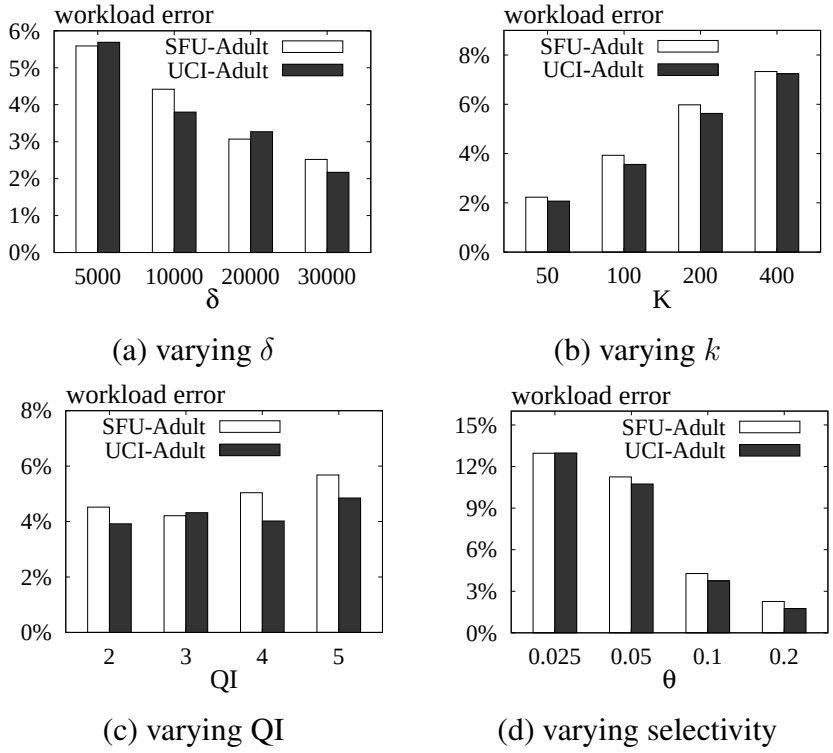
`WHERE pred(`A_1`) AND ... AND pred(`A_λ`) AND pred(`$\mathcal{SA}$`)`

Each A_i is a QI attribute. $\mathcal{SA}$ is a sensitive attribute. The query has predicates on λ randomly selected QI attributes and $\mathcal{SA}$. For each of those $\lambda + 1$ attributes A, $pred(A)$ has the form of $A \in R_A$, where R_A is an arbitrary interval in the

domain of A. The *expected selectivity* over the table is $0<\theta<1$. Assuming data are uniformly distributed, θ can be achieved if each attribute A selects records within a range of length $|A| \cdot \theta_A$ of its domain, such that $(\theta_A)^{\lambda+1} = \theta$. In effect, the length of R_A should be $|A| \cdot \theta^{\frac{1}{\lambda+1}}$, where $|A|$ is the domain length of attribute A.

We evaluate the same queries on both output and input streams, i.e.. $\mathcal{S}_{out}$ and $\mathcal{S}$. Given a query Q, if its result in i^{th} window of $\mathcal{S}_{out}$ is est, and if its result in i^{th} window of $\mathcal{S}$ is act, then the *relative error* of this query is defined as $|act - est|/|act|$. Given a window W in a data stream, we run 5,000 queries on W, and take the median relative error of these queries as the *window error*. As W advances, a sequence of windows are generated. We calculate all the window errors, and take their average as the *workload error*. Furthermore, the metric [82] assumes a uniform distribution of tuples in each EC, so attributes with a skewed distribution of values will not be included in the experiments. Among all the 30,162 tuples in UCI-Adult dataset, 27,624 ones have the value of 0 on attribute *capital-gain*, which has a range [0,99999]. Similarly, attribute *capital-loss* has the value of 0 in 28,735 tuples, even though its range is [0,4356]. Therefore, neither *capital-gain* nor *capital-loss* is involved in the evaluation.

Figure 5.9 reports the experiment results. We set the window size $|W|$ as large as δ. In Figure 5.9(a) we set $|QI| = 4$, $k = 100$ and $\theta = 0.1$, and vary δ. When δ increases, CASTLE can select nearest neighbors among more tuples, thus the formed clusters are more likely to have smaller minimum bounding boxes. Hence, the utility increases as a function of δ. In Figure 5.9(b) we consider 4 QI attributes, set $\delta = 10000$ and $\theta = 0.1$, and vary k. When k increases, utility degrades since a larger k requires a cluster containing more tuples, which implies a larger generalization. Figure 5.9(c) presents the result



(a) varying δ (b) varying k

(c) varying QI (d) varying selectivity

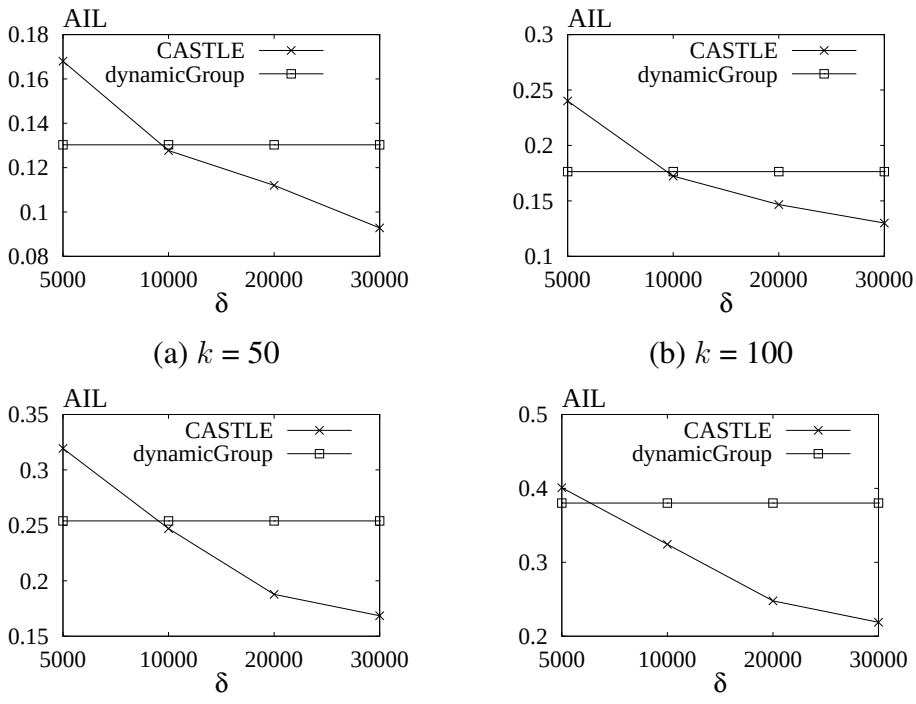
Figure 5.9: Workload error

of varying QI when $k = 100$, $\delta = 10000$, and $\theta = 0.1$; the workload error does not change uniformly as the size of QI increases. However, in general utility is reduced when QI has a higher dimensionality. Figure 5.9(d) is the result by varying selectivity θ with 4 QI attributes, $k = 100$, and $\delta = 10000$. Bigger θ yields a higher accuracy since a bigger θ indicates bigger R_A, thus allowing a larger generalization of attribute A. In all these experiments, the workload error of CASTLE is less than 13%, indicating high utility of anonymized streams.

5.7.3 Comparative study

As there is no previous work of continuously anonymizing data streams, we employ a comparison benchmark presented in [10], hereafter called dynamicGroup.

dynamicGroup outputs the anonymized data *only* after the whole process is complete (i.e., after scanning the entire data set). To create a level playing field, CASTLE anonymizes the stream data up to its end as follows: after the last tuple from the stream is pushed to a cluster, CASTLE outputs all the clusters with sizes not less than k. Then CASTLE generalizes all the tuples which fall in k_s-anonymized clusters. Finally, CASTLE merges all the remaining non-anonymized tuples to form a cluster and outputs it. dynamicGroup uses historical data to build the first set of clusters in the memory. We take the first n tuples in UCI-Adult as the historical data, and all the remaining ones as the streaming data. We vary n from 2,000 to 8,000, and select the best one for dynamicGroup, that is 8,000.



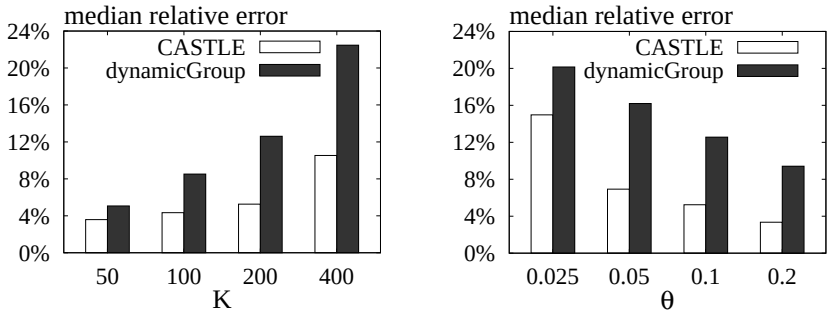
(a) k = 50 (b) k = 100

(c) k = 200 (d) k = 400

Figure 5.10: A comparison with dynamicGroup on information loss

The first experiment is on average information loss, with *UCI-Adult* as the

test dataset, and its 6 continuous attributes as QI. Note that dynamicGroup can process continuous attribute only. Figure 5.10 reports the results by varying δ. We have studied several values of k (see Figures 5.10(a), (b), (c), and (d)). It is important to note that dynamicGroup does not consider the delay constraint. Thus, it retains tuples till the end of the process, which obviously influences the information loss. For a fair comparison between CASTLE and dynamicGroup we must consider only the average information loss of CASTLE with δ set to *infinity*. As shown in Figure 5.10, the information quality of CASTLE increases as a function of δ; CASTLE outperforms dynamicGroup when δ is 10,000.



(a) varying k (b) varying selectivity

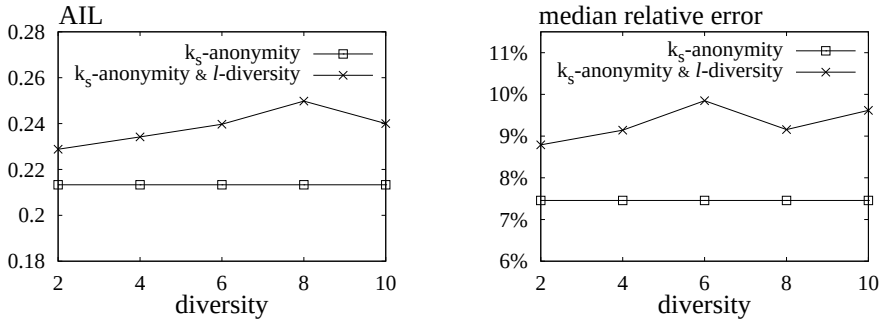
Figure 5.11: A comparison with dynamicGroup on median relative error

We also compare CASTLE with dynamicGroup with respect to the utility. In Figure 5.11 we consider 4 continuous QI attributes (excluding *capital-gain* and *capital-loss* since they are not uniformly distributed), and set $\delta = 20,000$. In Figure 5.11(a) we vary k, by setting $\theta = 0.1$, whereas in Figure 5.11(b) we vary the selectivity θ, by setting $k = 200$. In both experiments CASTLE outperforms dynamicGroup clearly.

5.7.4 k_s-anonymity and ℓ-diversity

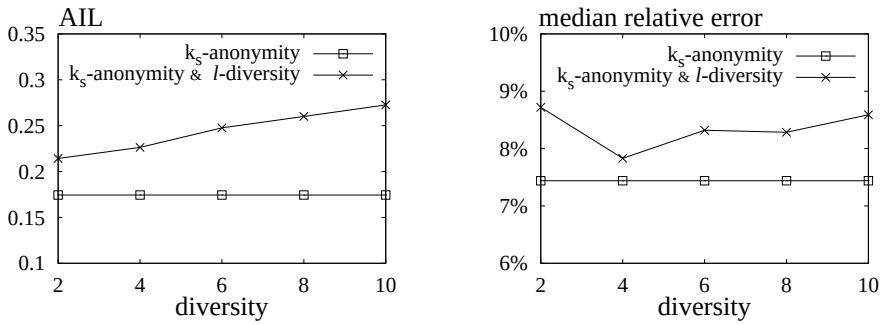
In our last experiment, we extend CASTLE to support ℓ-diversity. Figure 5.12 and Figure 5.13 report the results on the UCI-Adult data set, while setting

$\delta = 10000$, $k = 100$, and $|QI| = 5$. In Figure 5.12 the sensitive attribute is *age*, whereas in Figure 5.13 *occupation* is used as the sensitive attribute. When the diversity increases, a cluster should contain potentially more tuples, thus both information quality and utility decrease. However, when k increases and/or diversity ℓ increases, the security of the anonymized data stream is higher. Therefore, based on all the above experiments, we can conclude that the information loss/utility and the security is a trade-off: to better preserve the privacy we may need to degrade information quality/utility; to preserve more information/utility, we need to reduce k and/or ℓ.



(a) information loss (b) utility

Figure 5.12: k_s-anonymity and ℓ-diversity: Age



(a) information loss (b) utility

Figure 5.13: k_s-anonymity and ℓ-diversity: Occupation

5.8 Summary

In this chapter we have presented CASTLE a cluster-based framework to k-anonymize data streams. Relevant features of CASTLE are the enforcement of delay constraints, its adaptability to data distributions, and the cluster reuse strategy that improves the performance without compromising security. Performance evaluation reported in this chapter have shown that CASTLE is efficient and effective.

CHAPTER 6

SABRE$_W$: WINDOW-BASED t-CLOSENESS ON DATA STREAMS

In this chapter we study the notion of t-closeness for streaming data, thus serving anonymized tuples with stronger privacy than what CASTLE (Chapter 5) ensures. Since data streams are continuous and unbounded, the existing t-closeness concept is not directly applicable. Thus, we revise the definition of t-closeness, by restricting closeness constraint in each window only, instead of the whole data stream. In particular, we propose (ω, t)-closeness: for any EC, there exists a window that has a size of ω and contains the EC, so that the difference of the $\mathcal{SA}$ distribution between the EC and the window is no more than a threshold t. At the same time, we design a customized algorithm to ensure (ω, t)-closeness.

6.1 Introduction

As far as we know, CASTE, presented in Chapter 5, is the first scheme proposed up to now that is able to continuously anonymize streaming data, while ensuring the maximum delay between any tuple's input and its output not more than a threshold δ. The anonymized data output by CASTLE are served with a privacy guarantee enabled by k-anonymity and/or ℓ-diversity. However, k-anonymity

suffers from *homogeneous attack*, when most tuples in an EC share a same $\mathcal{SA}$ value (Section 1.1.1). ℓ-diversity copes with this limitation with a diversity requirement on the $\mathcal{SA}$ values in the EC. Nevertheless, it is susceptible to skewness and similarity attacks (Section 1.1.2), since it pays no attention to the unavoidable adversarial knowledge on the global $\mathcal{SA}$ distribution in the whole table. CASTLE is developed according to k-anonymity and/or ℓ-diversity. Therefore, it inevitably shares the same drawbacks with its underlying privacy models.

Among all the solutions to strengthen the security of streaming tuples, a possible one is applying an enhanced privacy model on the data. Such a solution is straightforward but effective, since the extent to which the anonymized data are immune from attacks is up to their supporting privacy model. Here, we make use of t-closeness [52], a successor of k-anonymity and ℓ-diversity with improved security. t-closeness is originally proposed for static microdata, with an assumption that all the data involved are available before any anonymization. It takes the global $\mathcal{SA}$ distribution of the whole data set as the baseline of the prior knowledge, and constrains the extra information an attacker gains after seeing an EC. However, data streams are continuous and unbounded. Consequently, such a global distribution is unavailable. Therefore, the t-closeness model cannot be applied in the context of streaming data directly.

In this chapter we first revise the definition of t-closeness with respect to the unique characteristics of data streams. We restrict the closeness constraints in each window instead of the whole data stream by (ω, t)-closeness — for any EC, there exists a window that has a size of ω and contains the EC, so that the difference of the $\mathcal{SA}$ distribution between the EC and the window is no more than a threshold t. As discussed in Section 5.1, the output stream may be used to real-time react to some anomalies. Therefore, we enforce *expiring constraint* on

the anonymization procedure to make sure the freshness of output data. Given the window size ω and $\mathcal{I}$, the time interval of window advance, the delay between the input of a tuple and its output is at most $\omega + \mathcal{I}$. With (ω, t)-closeness principle and expiring constraint as the underlying model requirements, we put forward $\mathsf{SABRE_W}$, a window-based t-closeness scheme, which reuses static t-closeness framework SABRE (see Chapter 3) as a building block to anonymize streaming tuples. Coming along with the algorithm is a solid theory foundation, proving that $\mathsf{SABRE_W}$ strictly follows the stated model constraints. We compare $\mathsf{SABRE_W}$ with schemes extended from k-anonymity algorithms; the evaluation results show that $\mathsf{SABRE_W}$ outperforms them in terms of both information quality and time efficiency.

The remaining of the chapter is organized as follows. The next section introduces a window-based t-closeness-resembling model. We develop an algorithm tailored for the model in Section 6.3, and prove its soundness by a solid theory foundation in Section 6.4. After that, Section 6.5 reports the results of an experimental evaluation. Finally, we discuss how to reuse the algorithm to adapt β-likeness to data streams in Section 6.6, and conclude this chapter in Section 6.7.

6.2 The privacy modeling

In this section we present the privacy model underlying our anonymization scheme. It preserves the privacy of streaming tuples and ensures their freshness through two concepts: *(ω,t)-closeness* and *expiring constraint*. Table 6.1 lists some symbols that will be used throughout the chapter; some notations and the relationship among them will be later clarified.

Notation	Denotation
$\mathcal{S}_{in}$	The input data stream
$\mathcal{S}_{out}$	The output data stream generated from $\mathcal{S}_{in}$
$\mathcal{W}$	A window in $\mathcal{S}_{in}$
$\mathcal{I}$	Time interval of window advance
ω	Window size; $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer
$\mathcal{R}(\mathcal{W})$	All the tuples in $\mathcal{W}$
$\mathcal{R}(\mathcal{W}, o)$	All the already output tuples of $\mathcal{W}$
$\mathcal{R}(\mathcal{W}, \bar{o})$	All the not-yet-output tuples in $\mathcal{W}$, $\mathcal{R}(\mathcal{W}) = \mathcal{R}(\mathcal{W}, o) \cup \mathcal{R}(\mathcal{W}, \bar{o})$
$\mathcal{R}(\mathcal{W}, e\bar{o})$	Expiring and not-yet-output tuples in $\mathcal{W}$
$\mathcal{SA}(\mathcal{W})$	$\mathcal{SA}$ distribution in $\mathcal{R}(\mathcal{W})$
$\mathcal{SA}(\mathcal{W}, \bar{o})$	$\mathcal{SA}$ distribution in $\mathcal{R}(\mathcal{W}, \bar{o})$
$\mathcal{D}_{\mathcal{W}\bar{o}}$	EMD($\mathcal{SA}(\mathcal{W})$, $\mathcal{SA}(\mathcal{W}, \bar{o})$)

Table 6.1: Streaming notations

Definition 6.1 ((ω,t)-closeness)**.** *Let $\mathcal{S}_{in}$ be an input stream, $\mathcal{SA}$ its sensitive attribute, and $\mathcal{S}_{out}$ the output stream generated from $\mathcal{S}_{in}$. We monitor $\mathcal{S}_{in}$ on a window of size ω. We say that $\mathcal{S}_{out}$ follows the (ω, t)-closeness, if and only if the following conditions hold:*

- *For each tuple $x \in \mathcal{S}_{in}$, there exists in $\mathcal{S}_{out}$ an equivalence class, which contains the corresponding anonymized tuple of x.*
- *For each equivalence class $\mathcal{G} \in \mathcal{S}_{out}$, there exists a window $\mathcal{W}$ in $\mathcal{S}_{in}$, such that $\mathcal{G}$ is generated from the tuples in $\mathcal{W}$, and the $\mathcal{SA}$ distribution in $\mathcal{G}$ does not differ from that of all tuples in $\mathcal{W}$ by more than a given threshold t.*

(ω, t)-closeness considers the $\mathcal{SA}$ distribution of all the tuples in a window of size ω as the prior knowledge, and the $\mathcal{SA}$ distribution in an output EC as the posterior knowledge. It limits the information gain from the prior knowledge to the posterior one, by requiring that the difference of the two mentioned $\mathcal{SA}$

distributions be at most t, a threshold. (ω, t)-closeness assumes that the $\mathcal{SA}$ distribution in a window of size ω is non-sensitive, thus publishable. How to better determine a non-sensitive prior knowledge of $\mathcal{SA}$ distribution by allowing flexible window size is a future work.

We have discussed in Section 5.1 a feature that distinguishes data streams from static datasets, that is, multiple tuples belonging to a single person may appear in the streaming data. Thus, without a careful management there does exist the possibility, no matter how small it is, that all the tuples in a single EC belong to a same person. In this case, the QI value of this person will remaining unchanged even after generalization. Consequently, he/she will be re-identified through linking attack. To address this arising issue, we combine (ω,t)-closeness with k-anonymity in the experiments (Section 6.5), requiring that the set of persons referred to by all the tuples in an EC has a cardinality of at least k.

Definition 6.2 (Expiring constraint)**.** *Assume an (ω, t)-closeness scheme T, which takes as input a data stream $\mathcal{S}_{in}$ and generates an output data stream $\mathcal{S}_{out}$. Let $\mathcal{I}$ be a positive integer, and $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer. Assume that each window in $\mathcal{S}_{in}$ advances by $\mathcal{I}$ tuples. T satisfies the expiring constraint if and only if, each time a window in $\mathcal{S}_{in}$ advances its first $\mathcal{I}$ tuples expire and are output.*

The above concept synchronizes the output of tuples with the window advance. Each time a window slides forwards, the tuples outside it will be output. Such a synchronization controls the delay of tuples (i.e., also their freshness) with regard to window size and the time interval of window sliding.

CASTLE in Section 5.3 has already defined *delay constraint*, requiring the delay between the input of a tuple and its output below a threshold δ. The above

Definition 6.2 introduces another time constraint, confining the upper bound of the delay by two parameters, i.e., ω and $\mathcal{I}$ (Lemma 6.1 below will prove that the upper bound is $\omega + \mathcal{I}$). Such a constraint synchronizes the tuple output with window sliding. In addition, it allows for more flexibility in the anonymization process: Users can tune ω and $\mathcal{I}$ based on the requirements of specific applications. Anyway, the two concepts are related. On the one hand, expiring constraint can be reduced to delay constraint by setting $\mathcal{I} = 1$ and $\omega = \delta - 1$. On the other hand, delay constraint can also be seen as a special case of expiring constraint.

Lemma 6.1. *Assume a (ω, t)-closeness on input stream $\mathcal{S}_{in}$, which advances by time interval $\mathcal{I}$. Then the delay time between the arrival of a tuple and its output will not exceed $\omega + \mathcal{I}$.*

The proof of the lemma is trivial. So it is omitted.

As the window slides, a sequence of windows $\mathcal{W}_1$, $\mathcal{W}_2$, ..., $\mathcal{W}_i$, $\mathcal{W}_{i+1}$, ... are generated. When window $\mathcal{W}_i$ advances by $\mathcal{I}$ tuples, a new window $\mathcal{W}_{i+1}$ is generated. The first (oldest) $\mathcal{I}$ tuples in $\mathcal{W}_i$, which are left outside $\mathcal{W}_{i+1}$, should be output. We call these *expiring* tuples. Thus, in Figure 6.1, as the window slides from $\mathcal{W}_1$ to $\mathcal{W}_2$, the tuples in $\mathcal{W}_1 \setminus \mathcal{W}_2$ (the first $\mathcal{I}$ tuples in $\mathcal{W}_1$), are expiring and should be output.

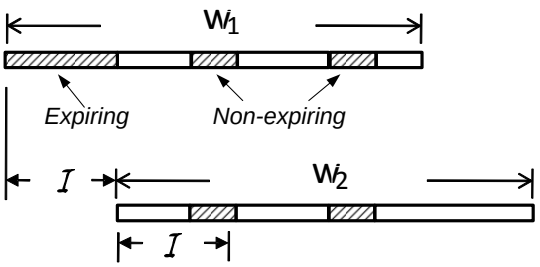


Figure 6.1: Windows and their advances

Assume that $\mathcal{S}_{in}$ is an input (original) stream, and an (ω,t)-closeness scheme is applied on it. As tuples arrive from $\mathcal{S}_{in}$, they are first buffered in memory. When the window advances from $\mathcal{W}_i$ to $\mathcal{W}_{i+1}$, we create ECs for the expiring tuples in $\mathcal{W}_i$. Still, the expiring tuples on their own may not satisfy t-closeness to the $\mathcal{SA}$ distribution of all tuples in $\mathcal{W}_i$; thus, the created ECs may contain both expiring and non-expiring tuples. As an example, let $\mathcal{W}_i$ be the advancing window and take $\mathcal{I} = 1$. With high probability, the single expiring tuple (the first tuple in $\mathcal{W}_i$) is not *t*-close to the distribution of all the tuples in $\mathcal{W}_i$. Therefore, we need to accompany the single expiring tuple with some non-expiring tuple(s) in $\mathcal{W}_i$ to form an EC. After the ECs for the expiring tuples are generated, we output all these ECs.

As the window advances continuously, an output stream $\mathcal{S}_{out}$ is generated. Given that the output ECs may contain non-expired tuples, when the window slides to $\mathcal{W}_{i+1}$, some tuples in $\mathcal{W}_{i+1}$ may have already been output. For instance, in Figure 6.1, as the window slides from $\mathcal{W}_1$ to $\mathcal{W}_2$, the first $\mathcal{I}$ tuples in $\mathcal{W}_1$ are expiring. ECs are created for the expiring tuples, and contain both the expiring tuples and non-expiring ones; assume ECs are composed of the tuples in the shaded segments in $\mathcal{W}_1$. These ECs are output. Thus, by the time the window assumes the position $\mathcal{W}_2$, some of its tuples have already been output.

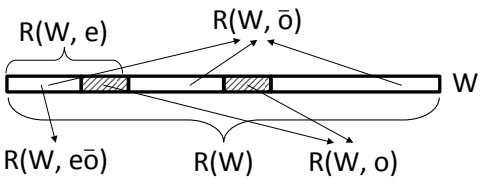


Figure 6.2: The classification of tuples

Based on the above distinctions, we define the following classes of tuples, illustrated in Figure 6.2 (also listed in Table 6.1):

1. $\mathcal{R}(\mathcal{W})$, all tuples in window $\mathcal{W}$;

2. $\mathcal{R}(\mathcal{W}, o)$, the already output tuples from $\mathcal{W}$;

3. $\mathcal{R}(\mathcal{W}, \bar{o})$, the not-yet-output tuples in $\mathcal{W}$, $\mathcal{R}(\mathcal{W}, o) \cup \mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$;

4. $\mathcal{R}(\mathcal{W}, e)$, the expiring tuples of $\mathcal{W}$ when it advances (i.e., the first $\mathcal{I}$ tuples of $\mathcal{W}$), and

5. $\mathcal{R}(\mathcal{W}, e\bar{o})$, the expiring and not-yet-output tuples of $\mathcal{W}$.

6.3 The algorithm

Now we are ready to present $\mathsf{SABRE_W}$, our window-based t-closeness scheme in the context of data streams. $\mathsf{SABRE_W}$ is extended from our static t-closeness framework SABRE (see Chapter 3), in particular, we reuse SABRE to anonymize a window of tuples (i.e., the set of tuples staying in a same window). Before presenting the detailed algorithm, we first give some intuition with an example.

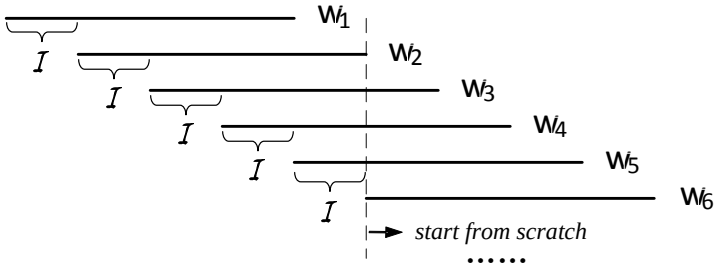


Figure 6.3: An example for Algorithm $\mathsf{SABRE_W}$

Example 6.1. *Assume that $\mathcal{S}_{in}$ is the original input stream of $\mathsf{SABRE_W}$. Figure 6.3 illustrates the sequence of windows, as the window advances in $\mathcal{S}_{in}$. When $\mathcal{W}_1$ is sliding at instant $\imath_1$, $\mathcal{R}(\mathcal{W}_1) = \mathcal{R}(\mathcal{W}_1, \bar{o})$; i..e, none of its tuples is yet output. We partition tuples in $\mathcal{W}_1$ into $S^1_{\mathcal{G}}$, a list of ECs, so that the $\mathcal{SA}$ distribution*

of each EC in $S^1_{\mathcal{G}}$ is t*-close to that of all the tuples in $\mathcal{W}_1$. ECs that contain expiring tuples are output and deleted from $S^1_{\mathcal{G}}$. After that $\mathcal{W}_1$ slides to $\mathcal{W}_2$. When $\mathcal{W}_2$ begins to advance at instant $\imath_2$, assume that $\mathcal{R}(\mathcal{W}_2, \bar{o})$ and $\mathcal{R}(\mathcal{W}_2)$ have similar $\mathcal{SA}$ distributions (i.e., the $\mathcal{SA}$ distribution of not-yet-output tuples in $\mathcal{W}_2$ is similar to that of all tuples in $\mathcal{W}_2$). Like what we did for $\mathcal{W}_1$, we partition $\mathcal{R}(\mathcal{W}_2, \bar{o})$ into a set of ECs, $S^2_{\mathcal{G}}$ (more in* Case 2 *of* SABRE$_{\mathsf{W}}$ *below). All the ECs with expiring tuples are output and removed from $S^2_{\mathcal{G}}$. Then, $\mathcal{W}_2$ slides to $\mathcal{W}_3$. When $\mathcal{W}_3$ slides at instant $\imath_3$, assume that $\mathcal{R}(\mathcal{W}_3, e\bar{o})$ is empty (i.e., all expiring tuples have already been output). Under this circumstance, we do not have to anonymize any tuple, and $\mathcal{W}_3$ simply slides to $\mathcal{W}_4$ (see* Case 1 *of* SABRE$_{\mathsf{W}}$ *below). When $\mathcal{W}_4$ advances at instant $\imath_4$, assume that the $\mathcal{SA}$ distribution of $\mathcal{R}(\mathcal{W}_4, \bar{o})$ is very 'different' from that of $\mathcal{R}(\mathcal{W}_4)$. If we partition $\mathcal{R}(\mathcal{W}_4, \bar{o})$ into a set of ECs, some of them (containing expiring tuples) may also be very 'different' from $\mathcal{R}(\mathcal{W}_4)$ with respect to $\mathcal{SA}$ distribution, and (ω, t)-closeness cannot be guaranteed any more. However, $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ (i.e., the expiring and not-yet-output tuples in $\mathcal{W}_4$) falls in $\mathcal{W}_2$, i.e. $\mathcal{R}(\mathcal{W}_4, e\bar{o}) \subset \mathcal{R}(\mathcal{W}_2, \bar{o})$[1]. Since $\mathcal{R}(\mathcal{W}_2, \bar{o})$ is already partitioned into $S^2_{\mathcal{G}}$, each tuple in $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ must belong to an EC of $S^2_{\mathcal{G}}$ (see* Case 3 *of* SABRE$_{\mathsf{W}}$ *below). Thus, we output all the ECs in $S^2_{\mathcal{G}}$ and delete them. Hence, we ensure that all tuples in $\mathcal{R}(\mathcal{W}_4, e\bar{o})$ are output, while now all the tuples in $\mathcal{W}_2$ have also been output. Then, as $\mathcal{W}_5$ slides at instant $\imath_5$, all its expiring tuples have already been output because they fall in $\mathcal{W}_2$. So $\mathcal{W}_5$ simply slides to $\mathcal{W}_6$. When $\mathcal{W}_6$ enters, none of its tuples is output, and the anonymization of* SABRE$_{\mathsf{W}}$ *starts from scratch.*

Algorithm SABRE$_{\mathsf{W}}$ anonymizes an input stream $\mathcal{S}_{in}$ into an output stream $\mathcal{S}_{out}$ to attain the *(ω,t)-closeness* and *expiring constraint*. Window $\mathcal{W}$ buffers

[1] $\mathcal{R}(\mathcal{W}_2, \bar{o})$ *is* not *the set of not-yet-output tuples of $\mathcal{W}_2$ at instant $\imath_4$; it is the set of not-yet-output tuples of $\mathcal{W}_2$ at instant $\imath_2$ when $\mathcal{W}_2$ is advancing.*

Algorithm: SABRE$_{\mathsf{W}}$($\mathcal{S}_{in}$, t, ω, $\mathcal{I}$)

```
1  Let S_in and S_out be the input and output streams respectively;
2  Let W be a window in S_in;
3  W buffers the first ω tuples from S_in;
4  Let S_G be a set of ECs, initialized to be empty;
5  while I new tuples have arrived from S_in do
6  Case 1:
7      if |R(W, eō)| = 0 then
8          Update W by advancing I tuples;
9          Continue;
10     Let SA(W, ō) be the SA distribution of R(W, ō);
11     Let SA(W) be the SA distribution of R(W);
12     Let D_{Wō} = EMD(SA(W, ō), SA(W));
13 Case 2:
14     if D_{Wō} < t then                          /* DIRECT begins */
15         S_G = SABRE (R(W, ō), SA, t − D_{Wō});
16         foreach x ∈ R(W, eō) do
17             Find G ∈ S_G, which contains x;
18             Put G to S_out;
19             Delete G from S_G                    /* DIRECT end */
20 Case 3:
21     else                                        /* INDIRECT begins */
22         foreach G ∈ S_G do
23             Add G to S_out;
24             Delete G from S_G                    /* INDIRECT ends */
25     Update W by advancing I tuples;
```

the first ω tuples from input stream $\mathcal{S}_{in}$ (steps 2-3). When $\mathcal{I}$ new tuples have arrived from $\mathcal{S}_{in}$ (step 5), the first $\mathcal{I}$ tuples in $\mathcal{W}$ expire and need to be output. $\mathcal{R}(\mathcal{W}, e\bar{o})$ is the set of expiring and not-yet-output tuples in $\mathcal{W}$ at the moment of its advance. We distinguish the following cases.

- *Case 1 (steps 7-9):* $|\mathcal{R}(\mathcal{W}, e\bar{o})| = 0$. All expiring tuples in $\mathcal{W}$ have already been output (when a window advanced in a previous iteration), so we do not need to anonymize any tuple. $\mathcal{W}$ simply slides by $\mathcal{I}$ tuples (step 8) and SABRE$_{\mathsf{W}}$ goes back to step 5 (step 9).

- *Case 2 (steps 14-19):* $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} < t$. In this case the not-yet-output tuples in $\mathcal{W}$ and all the tuples in $\mathcal{W}$ have similar $\mathcal{SA}$ distributions. We partition $\mathcal{R}(\mathcal{W}, \bar{o})$, the not-yet-output tuples in $\mathcal{W}$, into a set

of ECs and output those containing expiring tuples. Algorithm SABRE is called and $\mathcal{R}(\mathcal{W}, \bar{o})$ is anonymized into $S_{\mathcal{G}}$, a set of ECs (step 15). For each equivalence class $\mathcal{G} \in S_{\mathcal{G}}$, EMD$(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W}, \bar{o})) \leq t - \mathcal{D}_{\mathcal{W}\bar{o}}$, where $\mathcal{SA}(\mathcal{G})$ is the $\mathcal{SA}$ distribution in $\mathcal{G}$. Since EMD$(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W}, \bar{o}))$ + EMD$(\mathcal{SA}(\mathcal{W}, \bar{o}), \mathcal{SA}(\mathcal{W})) \leq t - \mathcal{D}_{\mathcal{W}\bar{o}} + \mathcal{D}_{\mathcal{W}\bar{o}} = t$, i.e. the EMD of transforming $\mathcal{SA}(\mathcal{G})$ to $\mathcal{SA}(\mathcal{W})$ via $\mathcal{SA}(\mathcal{W}, \bar{o})$ is at most t, we conclude that EMD$(\mathcal{SA}(\mathcal{G}), \mathcal{SA}(\mathcal{W})) \leq t$ (the proof is similar to that of Theorem 3.4). Therefore, $\mathcal{G}$ is *t*-close to $\mathcal{R}(\mathcal{W})$. Then, ECs that contain expiring tuples are output and deleted from $S_{\mathcal{G}}$ (steps 16-19). We denote the procedure of Case 2 by DIRECT, which means that ECs for $\mathcal{R}(\mathcal{W}, e\bar{o})$ are obtained *directly* from a partition of not-yet-output tuples in $\mathcal{W}$.

- *Case 3 (steps 21-24):* $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. In this case the $\mathcal{SA}$ distribution of $\mathcal{R}(\mathcal{W}, \bar{o})$ and that of $\mathcal{R}(\mathcal{W})$ are very 'different'. If we partition $\mathcal{R}(\mathcal{W}, \bar{o})$ into a set of ECs (as *Case 2*), some ECs containing expiring tuples may also be very 'different' from $\mathcal{R}(\mathcal{W})$ in terms of $\mathcal{SA}$ distribution, and (ω,t)-closeness may not be guaranteed. However, each tuple of $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $S_{\mathcal{G}}$ (see Theorem 6.1 below for a formal proof). To anonymize and output $\mathcal{R}(\mathcal{W}, e\bar{o})$ we output all the ECs in $S_{\mathcal{G}}$ and clear it (steps 22-24). We denote the procedure of Case 3 by INDIRECT, which means that ECs for $\mathcal{R}(\mathcal{W}, e\bar{o})$ are *not* obtained directly from a partition of not-yet-output tuples in $\mathcal{W}$.

After the anonymization of $\mathcal{R}(\mathcal{W}, e\bar{o})$, window $\mathcal{W}$ advances (step 25).

Now the presentation of Algorithm SABRE$_{\mathsf{W}}$ has completed. In the next section we will prove that the algorithm strictly conforms to the defined privacy model requirements, i.e., (ω, t)-closeness, and expiring constraint.

6.4 Formal analysis

The correctness of $\mathsf{SABRE_W}$ depends on the three cases discussed above in its algorithm. *Case 1* is trivial. The proof for *Case 2* has already been given in its description. Therefore, we only need to focus on *Case 3*. In the following we will first propose four fundamental lemmas. Based on them, we present Theorem 6.1, proving that *Case 3* also follows (ω, t)-closeness. After that, we will show that $\mathsf{SABRE_W}$ conforms to both (ω, t)-closeness and expiring constraint in Theorem 6.2.

In Algorithm $\mathsf{SABRE_W}$, each time $\mathcal{I}$ new tuples arrive (step 5), window $\mathcal{W}$ will advance by $\mathcal{I}$ tuples (steps 8, 25). Therefore, as $\mathcal{W}$ continuously advances, a sequence of windows is generated. We use $\mathcal{W}$ to represent each generated window, but, for the sake of clarity, in the following formal proof we will use different notations to represent different windows. For example, we use $\mathcal{W}$ to represent the window at instant $\imath$ and $\mathcal{W}'$ to represent the window at instant $\imath'$. We also note that DIRECT or INDIRECT is applied only when a window is advancing.

Lemma 6.2. *Assume that ω is the window size of input stream, each window slides by $\mathcal{I}$ tuples, and $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer. Then, for two overlapping windows $\mathcal{W}'$ and $\mathcal{W}$, with $\mathcal{W}'$ generated before $\mathcal{W}$, the first $\mathcal{I}$ tuples in $\mathcal{W}$ also fall in $\mathcal{W}'$.*

Proof. Since window size ω is a multiple of advance size $\mathcal{I}$, once two windows overlap, their overlapping area contains at least $\mathcal{I}$ tuples. Furthermore, since $\mathcal{W}'$ is generated before $\mathcal{W}$, the first $\mathcal{I}$ tuples in $\mathcal{W}$ are in the overlapping area, i.e., the first $\mathcal{I}$ tuples in $\mathcal{W}$ also fall in $\mathcal{W}'$. □

Lemma 6.2 shows the overlap relationship between two windows. The next

lemma says that once procedure INDIRECT is applied, that is, *Case 3* occurs, parameter j must be larger than 1.

Lemma 6.3. *Assume that ω is the window size of input stream, each window advances by $\mathcal{I}$ tuples, and $\omega = j \cdot \mathcal{I}$, where $j > 0$ is an integer. If* INDIRECT *is applied in Algorithm* SABRE$_{\mathsf{W}}$*, then $j > 1$.*

Proof. The proof is by contradiction. For $j = 1$ ($\omega = \mathcal{I}$), whenever a window advances, all its tuples are expiring and will be output. This makes the anonymizations in all the windows independent. Therefore, when a window $\mathcal{W}$ advances, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$. Hence the precondition for INDIRECT to be applied does not hold. □

The next two lemmas clarify the sequential relationship between the two procedures —DIRECT and INDIRECT. Lemma 6.4 indicates that one application of INDIRECT cannot be followed immediately by another application of INDIRECT. There must be at least one application of DIRECT between them. Lemma 6.5 says that there must exist at least one DIRECT before any INDIRECT.

Lemma 6.4. *Given a window $\mathcal{W}$, assume that* INDIRECT *is applied when it advances at instant $\imath$. Suppose that at instant $\imath'$, $\imath' < \imath$, there is an application of* DIRECT *and no further application of* DIRECT *between $\imath'$ and $\imath$. Then there is no application of* INDIRECT *between $\imath'$ and $\imath$ either.*

Proof. The advance of $\mathcal{W}$ incurs the call of INDIRECT, so $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$ at instant $\imath$ (the preconditions of *Case 3*). Assume that there is AI, an application of INDIRECT, between $\imath'$ and $\imath$. Let $\mathcal{W}'$ be the window, whose advance at instant $\imath'$ incurs the application of DIRECT. Suppose that DIRECT partitions $\mathcal{R}(\mathcal{W}', \bar{o})$ (not-yet-output tuples of $\mathcal{W}'$ at instant $\imath'$) into $S_{\mathcal{G}}$, a list of

ECs. AI releases those and only those ECs that are in $S_{\mathcal{G}}$ to the output stream (steps 22-24). We make two points: First, all tuples in $\mathcal{W}'$ have been output after AI; Second, AI outputs tuples only in $\mathcal{W}'$. There are two possibilities for an overlapping relationship between $\mathcal{W}'$ and $\mathcal{W}$. *Possibility 1: $\mathcal{W}'$ overlaps with $\mathcal{W}$.* By Lemma 6.2, $\mathcal{R}(\mathcal{W}, e\bar{o})$, a subset of the first $\mathcal{I}$ tuples of $\mathcal{W}$, falls in $\mathcal{W}'$. All the tuples in $\mathcal{W}'$ are already output after AI (point 1), so $|\mathcal{R}(\mathcal{W}, e\bar{o})| = 0$ at instant $\imath$. This conclusion contradicts the fact that $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$. *Possibility 2: $\mathcal{W}'$ does not overlap with $\mathcal{W}$.* In $\mathsf{SABRE_W}$ only DIRECT and INDIRECT output tuples. During the period between $\imath'$ and $\imath$ there is no further application of DIRECT, hence only AI outputs tuples in this period. Since AI outputs tuples only in $\mathcal{W}'$ (point 2), none of the tuples in $\mathcal{W}$ has been output when $\mathcal{W}$ begins to slide. Therefore, at instant $\imath$, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$; this conclusion contradicts the fact that $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. In conclusion, under the assumption that there is an application of INDIRECT between $\imath'$ and $\imath$, none of the two possibilities is borne. By reductio ad absurdum, we conclude that there is no application of INDIRECT between $\imath'$ and $\imath$. □

Lemma 6.5. *Given a window $\mathcal{W}$, assume that* INDIRECT *is applied when it advances at instant $\imath$. Then there is at least one application of* DIRECT *before instant $\imath$.*

Proof. We prove this also by contradiction. Assume that there is no application of DIRECT before instant $\imath$. Let the list of generated windows be $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_i, \mathcal{W}_{i+1}, \ldots$. When $\mathcal{W}_1$ begins to advance, all its tuples are not-yet-output (i.e., $\mathcal{R}(\mathcal{W}_1, \bar{o}) = \mathcal{R}(\mathcal{W}_1)$). Thus, DIRECT is applied. Therefore, for any window $\mathcal{W}_i$ ($i > 1$), if $\mathcal{W}_i$ advances at instant $\imath_i$, there is at least one application of DIRECT before instant $\imath_i$. Since there is no application of DIRECT before instant $\imath$ at the advancing moment of $\mathcal{W}$, then $\mathcal{W}$ must be $\mathcal{W}_1$. However, this conclusion

contradicts the precondition that INDIRECT is applied when $\mathcal{W}$ slides. By contradiction, the assumption is incorrect, and there is at least one application of DIRECT before instant $\imath$. □

Putting the above four lemmas together, the following theorem proves that *Case 3* (i.e., procedure INDIRECT), anonymizes data strictly complying with the principle of (ω, t)-closeness.

Theorem 6.1. *In Algorithm* SABRE$_{\mathsf{W}}$*, given a window $\mathcal{W}$, assume that* INDIRECT *is applied when it advances at instant $\imath$. Let $\mathcal{R}(\mathcal{W}, e\bar{o})$ be the expiring and not-yet-output tuples in $\mathcal{W}$ at instant $\imath$. Then each tuple in $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $S_{\mathcal{G}}$.*

Proof. The advance of $\mathcal{W}$ incurs the call of INDIRECT, so $|\mathcal{R}(\mathcal{W}, e\bar{o})| > 0$ and $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$ at instant $\imath$ (the preconditions of *Case 3*). Assume that LD, the last application of DIRECT before instant $\imath$, occurs at instant $\imath'$, $\imath' < \imath$ (there is no further application of DIRECT between $\imath'$ and $\imath$). By Lemma 6.5, it follows that LD exists. Let $\mathcal{W}'$ be the window whose advance at instant $\imath'$ incurs LD. Since $S_{\mathcal{G}}$ records the list of ECs generated by the last application of DIRECT, we know that LD partitions $\mathcal{R}(\mathcal{W}', \bar{o})$ (not-yet-output tuples of $\mathcal{W}'$ at instant $\imath'$) into $S_{\mathcal{G}}$, a list of ECs (step 15).

Assume that $\mathcal{W}'$ does not overlap with $\mathcal{W}$. During the period between $\imath'$ and $\imath$, there is neither an application of INDIRECT (Lemma 6.4) nor an application of DIRECT. Still, only DIRECT and INDIRECT output tuples. In effect, there is no tuple output between $\imath'$ and $\imath$. In addition, at instant $\imath'$ only ECs containing tuples in $\mathcal{W}'$ have been released to the output stream (steps 16-19). We conclude that none of tuples in $\mathcal{W}$ has been output when $\mathcal{W}$ slides. So at instant $\imath$, $\mathcal{R}(\mathcal{W}, \bar{o}) = \mathcal{R}(\mathcal{W})$ and $\mathcal{D}_{\mathcal{W}\bar{o}} = 0$, which contradicts the fact that $\mathcal{D}_{\mathcal{W}\bar{o}} \geq t$. By contradiction, it follows that $\mathcal{W}'$ overlaps with $\mathcal{W}$. By Lemma 6.2 we know that $\mathcal{R}(\mathcal{W}, e\bar{o})$,

a subset of the first $\mathcal{I}$ tuples in $\mathcal{W}$, falls in $\mathcal{W}'$, i.e., $\mathcal{R}(\mathcal{W}, e\bar{o}) \subset \mathcal{R}(\mathcal{W}', \bar{o})$. Therefore, each tuple in $\mathcal{R}(\mathcal{W}, e\bar{o})$ is contained in one EC of $S_{\mathcal{G}}$. In addition, each EC of $S_{\mathcal{G}}$ is *t*-close to $\mathcal{W}'$ with respect to its $\mathcal{SA}$ distribution. □

So far, the closeness proof for *Case 3* has completed. The next theorem based on Theorem 6.1 will show that $\mathsf{SABRE_W}$ follows the privacy model defined in Section 6.2.

Theorem 6.2. *Algorithm* $\mathsf{SABRE_W}$ *follows the* (ω, t)*-closeness and expiring constraint.*

Proof. Given an input stream $\mathcal{S}_{in}$, $\mathsf{SABRE_W}$ anonymizes it into an output stream $\mathcal{S}_{out}$ by the three cases we have presented. We discuss these three cases one by one.

Case 1 (steps 7-9). Since all the expiring tuples are already output, the expiring constraint is met. This case does not anonymize and output any tuple, so the satisfaction of (ω, t)-closeness depends on the remaining two cases.

Case 2 (steps 14-19). Given an advancing window $\mathcal{W}$ at instant $\imath$, for each expiring and not-yet-output tuple, an EC is formed and released to $\mathcal{S}_{out}$. As each expiring tuple is output, the expiring constraint is met. Besides, since each output EC is *t*-close to $\mathcal{W}$, (ω, t)-closeness is satisfied.

Case 3 (steps 21-24). Given an advancing window $\mathcal{W}$ at instant $\imath$, by Theorem 6.1, each expiring and not-yet-output tuple is contained in an EC formed by tuples from another window $\mathcal{W}'$, generated before $\mathcal{W}$. All ECs containing expiring tuples are output, hence the expiring constraint is met. Each output EC is *t*-close to $\mathcal{W}'$, so this case attains (ω, t)-closeness as well.

In conclusion, $\mathsf{SABRE_W}$ abides to both (ω, t)-closeness and the expiring constraint. □

6.5 Experiment evaluation

In this section we evaluate the performance of SABRE$_W$, our window-based t-closeness scheme, on streaming data. We reuse Algorithm SABRE-AK as a building block to anonymize a window of tuples (step 15 of Algorithm SABRE$_W$). In a similar way, we extend tIncognito [52] and tMondrian [53] to the context of data streams, thus, obtain tIncognito$_W$ and tMondrian$_W$, respectively. In Section 3.5 we have already made a comparison among SABRE-AK, tIncognito, and tMondrian, and found that SABRE-AK is most efficient with respect to both information quality and elapsed time. In the experiments carried out in this section, we discover the similar trends for the three extended window-based t-closeness schemes, that is, SABRE$_W$ is the most effective and efficient, followed by tMondrian$_W$, and tIncognito$_W$. Therefore, in the following we will briefly present the experiment results, by only comparing the three algorithms with regard to *General Information Loss metrics* (see Section 2.3) and elapsed time.

Attribute	Cardinality	Type
age	79	numerical (4)
sex	2	categorical (1)
education	17	numerical (4)
marital status	6	categorical (2)
race	9	categorical (1)
work class	10	categorical (3)
birth place	83	categorical (2)
salary	50	numerical

Table 6.2: The CENSUS dataset

The prototypes were implemented in Java and the experiments were run on a core-2 duo 2.33GHz CPU machine, with 4GB RAM, running windows XP. We simulate the data stream by the CENSUS dataset [3], which contains

500,000 tuples, and has 8 attributes as shown in Table 6.2; the value following the type is the height of the corresponding attribute hierarchy. For instance, attribute *marital status* is categorical and has a hierarchy of height 2. The first 7 attributes are used by default as the QI, and the last one (i.e., *salary*) as the sensitive attribute. To provide defence against the attack arising when all the tuples in an EC belong to a single person (Section 6.2), we combine (ω, t)-closeness and k_s-anonymity. Hence, the anonymized data are protected against both $\mathcal{SA}$ disclosure and identity disclosure. By default we set k to be 6 (i.e., the number of distinct persons referred to by all the tuples in an EC is at least 6). The closeness threshold *t* is a variable, and is set to 0.35 by default.

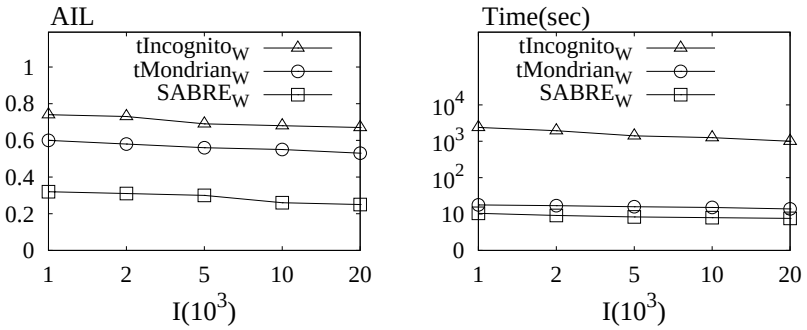


Figure 6.4: Effect of varying $\mathcal{I}$

We first set window size ω to 20,000, and study the effect of the value of $\mathcal{I}$, the time interval of window advance. Figure 6.4 (a) illustrates the result on information loss. Algorithms tIncognito$_W$ and tMondrian$_W$ are extended from k-anonymity algorithms, hence lacking a customization towards the features of (ω, t)-closeness. On the contrary, SABRE$_W$ has been specially designed for (ω, t)-closeness. Therefore, as expected, SABRE$_W$ achieves the best information quality, followed by tMondrian$_W$ and tIncognito$_W$. Figure 6.4 (b) is the result on the elapsed time. Let $|S|$ ($|S| >> \omega$) be the size of stream, then the number

of times to advance the window is $\frac{|S|-\omega}{\mathcal{I}} \approx \frac{|S|}{\mathcal{I}}$. Every time the window advances there is one occurrence of anonymization. So when $\mathcal{I}$ approaches ω, the anonymization process is invoked fewer times, hence the elapsed time decreases. Still, $\mathsf{SABRE_W}$ is the fastest method, and $\mathsf{SABRE_W}$ and $\mathsf{tMondrian_W}$ are two orders of magnitude faster than $\mathsf{tIncognito_W}$.

In our next experiment, we set $\mathcal{I}$ at 5000, and investigate the effect of the window size ω. As the value of ω grows, more tuples are involved in each occurrence of anonymization (step 15 of algorithm $\mathsf{SABRE_W}$). Therefore, the possibility of grouping tuples with similar QI values is improved, and the information loss of the output ECs is reduced. Figure 6.5 (a) presents the results consistent with our expectation; $\mathsf{SABRE_W}$ retains its superiority, outperforming $\mathsf{tIncognito_W}$ and $\mathsf{tMondrian_W}$ by a wide margin in information quality. Figure 6.5 (b) shows the time efficiency of the three algorithms. After $\mathcal{I}$ is fixed, the number of times of window advance will be closely approximate to $\frac{|S|}{\mathcal{I}}$, thus also fixed. However, the increase of window size ω also indicates an increment of the number of tuples participating in an anonymization. Therefore, the total elapsed time grows as a result of the increase of the time spent in each anonymization triggered by a window sliding. Similarly, $\mathsf{SABRE_W}$ and, to a certain extent, $\mathsf{tMondrian_W}$ perform much better than $\mathsf{tIncognito_W}$ in terms of time efficiency.

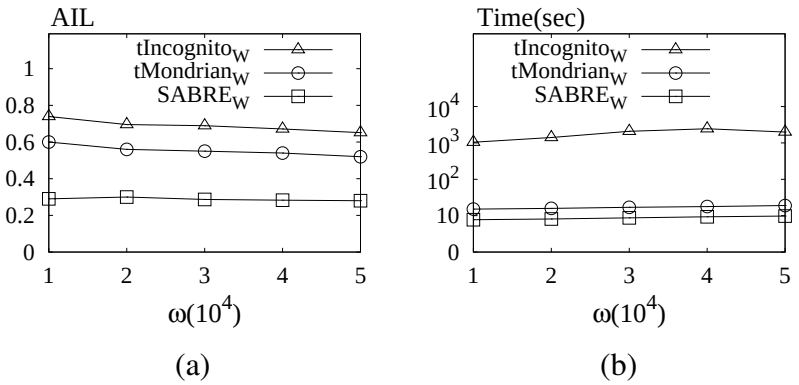
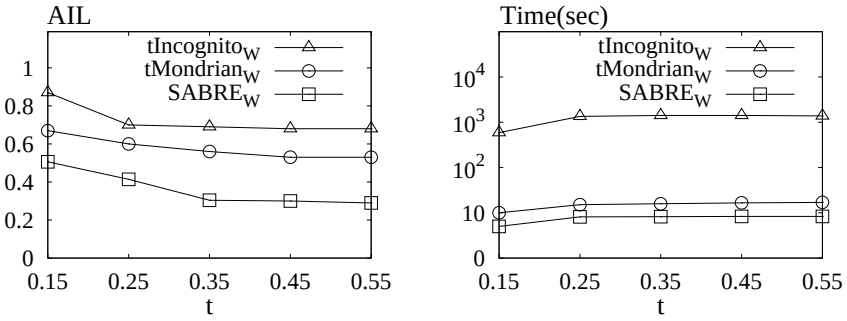


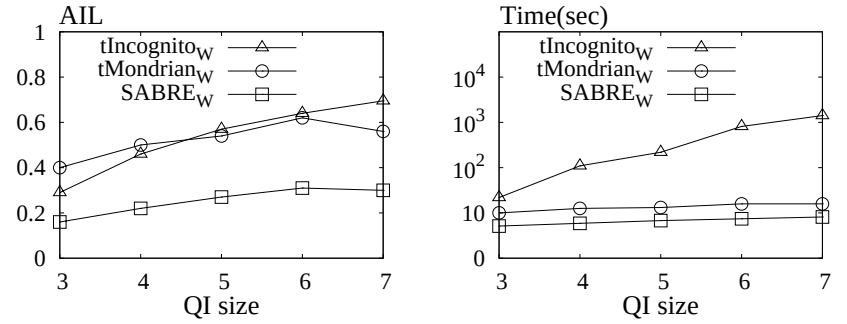
Figure 6.5: Effect of varying window size



(a) (b)

Figure 6.6: Effect of varying closeness threshold

Figure 6.6 shows the results of varying closeness threshold t from 0.15 to 0.55, where window size ω is set to 20,000 and window advance size $\mathcal{I}$ is set to 5,000. As expected, when the t value increases, the constraint on the similarity between the $\mathcal{SA}$ distribution in any EC and that in a window containing the EC is relaxed, hence the information quality of anonymized data of all the approaches is improved.



(a) (b)

Figure 6.7: Effect of varying QI

Finally, we vary the QI size from 3 to 7 to examine its effect on anonymized data. Again window size is set to 20,000 and window advance size is fixed to 5,000. Figure 6.7 (a) is the information loss of the anonymized streaming data by the three involved approaches. As the QI dimensionality grows, the data

is more sparse in the QI space[2], thus the formed ECs are more likely to have bigger minimum bounding boxes. As a result, the information quality degrades as a function of the QI size. Figure 6.7 (b) records the time complexity. When the QI dimensionality increases, the number of generalization hierarchies[3] to be searched by tIncognito$_W$ grows exponentially. So its elapsed time increases exponentially as displayed in the graph. Likewise, the other two methods, i.e., SABRE$_W$ and tMondrian, also spend more time in anonymizing the data stream as QI size grows. However, different from tIncognito$_W$, their increased cost is mainly due to the calculation of the minimum bounding boxes with higher dimensionality. In addition, they are more scalable.

6.6 A discussion on the extension to β-likeness

We have extended SABRE to the context of data streams in the above. The strategy to achieve this goal is general, and can be reused to adapt other methods, which consider global $\mathcal{SA}$ distribution as background knowledge, to streaming data. BUREL, the β-likeness approach in Chapter 4, falls in the category of such methods, and its extension towards streams can be done as follows. At the beginning, we will constrain the β-likeness requirements within windows, just as we define t-closeness in windows in Section 6.2. After that, we can also propose a tailored scheme for the window-based β-likeness principle. The steps of the scheme will be similar to those of Algorithm SABRE$_W$. However, instead of checking the closeness (steps 12, 14 of Algorithm SABRE$_W$), we will measure the likeness between the ECs and the windows containing them. In

[2]One QI attribute is one dimension in the QI space, and each streaming tuple is modelled as one point in the space.

[3]A generalization hierarchy can be a subtree of the domain hierarchy of a related QI attribute or the combined tree of two or more generalization hierarchies [48].

addition, instead of using SABRE to anonymize a window of tuples (step 15 of Algorithm $\mathsf{SABRE_W}$), we will reuse BUREL as a building block to process the tuples staying in a window, so that their output conforms to the window-based β-likeness requirements. As such, the solid theory foundation to prove the correctness of the tailored scheme will also be like that in Section 6.4.

6.7 Summary

In this chapter we presented a t-closeness-resembling privacy model in the context of streaming data. We have proposed an algorithm customized for the model, together with a solid theory foundation proving the soundness of the algorithm. Experimental evaluation has been conducted; the extensive results show that our tailored algorithm outperforms those approaches extended from existing k-anonymity methods, with regard to both information quality and time efficiency.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this chapter we will first summarize the contributions of our work, then we will discuss some research topics as future work.

7.1 Thesis summary

This thesis concentrates on the anonymization of microdata, with two goals in mind: protecting individuals from being linked to specific tuples and/or sensitive values, and at the same time, maximizing the utility of released data. To achieve such targets, we first proposed SABRE, a sophisticated framework that achieves t-closeness in an elegant and efficient manner. A solid theory foundation has been provided to ensure that the two particular phases of SABRE, namely *bucketization* and *redistribution*, as a whole strictly follow t-closeness constraints. We have shown the applicability of our scheme on both categorical and numerical attributes. The extensive experimental results have demonstrated that our two SABRE instantiations, SABRE-AK and SABRE-KNN, clearly outperform previous schemes with respect to information quality, while SABRE-AK also improves over them in terms of elapsed time. In conclusion, SABRE provides the best known resolution of the tradeoff between privacy, information quality, and computational efficiency, as far as t-closeness guarantee is concerned.

So far, all privacy preserving schemes that guarantee t-closeness [52, 53, 63]

including SABRE do not consider an adversary's information gain on each single SA value. Therefore, even though t-closeness is an improved model beyond k-anonymity [71] and ℓ-diversity [57], it still fails to translate t, the threshold, into a human-understandable privacy guarantee. To cope with this limitation, we proposed β-likeness, a robust privacy model for microdata anonymization. It requires that the relative difference of *each* $\mathcal{SA}$ value frequency between an EC and the whole table should not be more than a threshold β, thus precisely interpreting the parameter to a comprehensible privacy guarantee. Furthermore, we designed an algorithm BUREL, tailored for β-likeness model. A comparison with t-closeness schemes demonstrates that BUREL provides effective privacy guarantees in a way that state-of-the-art t-closeness schemes cannot, even when set to achieve the same information accuracy or privacy measured by the criterion of t-closeness. In the experiments, we have also shown that BUREL is more effective and efficient than a β-likeness algorithm extended from k-anonymization method [49].

There is a need of data publication for both static and streaming data. However, most of the developed privacy techniques, including SABRE and BUREL, are designed for static data sets. They are inapplicable to streaming data. Therefore, we proposed CASTLE, a cluster-based framework that continuously k-anonymizes arriving tuples. CASTLE ensures the freshness of released data, by imposing a delay constraint, so that the maximum delay between any tuple's input and its output is smaller than a threshold δ. Other features of CASTLE include its adaptivity to data distributions, and its cluster reuse strategy to improve the information quality without compromising security. The conducted performance evaluation has shown that CASTLE is efficient and effective with regard to the quality of the output data. We have further demonstrated that CASTLE

can be extended to support ℓ-diversity in a straightforward way.

Besides k-anonymity and ℓ-diversity, we have also revised t-closeness and applied it on data streams. We proposed (ω, t)-closeness, which requires that for any output EC, there exists a window, which has a size of ω and contains the EC, so that the difference of $\mathcal{SA}$ distribution between the EC and the window is no more than a threshold t. In this way, we restrict the closeness constraints within each window instead of the whole dataset, following the conventional wisdom that streaming tuples are processed in windows. Furthermore, an algorithm customized for (ω, t)-closeness has been introduced; its soundness is well supported with a solid theory foundation. The experimental study has shown that our tailored scheme outperforms methods extended from algorithms developed for k-anonymity model, in terms of both information quality and elapsed time.

7.2 Future work

In this section we bring forward three topics on the agenda of our future research.

7.2.1 Access control over data streams

Privacy-protection data publication treats each potential recipient (i.e., the user of the data) equally. However, there are applications, such as battlefield, network monitoring, and stock market, where users are classified into roles and each role is permitted to see only a part of the data based on pre-defined policies. For example, stock prices are delivered to paying clients based on their subscriptions. The concept of *role base access control* [68] was introduced with such security requirements in mind. We have proposed a general framework to

protect against unauthorized access to data streams [29][1]. Given a submitted query, we rewrite it according to its related role based access control policies in such a way that only authorized tuples/attributes will be returned to the user. In addition, we have implemented the framework in StreamBase (i.e., a popular commercial data stream engine), and demonstrated it [25]. The extension of our framework includes but not limited to the following directions: the optimization of rewritten queries, updates of queries and access control policies, and the support of sharing a common sub-query among users. It is important to remark that our access control model is discretionary, just like most models adopted in commercial data management systems. As such, it leaves the responsibility of correctly defining control policies to the security administrator. As a result, potential conflicts among policies exist, thus providing inference channels for the attackers. Therefore, another interesting direction for future work is investigating how our framework can be complemented by inference control techniques [20, 64].

7.2.2 Anonymization of transaction dataset

Transaction data have a wide range of applications, such as association rule mining [13, 14], query expansion [35], and predicting user behavior [7]. However, the publication of such data may put the privacy of individuals at risk—an attacker with the partial knowledge of transactions may associate individuals with sensitive information. As a result, a careful anonymization of the data before their release is indispensable. Transaction data are *set-valued*; each entry is a set of items, e.g., purchased items, query items, user preferences, chosen from a universal domain. Consequently, anonymization methods developed on

[1] A paper invited and accepted by TISSEC.

microdata, which has a fixed schema, cannot be applied directly on them. We have proposed ρ-uncertainty [28], an inference-proof privacy principle. Given a transaction dataset $\mathcal{DB}$, for any transaction $x \in \mathcal{DB}$, any subset of items $\chi \subset x$, and any sensitive item $\alpha \notin \chi$, ρ-uncertainty requires that the confidence of the sensitive association rule[2] $\chi \rightarrow \alpha$ be at most ρ. Obviously, ρ-uncertainty limits the sensitive inference arising from prior knowledge χ. We have designed an algorithm, which solves the problem of ρ-uncertainty in a non-trivial way by combining both generalization and suppression. Still, rendering a dataset ρ-uncertain is a challenging task, due to the huge amount of sensitive association rules existing in the data. Till now our algorithm can process only small transactions. Therefore, a new approach, which can process longer transactions and better preserve information, will be an item on our research agenda. Furthermore, we are interested in applying ρ-uncertainty to the cognate problem of anonymizing *functional dependencies* in a relational dataset.

7.2.3 Algorithm-based attacks

Like most other privacy approaches, the methods in this thesis assume *random worlds model* [18], i.e., given an anonymized dataset, its possible inputs can be many, and an attacker treats each of these "possible worlds" as equally likely. As an example, suppose that tuple x appears in an anonymized dataset $\mathcal{DB}'$. To determine the probability that $x.\mathcal{SA}$ is diabetes, an attacker will examine all the input instances, each with an output equal to DB', and compute the fractions of those inputs consistent with $x.\mathcal{SA} = diabetes$. Without further information, an attacker can only treat each input instance equally. However, using the knowledge of specific anonymization algorithms, an attacker can eliminate some input

[2] An association rule is sensitive, if its consequent contains at least one sensitive item.

instances, and his/her belief in certain event can be raised, thus some desired privacy requirements may possibly be broken. *Minimality attack* [77] is one case of such attacks; it is based on the observation that most anonymization methods try to minimize information loss and such an attempt enables the attack. Recently, Cormode et al. [33] have determined the scope of the effectiveness of this attack. Therefore, another interesting topic for future research can be examining the internal workings of our proposed mechanisms with regard to the analysis in [33], and then enhancing them to thwart the minimality attack.

REFERENCES

[1] http://ddm.cs.sfu.ca/.

[2] http://www.csee.usf.edu/~christen.

[3] http://www.ipums.org.

[4] www.ics.uci.edu/~learn/mlsummary.html.

[5] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik. The design of the borealis stream processing engine. In *Proc. of CIDR*, pages 277–289, 2005.

[6] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. B. Zdonik. Aurora: a new model and architecture for data stream management. *VLDB Journal*, 12(2):120–139, 2003.

[7] E. Adar, D. S. Weld, B. N. Bershad, and S. S. Gribble. Why we search: visualizing and predicting user behavior. In *Proc. of WWW*, pages 161–170, 2007.

[8] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. of VLDB*, pages 901–909, 2005.

[9] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. of VLDB*, pages 81–92, 2003.

[10] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *Proc. of EDBT*, pages 183–199, 2004.

[11] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proc. of PODS*, pages 153–162, 2006.

[12] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy,

D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. of ICDT*, pages 246–258, 2005.

[13] R. Agrawal, T. Imieliński, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of SIGMOD*, pages 207–216, 1993.

[14] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of VLDB*, pages 487–499, 1994.

[15] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. Stream: The stanford stream data manager. In *Proc. of SIGMOD*, 2003.

[16] M. Atzori. Weak k-anonymity: A low-distortion model for protecting privacy. In *Proc. of Information Security, 9th International Conference*, pages 60–71, 2006.

[17] B. Babcock, S. Babu, M. Datar, R. Motwani, and D. Thomas. Operator scheduling in data stream systems. *VLDB Journal*, 13(4):333–353, 2004.

[18] F. Bacchus, A. J. Grove, D. Koller, and J. Y. Halpern. From statistics to beliefs. In *Proc. of AAAI*, pages 602–608, 1992.

[19] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proc. of ICDE*, pages 217–228, 2005.

[20] J. Biskup and J.-H. Lochner. Enforcing confidentiality in relational databases by reducing inference control to access control. In *Information Security, 10th International conference*, pages 407–422, 2007.

[21] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proc. of KDD*, pages 70–79, 2008.

[22] Y. Bu, A. W.-C. Fu, R. C.-W. Wong, L. Chen, and J. Li. Privacy

preserving serial data publishing by role composition. *PVLDB*, 1(1):845–856, 2008.

[23] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *Secure Data Management*, pages 48–63, 2006.

[24] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan. Castle: A delay-constrained scheme for ks-anonymizing data streams. In *Proc. of ICDE*, pages 1376–1378, 2008.

[25] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan. Acstream: Enforcing access control over data streams. In *Proc. of ICDE*, pages 1495–1498, 2009.

[26] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan. Castle: Continuously anonymizing data streams. *Accepted by IEEE Transactions on Dependable and Secure Computing*, 2009.

[27] J. Cao, P. Karras, P. Kalnis, and K.-L. Tan. Sabre: A sensitive attribute bucketization and redistribution framework for t-closeness. *Accepted by VLDB Journal*, 2010.

[28] J. Cao, P. Karras, C. Raïssi, and K.-L. Tan. ρ-uncertainty: Inference-proof transaction anonymization. *PVLDB*, 3(1):1033–1044, 2010.

[29] B. Carminati, E. Ferrari, J. Cao, and K.-L. Tan. A framework to enforce access control over data streams. *ACM Transactions on Information & System Security (TISSEC)*, 13(3), 2010.

[30] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah. Telegraphcq: Continuous dataflow processing for an uncertain world. In *Proc. of CIDR*, 2003.

[31] B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala.

Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2(1-2):1–167, 2009.

[32] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. Niagaracq: a scalable continuous query system for internet databases. In *Proc. of SIGMOD*, pages 379–390, 2000.

[33] G. Cormode, N. Li, T. Li, and D. Srivastava. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *PVLDB*, 3(1):1045–1056, 2010.

[34] B. Cui, B. C. Ooi, J. Su, and K.-L. Tan. Indexing high-dimensional data for efficient in-memory similarity search. *TKDE*, 17(3):339–353, 2005.

[35] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proc. of WWW*, pages 325–332, 2002.

[36] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of KDD*, pages 71–80, 2000.

[37] A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of PODS*, pages 211–222, 2003.

[38] B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Anonymity for continuous data publishing. In *Proc. of EDBT*, pages 264–275, 2008.

[39] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *Proc. of ICDE*, pages 205–216, 2005.

[40] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *Proc. of VLDB*, pages 758–769, 2007.

[41] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints.

TODS, 34(2):1–47, 2009.

[42] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proc. of FOCS*, pages 359–366, 2000.

[43] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of KDD*, pages 279–288, 2002.

[44] P. Karras. Multiplicative synopses for relative-error metrics. In *Proc. of EDBT*, pages 756–767, 2009.

[45] P. Kooiman, L. Willenborg, and J. Gouweleeuw. Pram: A method for disclosure limitation for microdata. *Research paper / Statistics Netherlands*, 1997.

[46] N. Koudas, D. Srivastava, T. Yu, and Q. Zhang. Distribution based microdata anonymization. *PVLDB*, 2(1):958–969, 2009.

[47] Y.-N. Law, H. Wang, and C. Zaniolo. Query languages and data models for database sequences and data streams. In *Proc. of VLDB*, pages 492–503, 2004.

[48] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of SIGMOD*, pages 49–60, 2005.

[49] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, number 25, 2006.

[50] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proc. of KDD*, pages 277–286, 2006.

[51] J. Li, Y. Tao, and X. Xiao. Preservation of proximity privacy in publishing numerical sensitive data. In *Proc. of SIGMOD*, pages 473–486, 2008.

[52] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *Proc. of ICDE*, pages 106–115, 2007.

[53] N. Li, T. Li, and S. Venkatasubramanian. Closeness: A new privacy

measure for data publishing. *IEEE Trans. Knowl. Data Eng.*, 22(7):943–956, 2010.

[54] T. Li and N. Li. On the tradeoff between privacy and utility in data publishing. In *Proc. of KDD*, pages 517–526, 2009.

[55] L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. *TKDE*, 11(4):610–628, 1999.

[56] C. Luo, H. Thakkar, H. Wang, and C. Zaniolo. A native extension of sql for mining data streams. In *Proc. of SIGMOD*, pages 873–875, 2005.

[57] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE*, number 24, 2006.

[58] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of PODS*, pages 223–228, 2004.

[59] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Trans. Knowl. Data Eng.*, 13(1):124–141, 2001.

[60] M. E. Nergiz, M. Atzori, and C. Clifton. Hiding the presence of individuals from shared databases. In *Proc. of SIGMOD*, pages 665–676, 2007.

[61] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining k-anonymity against incremental updates. In *Proc. of SSDBM*, 2007.

[62] L. Qiu, Y. Li, and X. Wu. Protecting business intelligence and customer privacy while outsourcing data mining tasks. *Knowl. Inf. Syst.*, 17(1):99–120, 2008.

[63] D. Rebollo-Monedero, J. Forné, and J. Domingo-Ferrer. From t-closeness-like privacy to postrandomization via information theory.

IEEE Trans. Knowl. Data Eng., 22(11):1623–1636, 2010.

[64] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proc. of SIGMOD*, pages 551–562, 2004.

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

[66] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.

[67] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of PODS*, 1998.

[68] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[69] U. Schreier, H. Pirahesh, R. Agrawal, and C. Mohan. Alert: An architecture for transforming a passive dbms into an active dbms. In *Proc. of VLDB*, pages 469–478, 1991.

[70] M. Sullivan. Tribeca: A stream database manager for network traffic analysis. In *Proc. of VLDB*, page 594, 1996.

[71] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[72] Y. Tao, X. Xiao, J. Li, and D. Zhang. On anti-corruption privacy preserving publication. In *Proc. of ICDE*, pages 725–734, 2008.

[73] T. M. Truta and A. Campan. k-anonymization incremental maintenance and optimization techniques. In *ACM Symposium on Applied Computing*, pages 380–387, 2007.

[74] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proc. of KDD*, pages 414–423, 2006.

[75] K. Wang, Y. Xu, A. W.-C. Fu, and R. C.-W. Wong. Ff-anonymity: When quasi-identifiers are missing. In *Proc. of ICDE*, pages 1136–1139, 2009.

[76] R. C.-W. Wong and A. W.-C. Fu. *Privacy-Preserving Data Publishing: An Overview*. Morgan & Claypool Publishers, 2010.

[77] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proc. of VLDB*, pages 543–554, 2007.

[78] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *KDD*, pages 754–759, 2006.

[79] W. K. Wong, N. Mamoulis, and D. W.-L. Cheung. Non-homogeneous generalization in privacy preserving data publishing. In *Proc. of SIGMOD*, pages 747–758, 2010.

[80] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of VLDB*, pages 139–150, 2006.

[81] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proc. of SIGMOD*, pages 689–700, 2007.

[82] X. Xiao and Y. Tao. Dynamic anonymization: accurate statistical analysis with privacy preservation. In *Proc. of SIGMOD*, pages 107–120, 2008.

[83] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *Proc. of KDD*, pages 785–790, 2006.

[84] S. B. Zdonik, M. Stonebraker, M. Cherniack, U. Çetintemel,

M. Balazinska, and H. Balakrishnan. The aurora and medusa projects. *IEEE Data Eng. Bull.*, 26(1):3–10, 2003.

[85] P. Zhang, X. Zhu, and Y. Shi. Categorizing and mining concept drifting data streams. In *Proc. of KDD*, pages 812–820, 2008.

[86] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *Proc. of ICDE*, pages 116–125, 2007.

[87] Y. Zhu, E. A. Rundensteiner, and G. T. Heineman. Dynamic plan migration for continuous queries over data streams. In *Proc. of SIGMOD*, pages 431–442, 2004.