PRIVACY PRESERVING PUBLISHING OF HIERARCHICAL DATA

by İsmet Özalp

Submitted to the Graduate School of Engineering and
Natural Sciences
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Sabancı University
August, 2017

PRIVACY PRESERVING PUBLISHING OF HIERARCHICAL DATA

APPROVED BY:

Prof. Yücel Saygın (Thesis Supervisor)	[illegible]
Prof. Erkay Savaş	[illegible]
Prof. Osman Uğur Sezerman	[illegible]
Assoc. Prof. Hüsnü Yenigün	[illegible]
Asst. Prof. Ali İnan	[illegible]

DATE OF APPROVAL: 02.08.2017

Dedicated to my parents and my brother

for their endless love, support and encouragement

Acknowledgments

First of all I would like to convey my deepest appreciation to my thesis supervisor Prof. Yücel Saygın. His guidance and wisdom was always there to help me navigate throughout my Ph.D. journey and this thesis.

Also I would like to express sincere thanks to my thesis co-supervisor Assoc. Prof. Mehmet Ercan Nergiz for his continuous support and encouragement. Without his mentoring and guidance this research would not be possible.

Furthermore, I especially like to thank my thesis committee Prof. Erkay Savaş, Prof. Uğur Sezarman, Assoc. Prof. Hüsnü Yenigün and Asst. Prof. Ali İnan for their comments and inputs.

In addition, I want to thank my dear collages Dr. Emre Kaplan and Mehmet Emre Gürsoy for their invaluable inputs and discussions during my research. Also I want to thank Mr. Bülent Dandin, Mehmet Önder and Özgür Aydınlı for their understanding and support through out my thesis processes.

PRIVACY PRESERVING PUBLISHING OF HIERARCHICAL DATA

İsmet Özalp
Computer Science and Engineering

Ph.D. Thesis, 2017

Thesis Supervisor: Prof. Yücel Saygın

Thesis Co-supervisor: Assoc. Prof. Mehmet Ercan Nergiz

Keywords: privacy, data publishing, hierachical data, k-anonimity, ℓ-diversity, anatomization

Abstract

Many applications today rely on storage and management of semi-structured information, e.g., XML databases and document-oriented databases. This data often has to be shared with untrusted third parties, which makes individuals' privacy a fundamental problem. In this thesis, we propose anonymization techniques for privacy preserving publishing of hierarchical data. We show that the problem of anonymizing hierarchical data poses unique challenges that cannot be readily solved by existing mechanisms. We addressed these challenges by utilizing two major privacy techniques; generalization and anatomization.

Data generalization encapsulates data by mapping nearly low-level values (e.g., influenza) to higher-level concepts (e.g., respiratory system diseases). Using generalizations and suppression of data values, we revised two standards for privacy protection: k-anonymity that hides individuals within groups of k members and ℓ-diversity that bounds the probability of linking sensitive values with individuals. We then apply these standards to hierarchical data and present utility-aware algorithms that enforce the standards. To evaluate our algorithms and their heuristics, we experiment on synthetic and real datasets obtained from two universities. Our experiments show that we significantly outperform related methods that provide comparable privacy guarantees.

Data anatomization masks the link between identifying attributes and sensitive attributes. This mechanism removes the necessity for generalization and opens up the possibility for higher utility. While this is so, anatomization has not been proposed for hierarchical data where utility is a serious concern due to high dimensionality. In this thesis we show, how one can perform the non-trivial task of defining anatomization in the context of hierarchical data. Moreover, we extend the definition of classical ℓ-diversity and introduce (p,m)-privacy that bounds the probability of being linked to more than m occurrences of any sensitive values by p. Again, in our experiments we have observed that even under stricter privacy conditions our method performs exemplary.

HİYERARŞİK VERİLERDE MAHREMİYETİN KORUNMASI

İsmet Özalp
Bilgisayar Bilimi ve Mühendisliği
Doktora Tezi, 2017
Tez Danışmanı: Prof. Dr. Yücel Saygın
Tez Eş Danışmanı: Doç. Dr. Mehmet Ercan Nergiz

Anahtar Sözcükler: mahremiyet, veri yayınlanması, hiyerarşik veri, k-anonim, ℓ-çeşitlilik, anatomlama

Özet

Günümüzde bir çok uygulama kısmi belirli verilerin saklanması ve yönetimi (XML veritabanları ve belge odaklı veritabanları gibi) üzerine kurulmuştur. Bu veriler çoğu zaman güvenilmeyen üçüncü şahıs ve kurumlarla paylaşılmaktadır. Bu durum bireylerin veri mahremiyetine yönelik temel sorunları da beraberinde getirmektedir. Bu çalışmada, hiyerarşik verilerde kullanılmak üzere geliştirilmiş anonimleştirme teknikleri gösterilmektedir. Ayrıca bu çalışma ile hiyerarşik verilerin anonimleştirilmesi için günümüz tekniklerinin kolaylıkla çözemeyeceği veri mahremiyeti sorunlarına genelleştirme ve anatomlaştırma tekniklerine dayalı yenilikçi çözümler getirilmektedir.

Veri genelleştirmesi, verilerin neredeyse düşük seviye değerlerini (ör: grip) daha yüksek seviye kavramlara (ör: solunum yolu hastalığı) dönüşmesini ihtiva eder. Veri değerlerine genelleme ve silme yapılarak, iki önemli mahremiyet standardı k-anonimleme (fertleri k tane elemanlı gruplara koyarak saklar) ve ℓ-çeşitlilik (bir kişinin, herhangi bir mahrem bilgiyle ilişkilendirilebilme ihtimalini limitler) revize edilmiş ve hiyerarşik verilere uygulanmıştır. Bu standartları destekleyen fayda duyarlı algoritmalar sunulmuştur. Algoritmaların ve buluşsal yöntemlerin değerlendirmesi için iki farklı üniversite veri setiyle, biri sentetik diğeri gerçek veri seti olmak üzere, deneyler yapılmıştır. Deney sonuçlarına göre karşılaştırılabilir gizlilik garantileri sağlayan ilgili yöntemlerden önemli ölçüde daha iyi performans elde edilmiş ve gösterilmiştir.

Veri anatomlaşlaştırması, belirteç verilerle, mahrem veriler arasındakı bağlantıyı maskeler ve genelleme zorunluluğunu ortadan kaldırır. Bu sayede daha yüksek verim sağlamaya imkan tanır. Hiyerarşik verilerde yüksek boyutluluk sebebiyle verim sağlamanın ciddi endişe kaynağı olmasına rağmen anatomlaştırma avantajı hiyerarşik verilerde bu güne kadar önerilmemiştir. Bu tezde, anatomlaştırma işleminin hiyerarşik verilere nasıl uygulanağını tanımlanmış ve gösterilmiştir. Ayrıca klasik ℓ-çeşitlilik yöntemi geliştirilerek yeni bir mahremiyet standardı (p,m)-gizliliği önerilmiştir. (p,m)-gizliliği, m tane herhangi bir mahrem verinin bir kişiyle ilişkilendirilme ihtimalini p ile limitler. Deneyler sonucunda daha zor mahremiyet standartlarında bile örnek teşkil edecek performans sağladığını gözlemlenmektedir.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Today, exabytes of data flows around globe daily. Massive amounts of data created and shared through search engines, social networks, streaming services, business applications, software as a service systems and government branches. Large corporations such as Facebook, Google, IBM, Netflix and Uber are collecting personal data in exchange of their service. The reason behind sharing data can be due to obligation [1] or commercial/public benefit. For instance, National Institutes of Health which is responsible for medical research under U.S. Department of Health and Human Services expects some funded projects to include a plan for sharing research data [2] . Another aspect is these entities may want to share data to a third party like a data analytics company, with purpose of research or create more business value.

However, data in today's world often comes in various complex structures and formats. In particular, hierarchical data has become ubiquitous with the advent of document-oriented databases following the NoSQL trend (e.g., MongoDB) and the popularity of markup languages for richly structured documents and objects (e.g., XML, JSON, YAML). All the ever-increasingly collected data, when combined together pose a threat to privacy. Simple deductive reasoning or sophisticated knowledge discovery techniques may link individuals with sensitive information such as sexual preference, political views, alcohol usage or health condition. Due to such potential risks to individual privacy, many countries have laws enforcing regulations on data sharing and publishing [3] [4].

Due to inherit privacy risks, data owners are required to de-identify personal data before sharing it. This is not a straightforward task. Removing personal identifiers from data, which may seem to be a proper de-identification, is not enough to ensure privacy. It has been shown that even without the personal identifiers, an attacker can still identify a person with great accuracy via joining released data with external sources [5]. Besides, while protecting privacy is paramount, preserving utility is as important. All privacy preserving data publishing techniques' main concern is to balance privacy requirements and amount of information published. They all try to publish as much information as they can while preserving patterns and statistics in the data. So that when anonymized data is published, it will be useful for applying knowledge discovery techniques.

Since the risks of identification have been realized, numerous privacy standards and a variety of methods to enforce these standards have been proposed in the literature. Due to its simplicity, prior research on privacy preserving data publishing addressed tabular data. Even though a considerable portion of today's data is stored and maintained in a hierarchical form, very few existing work [6] address how privacy can be achieved in a multirelational setting. Direct application of classical techniques unfortunately does not satisfy privacy in this setting. Defining and enforcing privacy standards while preserving utility in high dimensional hierarchical data poses a unique challenge for researchers.

In this thesis, we address the aforementioned challenge by presenting hierarchical anonymization techniques. In particular, we used generalization and anatomization.

We motivate privacy-related attacks on hierarchical data using the example in Figure 1.1. This record fits the hierarchical education schema given in Figure 1.2. Student S, born in *1993* and majoring in *Computer Science*, took two courses: *CS201* and *CS306*. For *CS201*, S submitted evaluations for two of his instructors. For *CS306*, S submitted one evaluation and also reported that he bought the *Intro to Databases* book. We say that all of this knowledge are QIs of S. Notice that we write QIs as labels of vertices. Knowing some or all of these QIs, the goal of the adversary is to learn sensitive information about S (e.g., GPA, letter grades S received from the two courses, his evaluation scores etc.). Without anonymization this could be trivial: If there is only one Computer Science student born in 1993 in the database, then the adversary immediately learns the GPA of S (and consequently, every other sensitive value in S's data record). Our anonymization strategy is to create equivalence classes of size $\geq \ell$ for an input parameter ℓ, such that even though the adversary knows all of S's QIs, he can only link S to a group of ℓ records. Furthermore, using ℓ-diversity, we ensure that sensitive values for each vertex are well-represented, e.g., if $\ell = 3$, an equivalence class of size 3 that contains S will have two more students that took *CS201* and they all received different letter grades. Therefore, the adversary (1) cannot distinguish S from the other two records, and (2) cannot infer with probability $> 1/\ell$ any particular sensitive value of S. In the upcoming sections we show that it is not trivial to offer this privacy guarantee. In particular, straightforward application of existing k-anonymity and ℓ-diversity algorithms are not sufficient.

Adversarial Model. We assume that adversaries have background information regarding their victims' QI values. An adversary may know any combination of QI values in the same or different vertices of his victims' records. An adversary may also exploit structural/semantic links, e.g., S has taken 2 courses and bought exactly one book for *CS306*. Our anonymization technique therefore ensures anonymity with respect to records' structure as well as QIs. Our approach also covers negative knowledge (e.g., S did not take *CS204*) as well as positive knowledge (e.g., S took *CS201*). We assume that adversaries have no knowledge (positive or negative) of individuals' sensitive values.

Contributions. This thesis makes the following contributions:

- We demonstrate the plausibility of privacy attacks on hierarchical data, e.g., XML. We show how hierarchical data anonymization differs from other data models in the literature.
- We formally define two notions of privacy, k-anonymity and ℓ-diversity, for hierarchical data. We extend popular anonymization methods (generalizations and suppressions) and utility metrics (e.g., *Information Loss Metric* LM) so that they

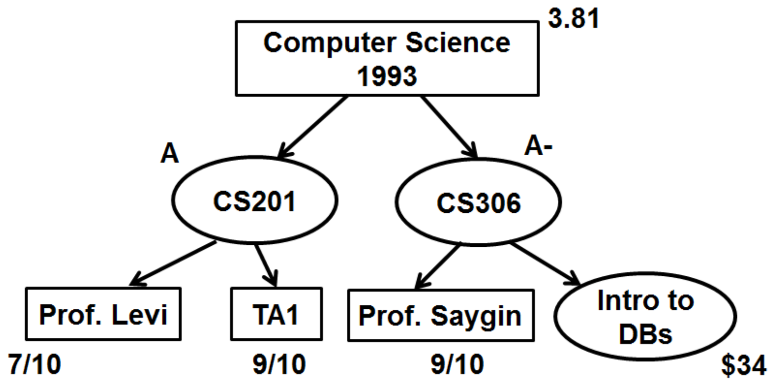


Figure 1.1: A student's hierarchical data record

can be applied to hierarchical data.

- We devise an anonymization algorithm that, given a collection of hierarchical data records, generates an ℓ-diverse output. We experimentally validate the usefulness of our algorithm and its heuristics.

- We show how anatomization technique can be used to increase utility in released hierarchical databases.

- We introduce a new privacy metric (p,m)-privacy that bounds the probability of being linked to more than m occurrences of any sensitive values by p. The new metric protects against the disclosure of frequent behaviour where frequency is controlled by the m parameter.

- We empirically demonstrate that anatomization technique can effectively increase the utility of hierarchical databases, even under strong privacy requirements.

Organization. The remainder of this thesis is organized as follows: An overview of related work is given in Section 1.2. In Section 1.3, we formally define our data model and anonymization techniques, and state related assumptions. Sections 1.4, 1.5, 1.6, 1.7 motivates our approach by explaining why ℓ-diversity is needed and why existing tabular ℓ-diversity methods are unable to ensure ℓ-diversity in hierarchical data. Chapter 2 proposes a novel anonymization algorithm based on clustering, with certain heuristics and Chapter 3 proposes new privacy technique based on anatomization with two different extensions. Finally Chapter 4 re-iterates the main points, briefly touches on future work and concludes this thesis.

1.2 Related Work

Privacy is a term that is inquiry to several disciplines. Thus, definition may vary on the context and discipline on which it is studied on. Generic explanation of privacy is a state where individuals can have freedom from interference or intrusion and has the right to be let alone. Although it has been introduced in late 19th century in ”The Right to Privacy” [10], it still remains popular due to secrecy need of man.

In this thesis, we introduce methods to meet privacy demand of the users and the applications which lies in the context of *”data privacy”*. In the domain of computing the concept of *data* has it’s origins back in early 1900’s, in the endeavours of Claude Shannon. Shannon, who is an American mathematician and is the author of highly influential article called ”A Mathematical Theory of Communication” [11], is also known as the father of information theory. In it’s simplest form *data* is information which is transformed into a structure that is adequate for movement and processing. Data privacy is ones ability to control their data in a computer system, in such fashion one has the ability to decide how much information to disclose to 3rd parties, or not release at all. Data privacy is investigated in several disciplines like health care, education and communication technologies together with growing trends of mobile computing devices.

As governments, institutions and corporations have massive of amount data which they want to publish for research purposes. In order to harvest value among data stores and discover hidden patterns, while retaining the individuals privacy demand the field of privacy preserving data publishing has emerged to satisfy requirements.

Privacy in tabular data has been widely studied. A prominent method in data anonymization is k-anonymity [5], which states that each record in a k-anonymous dataset must be indistinguishable from $k - 1$ other records with respect to their QIs. Such QI-wise

Table 1.1: Related work on hierarchical data publishing

	Data Model	**Adversarial Knowledge**	**Privacy Notion**	**Anonymization Operations**
[7]	XML	XML constraints, functional dependencies	Preventing inferences due to constraints and dependencies	Vertex and tree removal
[6]	Multi-relational SQL	Quasi-identifiers	k-anonymity	Generalization (local recoding), suppression
[8]	XML	Quasi-identifiers, dependencies	Anatomy, δ-presence	Disassociation of QIs and SAs, schema modification
[9]	Hierarchical (one label per vertex)	$\leq m$ vertex labels, $\leq n$ edges	$k^{(m,n)}$-anonymity	Generalization (global recoding), structural disassociation
Chapter 2	Hierarchical	Quasi-identifiers and their relationships	ℓ-diversity	Generalization (local recoding), suppression (partial and full)
Chapter 3	Hierarchical	Quasi-identifiers and their relationships	Anatomy	Suppression (partial and full)

equivalent groups are called equivalence classes (EC). k-anonymity is a promising step towards privacy, but it is still susceptible to attacks [12, 13]. The main concern regarding k-anonymity is that it does not consider the distribution of sensitive attributes, e.g., all individuals in an EC may have the same sensitive value. ℓ-diversity [12] was proposed to address this problem, and requires that sensitive values in each EC are *well-represented*. To achieve this, given an EC we limit an adversary's probability of inferring a sensitive value by $1/\ell$. Two popular ways of achieving k-anonymity and ℓ-diversity are *generalizations* and *suppressions*. Generalizations replace specific values by more general ones, e.g., course ID "*CS305*" can be replaced by "*CS 3rd year*" or "*CS3***". Suppressions conceal information by deleting it: Records that exist in the original data are completely removed from the final output. Since we are working with records with complex structures, we will not only use removal of entire records (i.e., full suppressions), but also partial suppressions (i.e., pruning data records by removing vertices, edges and subtrees). Data perturbation and the addition of counterfeits (i.e., fake information) is beyond the scope of our anonymization strategy, since we would like the data publisher to remain truthful (i.e., all data in the output must have originated from the input, and not be randomly spawned by the anonymization algorithm). k-anonymity was proposed by Sweeney and Samarati and since then has become a standard for privacy protection [14, 5]. It has been shown that optimal k-anonymity using generalizations and suppressions is NP-hard [15, 16]. Yet, achieving practical and efficient k-anonymity on tabular data has been an active area of research [17, 18, 19, 20, 21]. The main concern regarding k-anonymity is that it does not consider the distribution of sensitive values [13] and it is therefore susceptible to attribute linkage attacks [22]. In this thesis, we use ℓ-diversity [12] that addresses this problem. In [23], authors show that achieving optimal ℓ-diversity through generalizations is NP-hard for $\ell \geq 3$. Among notable ℓ-diversity algorithms are those in [24, 12] and [23].

Privacy notions such as k-anonymity and ℓ-diversity were initially introduced for tabular data, but they are being extended and applied to various types of complex data. Here we describe the differences between our data model and those presented in earlier works in complex data anonymization. In [25], [26] and [27], authors study variations of k-

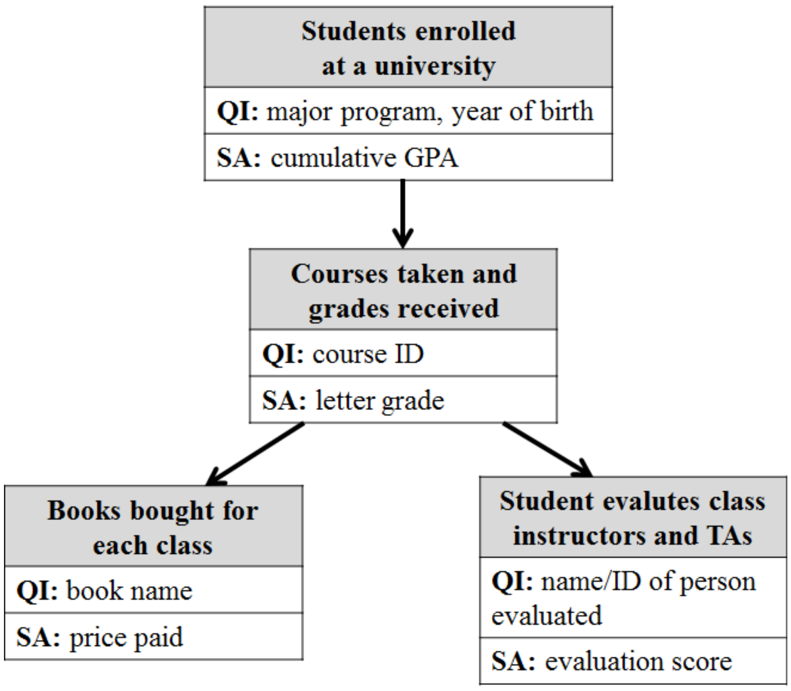

Students enrolled at a university
QI: major program, year of birth
SA: cumulative GPA
Courses taken and grades received
QI: course ID
SA: letter grade
Books bought for each class
QI: book name
SA: price paid
Student evalutes class instructors and TAs
QI: name/ID of person evaluated
SA: evaluation score

Figure 1.2: Schema for education data

anonymity (e.g., k-isomorphism) to anonymize graph data. In graph data and social network anonymization ([28]) data often comes in the form of one large graph, and the goal is to make each vertex isomorphic or indistinguishable from $k-1$ other vertices. On the other hand, our data model assumes one disjoint record per individual. Also, we presume an explicit hierarchy between vertices, and do not allow cyclic graphs. In [29], [30], [31] and [32], authors investigate privacy preserving publishing of transactional databases and set-valued data. Elements in set-valued data do not contain an order or a hierarchy, and all elements in a database originate from the same domain (e.g., market purchases, search logs). Our work considers multiple QI and sensitive attributes that each have a separate domain. Several studies (e.g., [33], [34] and [35]) use generalizations and suppressions for privacy preservation in spatio-temporal and trajectory data publishing. A trajectory is an ordered set of points where each point has one immediate neighbor (i.e, $a \rightarrow b \rightarrow c$). Whereas in hierarchical data, each vertex has multiple children that are potentially from different domains. Finally, some works such as [36] and [37] assume that the data is in tabular form, but the domains of sensitive attributes are hierarchically organized. They propose privacy definitions applicable to this particular scenario. However, we assume no ordering or hierarchy among sensitive values, and instead propose that quasi-identifying information is organized hierarchically.

Several studies investigate privacy in semi-structured and hierarchical data from the point of view of access control. In particular, access control systems for XML documents have been designed and implemented for over a decade [38, 39, 40]. However, these are orthogonal to our approach: We assume that an adversary will have full knowledge over the database once it is published. In contrast, access control methods stop unauthorized users (such as adversaries) from gaining access to sensitive information in the data.

Most closely related to our work are [9], [8], [6] and [7] that study privacy preserving publishing of hierarchical or tree-structured data. Information regarding these works is given next, and is also summarized in Table 1.1. In [7], authors focus on cases where functional dependencies in XML data cause information leakage. They formulate such dependencies as XML constraints. They propose an algorithm that sanitizes XML documents according to these constraints so that the resulting document no longer leaks information. Our adversarial model is broader: We study adversaries that also have background knowledge regarding their victims. In [8], authors introduce two anonymization schemes for XML data: an extension of *anatomy* [41] (another well-known privacy protection method) and δ-dependency. However, these methods transform the schema of XML documents by de-associating QIs and SAs. Also, they support generalizations of SAs, which intuitively work against our goal of making records ℓ-diverse. Simultaneous to our study, [9] proposed the $k^{(m,n)}$-anonymity definition for tree-structured data. In their work, attackers' background knowledge is limited to m vertex labels and n structural relations between vertices (i.e., ancestor/descendant relationships). Also contrary to our approach, they support *structural disassociations* which modify the original schema of records. In addition, they employ a global recoding approach, i.e., if a value is generalized, then all its appearances in the database must be replaced by the generalized value. This requirement can be too constraining for high-dimensional and sparse data, and therefore our solution uses local recoding that allows a value and its generalization to co-exist in the output. Furthermore, their solution is exponential in m. In [6], authors extend k-anonymity to anonymize multi-relational databases that have snowflake-shaped entity-relationship diagrams. Their definitions are primarily concerned with k-anonymity, and although they propose a method for ℓ-diversity, (1) their solution k-anonymizes the database first and then iteratively tries to find an output that is ℓ-diverse, and (2) they do not provide any experimental results. The effectiveness of their approach relies heavily on the k-anonymized database, which is obtained without taking SAs into account. On the other hand, our algorithms checks for ℓ-diversity at each anonymization step.

1.3 Preliminary

In this chapter we formally state our definitions and assumptions. We introduce concepts and terms and discuss further on the motivation of our work. In this section, we describe terms and notions used in both of the works discussed in Chapter 2 and Chapter 3. We present both formal and verbal descriptions that falls into three categories namely Data Model, Anonymization and Anatomization.

Definition 1. ***(Rooted tree)*** *Let T be a graph with n vertices. We say that T is a rooted tree if and only if:*

1. *T is a directed acyclic graph with $n - 1$ edges.*
2. *One vertex is singled out as the root vertex, and there is a single path from the root vertex to every other vertex in T.*
3. *Let $children(v) = \{c_1, ..., c_m\}$ denote the children of vertex v, i.e., there exists an edge $v \rightarrow c_i$ if and only if $c_i \in children(v)$. Then, $c_1, .., c_m$ are called siblings of one another, and we assume no ordering among them.*

We denote such trees by $T(V, E)$ where V is the set of vertices and E is the set of edges in the tree.

Definition 2. ***(Hierarchical data record)*** *We say that a hierarchical data record satisfies the following conditions:*

1. *It follows a rooted tree structure.*
2. *Each vertex v has two j-tuples ($j \geq 0$) v_{QIt} and v_{QI}, where v_{QIt} contains the names of QI attributes and v_{QI} contains the values of corresponding QIs.*
3. *Each vertex v also has two m-tuples ($0 \leq m \leq 1$) v_{SAt} and v_{SA}, where v_{SAt} contains the name of SA and v_{SA} contains the value of corresponding SA.*
4. *We assume $(|v_{QI}| + |v_{SA}|) \geq 1$ to eliminate empty vertices.*

In our examples we adopt the following notation to represent hierarchical records: We write QI values (v_{QI}) as labels of tree vertices and associated SA values (v_{SA}) right next to the vertices (as contiguous information). For the root vertex in Figure 1.1, v_{QIt}=*(major program, year of birth)*, v_{SAt}=*(GPA)*, v_{QI}=*(Computer Science, 1993)* and v_{SA}=*(3.81)*. An edge between two vertices signals that information is semantically linked, e.g., the evaluation score of *9/10* for *Prof. Saygin* in Figure 1.1 was given by this particular student and for the *CS306* course. Such links can be established through primary and foreign keys in a multi-relational SQL database, or through hierarchical object representations in XML or JSON. Conversion of any type of hierarchical data to the structure defined above is trivial, given which attributes are quasi-identifiers and which ones are sensitive.

We say that an individual's record in the database conforms to the definition of a hierarchical data record, and only one hierarchical record exists per individual. The database is a collection F that contains n hierarchical records, denoted $T_1, ..., T_n$.

Let $v_X[i]$ denote the i'th element in the r-tuple v_X, where $r = j$ or m. Let $\Omega(A)$ denote the domain of attribute A. We assume, without loss of generality, that the domains of different attributes are mutually exclusive: $\Omega(A) \cap \Omega(A^{'}) = \emptyset$ for $A \neq A^{'}$. We also require: $\forall i \in \{1, .., |v_{QI}|\}$, $v_{QI}[i] \in \Omega(v_{QIt}[i])$. Likewise, if the vertex contains a sensitive attribute (i.e., $|v_{SA}| = 1$), then $v_{SA}[1] \in \Omega(v_{SAt}[1])$.

Definition 3. ***(Union-compatibility)*** *Two vertices v and $v^{'}$ are union-compatible if and only if $v_{QIt} = v^{'}_{QIt}$ and $v_{SAt} = v^{'}_{SAt}$.*

We use union-compatibility akin to database relations: Two database relations are union-compatible if they share the same number of attributes and each attribute is from the same domain. Similarly, in our case, two vertices are union-compatible if they follow the same schema (i.e., same QIs and SAs).

In tabular data, suppression of a row refers to the removal of that row from the published dataset (or equivalently, all values in that row are replaced by "*"). In our setting, this translates to completely removing an individual's hierarchical record. Although this might be necessary and we support this operation, its effect is also drastic: If the deleted record is large (i.e., contains a lot of vertices), then a lot of useful information might be lost. We therefore introduce partial suppressions.

Definition 4. ***(Partial suppression)*** *We say that a hierarchical data record T^* is a partially suppressed version of T, if T^* is obtained from T by first removing exactly one edge from T (call this e) and then deleting all vertices and edges that are no longer accessible from the root of T (i.e., there is no longer a path from the root to them). We write $T^* = \varphi_e(T)$ to denote this operation.*

Intuitively, a partial suppression is nothing but tree pruning. Such pruning can lead to the deletion of a single vertex or a subtree containing multiple vertices and edges. Note that the remainder of the data record is untouched, i.e., vertices that "survive" the partial suppression operation incur no changes to their QIs or sensitive values. Figure 2.7 contains several examples: From Figure 2.7a to Figure 2.7b, the upper record loses the vertex with *TA5* under *CS404*. From Figure 2.7a to Figure 2.7c, the edge between the root and *CS404* is broken, which leads to the suppression of a larger subtree (i.e., children of *CS404* are also deleted). We explicitly replace suppressed vertices with dashed lines and lost information (both v_{QI} and v_{SA}) with "*" for demonstration purposes. They are otherwise not part of the output.

Definition 5. ***(Structural isomorphism)*** *Let $T_1(V_1, E_1)$, $T_2(V_2, E_2)$, ..., $T_n(V_n, E_n)$ denote a group of trees with vertex sets V_i and edge sets E_i respectively. Let $\mathcal{R}(T_i) = \{v_1^i, v_2^i, .., v_m^i\}$ denote the breadth-order (level-order) traversal of T_i. The group of trees is structurally isomorphic if:*

1. *For $i \in [1, n-1]$, we have: $|\mathcal{R}(T_i)| = |\mathcal{R}(T_{i+1})| = m$.*

2. *For $j \in [1, m]$, let $\mathcal{I}_j = \bigcup_{i\in[1,n]} v_j^i$ denote the set of vertices at the j'th index of the traversal. Then, all pairs of vertices in I_j are union-compatible.*

Definition 6. *(**ℓ-diversity**) Let $X = \{s_1, s_2, ..., s_n\}$ be a multiset of values from the domain of a sensitive attribute A, i.e., $s_i \in \Omega(A)$. Let $f(s_i)$ denote the frequency of value s_i in X. Then, X is ℓ-diverse if for all s_i, $f(s_i) \leq 1/\ell$.*

Informally, this probabilistic ℓ-diversity definition states that the frequency of all sensitive values must be bounded by $1/\ell$.

Sensitive attributes can be categorical (e.g., letter grade) or continuous (e.g., GPA). The domain of categorical SAs consists of discrete values (e.g., letter grades from A to F), and it is straightforward to evaluate ℓ-diversity on a set of discrete values as above. However, continuous SAs require an intermediate discretization step. The domain of a continuous SA is divided into non-overlapping buckets, and X then contains the buckets data values fall into. (E.g., GPA domain $[0.0 - 4.0]$ can be divided into 8 buckets of size 0.5. A GPA value 3.26 can then translate to the bucket $[3.0 - 3.50)$.) We do not enforce a specific discretization, instead our algorithms can work with an arbitrary discretization that meets the demands and preferences of the data publisher. We also allow discretizations to contain buckets with different sizes.

Definition 7. *(**Diversity of vertices**) Let $V = \{v^1, ..., v^n\}$ be a set of vertices from hierarchical data records. We study two cases:*

- *For $v^j \in V$, $|v^j_{SA}| = 0$. Then, V is ℓ-diverse if and only if all vertices in V are pairwise union-compatible.*

- *For $v^j \in V$, $|v^j_{SA}| = 1$. Let X be defined as $X = \{v^1_{SA}[1], v^2_{SA}[1], ..., v^n_{SA}[1]\}$. Then, V is ℓ-diverse if and only if all vertices in V are pairwise union-compatible and X is ℓ-diverse.*

Students enrolled at the university

studentID	major	YoB	GPA
S1	Computer Science	1992	3.25
S2	Computer Science	1993	2.67

Courses and letter grades

studentID	course	grade
S1	CS201	A
S1	CS202	A-
S2	CS401	B
S2	CS404	B+

Figure 1.3: Students *S1* and *S2* and their courses as two tables linked using studentIDs (primary key in Table 1, foreign key in Table 2)

Students enrolled at the university

studentID	major	YoB	GPA
S1	Computer Science	199*	3.25
S2	Computer Science	199*	2.67

Courses and letter grades

studentID	course	grade
S1	CS2**	A
S1	CS2**	A-
S2	CS4**	B
S2	CS4**	B+

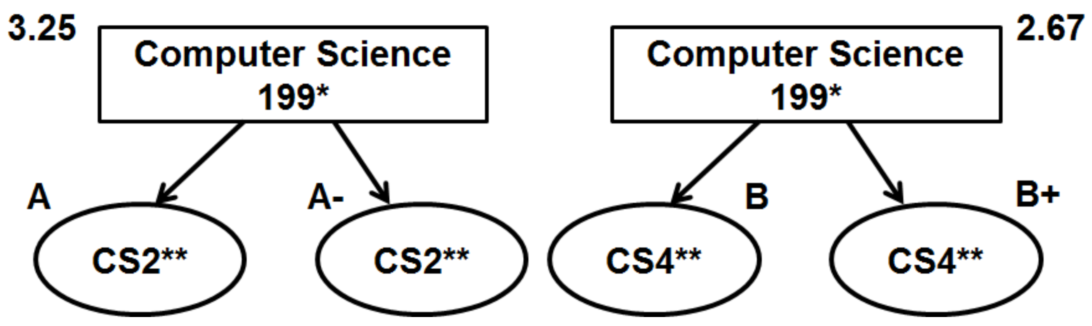


Figure 1.4: Potential result if the two tables in Figure 1.3 are anonymized independently

1.4 ℓ-diversity vs. k-anonymity in Hierarchical Data

Prior approaches in hierarchical (and tree-structured) data anonymization against linkage attacks can be divided into two camps: providing privacy by disassociating QIs and SAs [8] and extensions of k-anonymity (e.g., multi-relational k-anonymity [6] and $k^{(m,n)}$-anonymity [9]). The former publishes QI values and SA values separately, hence an adversary cannot determine the sensitive value of a particular vertex (e.g., the letter grade S received from course *CS201*). In the latter, records are anonymized in terms of structure and labels (QIs in our case), but sensitive values are left unattended. (In particular, [9] has no distinction between QI and SA.) Both may result in equivalence classes that leak sensitive values with significant probabilities.

Let us demonstrate the plausibility of homogeneity and background knowledge attacks on hierarchical data, where data is k-anonymized according to [6] or [9]. Say that a 2-anonymous dataset has been published, such as the one in Figure 2.7b. Let the adversary know beforehand that there will be at most two students that majored in *Computer Science* and were born in the *1990s*. His victim S is among these two students. The adversary links S to the records in Figure 2.7b. At this point, the published dataset leaks the following pieces of information: (1) S received an *A-* from *CS404*. (2) S submitted an evaluation score of *8* for *Prof. Levi* in *CS201*. The peculiarity of this example comes from the fact that the adversary had no knowledge of QI values for the vertices that leaked these information (e.g., the adversary did not know that S evaluated *Prof. Levi*). Both of these privacy leaks could have been avoided if the published data was 2-diverse as in Figure 2.7c.

1.5 ℓ-diversity in Tabular vs. Hierarchical Data

As reported earlier, several algorithms that apply ℓ-diversity to tabular data have been implemented. In applicable situations, one way of processing hierarchical data is to reduce it to tabular data and then run tabular algorithms on it. There are also arguments that say in most scenarios, converting hierarchical data to a single giant relation and then using single-table algorithms is undesirable because of potential loss of information and semantic links between data records [42]. We now demonstrate that such conversions and reductions are not sufficient also for privacy protection.

1.6 Anonymizing Relations Separately

A hierarchical schema (e.g., Figure 1.2) can be represented using multiple database relations that are linked via primary and foreign keys (i.e., join keys). Then, a straightforward approach would be to consider each relation independently and run tabular ℓ-diversity algorithms on them.

Consider the two tables in Figure 1.3, where *studentID*s are added and used as join keys. When these two tables are treated independently, a resulting anonymization could be the one in Figure 1.4. It can easily be verified that both tables are 2-diverse by themselves. Converting the result into our hierarchical representation, though, we see that students *S1* and *S2* are neither 2-anonymous nor 2-diverse. An adversary that knows *S1* took *CS201* learns the GPA of *S1*, since *S2* has not taken any *CS200*-series courses.

The main problem of this *independent anonymization* approach is that anonymizations are not guaranteed to be consistent between multiple tables. In the first table, *S1* and *S2*'s tuples are anonymized with respect to each other, but a tabular anonymization algorithm does not acknowledge this when anonymizing the second table. Hence, *S1*'s tuples may be bundled together and *S2*'s tuples may be bundled together while creating a 2-diverse version of the second table.

1.7 Constructing and Anonymizing a Universal Relation

Another approach is to *flatten* hierarchical data into one big relation called the *universal relation*, i.e., the universal relation is obtained by joining all relations in a hierarchical schema using join keys. Figure 1.5 provides a sample universal relation. Notice that this creates a significant amount of redundancy and undesirable dependencies. Information in deeper vertices of the records have to be rewritten for each descendant connected to that

Join key	*Quasi-Identifiers*			*Sensitive Attributes*	
studentID	**major**	**YoB**	**course**	**grade**	**GPA**
S3	Computer Science	1993	CS301	C+	2.17
S4	Computer Science	1995	CS305	A	3.78
S4	Computer Science	1995	CS306	A-	3.78
S5	Computer Science	1994	CS301	B	2.96

Figure 1.5: Universal relation constructed by joining the *Enrollment* and *Courses* relations with students *S3*, *S4* and *S5* using studentIDs

vertex (e.g., QIs major and year of birth are repeated for each course taken). A second problem is that leaf vertices may be at different depths, which will force work-arounds such as having *null* values in the universal relation. For example, in Figure 1.5 if *S3* had not taken any courses, we would either have to remove him from the universal relation, or enter *null*s for his course and grade. Here we show the ineffectiveness of the *universal relation* approach even ignoring the problems discussed up to this point.

The table in Figure 1.6 is 2-diverse in terms of the two sensitive attributes, *GPA* and *grade*. However, the hierarchical records of *S3*, *S4* and *S5* are not anonymous: *S3* and *S5* are shown having taken one *CS3*** course each, but *S4* has taken two. An adversary that knows *S4* is the only student who has enrolled in more than one *CS3*** course can learn the grades *S4* received from these courses, together with *S4*'s GPA. The problem this time arises from the fact that each individual may have an unknown number of entries in the universal relation.

1.8 Problem Definition

Having established the preliminaries, in this section we formally define and state the problem.

We now discuss why we require every record T_i^* to belong to *exactly one* ℓ_j-diverse equivalence class. If T_i^* does not belong to exactly one ℓ_j-diverse equivalence class, then

Join key	*Quasi-Identifiers*			*Sensitive Attributes*	
studentID	**major**	**YoB**	**course**	**grade**	**GPA**
S3	Computer Science	199*	CS3**	C+	2.17
S4	Computer Science	199*	CS3**	A	3.78
S4	Computer Science	199*	CS3**	A-	3.78
S5	Computer Science	199*	CS3**	B	2.96

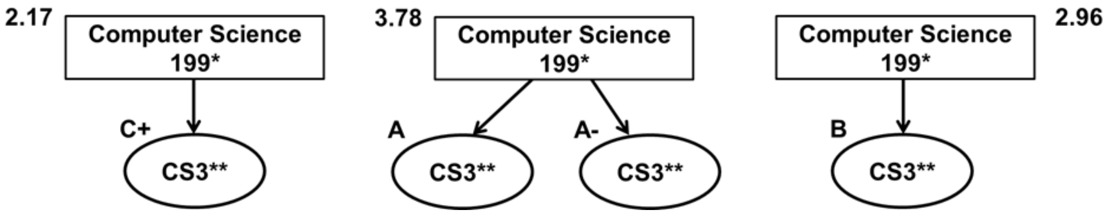


Figure 1.6: 2-diverse version of the universal relation in Figure 1.5

it either belongs to less than one ℓ_j-diverse equivalence class or multiple equivalence classes. Say that T_i^* does not belong to an ℓ_j-diverse equivalence class where $\ell_j \geq \ell$. That is, T_i^* belongs to a t-diverse equivalence class where $t < \ell$. Then, clearly it is possible with certain background knowledge, an adversary will be able to infer the sensitive attribute in T_i^* with probability $1/t$, which is greater than $1/\ell$. This defeats the purpose of ℓ-diversity and the privacy protection we offer in this thesis. On the other hand, say that T_i^* belongs to multiple equivalence classes that are ℓ_j-diverse. We construct an example to demonstrate the privacy breach here: Let T_1, T_2 and T_3 be three records that each contain a single vertex, ℓ be 2, and T_1-T_2 and T_2-T_3 be the two equivalence classes (notice that T_2 appears in both equivalence classes). Since T_1-T_2 and T_2-T_3 constitute equivalence classes, due to QI-isomorphism, we know that they have the same QIs. Say that an adversary has knowledge of these QIs and tries to infer a sensitive attribute. If T_1 and T_3 have the same sensitive value (and T_2 has a different sensitive value) then the probability of an adversary inferring a sensitive value is $2/3$, which is greater than $1/2$ $(1/\ell)$. Whereas if T_2 was not part of both equivalence classes (e.g., T_1-T_2 was an equivalence class, and there was a fourth record T_4, where T_3-T_4 was an equivalence class) then the probability of inference would be at most $2/4$, even if all four records had the same QIs. $2/4 = 1/2$

(i.e., $1/\ell$), hence there would be no privacy breach.

Given a collection of hierarchical data records $F(T_1, T_2, ..., T_n)$, an anonymized output F^* is generated via the following principle: For each record $T_i \in F$, either T_i is fully suppressed and does not appear in F^*, or T_i is transformed into $T_i^* \in F^*$ by performing a set of generalizations $\{\Delta\}$ and partial suppressions $\{\varphi_e(T_i)\}$. With these definitions in mind, the problem we study in this thesis can be stated as follows: Given a set of hierarchical data records F, we would like to compute an ℓ-diverse output F^* with minimal information loss, using the anonymization principle above.

Chapter 2

Privacy Preserving Generalization of Hierarchical Data

2.1 Overview

In this chapter we present a novel privacy preserving publishing technique on hierarchical datasets. The least one can do to protect privacy is to delete explicitly identifying information (e.g., SSN, name). However, it has been shown that this is ineffective: [43] and [5] report that a set of *quasi-identifier (QI)* attributes (e.g., gender, zipcode, date of birth) can uniquely identify the majority of a population and also lead to *linkage attacks* [22]. An adversary performs a linkage attack by knowing one or more QI values of his victim, and trying to infer the victim's *sensitive attribute (SA)* (e.g., GPA, health condition) values.

2.2 Generalization of Hierarchical Data

Domain generalization hierarchies (DGH) [12] are taxonomy trees that provide a hierarchical order and categorization of values. We assume that a DGH is either available or easily inferable for each QI. Note that this assumption is widely adopted in the anonymization literature [22, 6]. Values observed in the database appear as the leaves of DGHs. The root vertices of DGHs contain "*" to mean "any value", i.e., value completely hidden. A

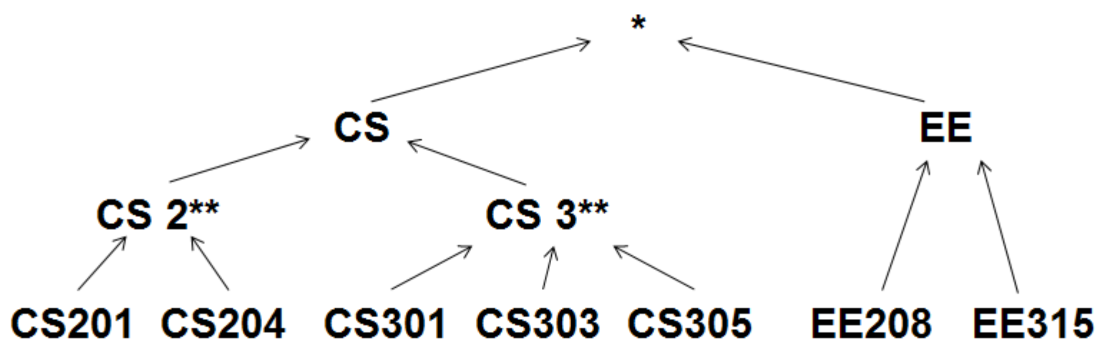


Figure 2.1: Sample generalization hierarchy for course IDs

DGH is given for attribute *course ID* in Figure 2.1.

Definition 8. ***(Generalization function)*** *For two data values x and x^* from the same QI attribute A, x^* is a valid generalization of x, written $x^* \in \phi(x)$, if and only if x^* appears as an ancestor of x in the DGH of A. We abuse notation and write $\phi_l^{-1}(x^*)$ to indicate all possible leaves that can be generalized to value x^* using valid generalizations.*

For example, for the QI *course ID*, *CS3*** $\in \phi$(*CS303*) and *CS* $\in \phi$(*CS303*), whereas *CS2*** $\notin$ ϕ(*CS303*). Also, ϕ_l^{-1}(*CS3***) = {*CS301, CS303, CS305*}, and ϕ_l^{-1}(*CS305*) = {*CS305*}.

Definition 9. ***(Vertex generalization)*** *We say that vertex v^* is a valid generalization of v and write $v^* \in \Delta(v)$, if:*

1. *v and v^* are union-compatible.*

2. *$v_{QI} \neq v^*_{QI}$.*

3. *$\forall a^* \in v^*_{QI}$, either $a^* \in v_{QI}$ or there exists $a \in v_{QI}$ such that $a^* \in \phi(a)$.*

4. *$v_{SA} = v^*_{SA}$.*

In words, a vertex is generalized when at least one of its QI values gets replaced by a value that is more general according to the attribute's DGH. A vertex generalization leaves sensitive values intact.

Various metrics were proposed and used in relevant literature to calculate costs of anonymization [17, 44, 18, 45]. In this thesis, we will use an extension of the general *loss metric* (LM) [20]. Similar extensions were previously applied in a number of settings, including medical health records [46] and multi-relational databases [6].

Definition 10. ***(Individual LM cost)*** *Given a DGH for attribute A and a value $x \in \Omega(A)$ (i.e., x exists in A's DGH), the individual LM cost of value x is:*

$$LM'(x) = \frac{|\phi_l^{-1}(x)| - 1}{|\phi_l^{-1}(r)| - 1}$$

where r denotes the root of A's DGH.

Definition 11. ***(LM cost of a collection of hierarchical records)*** *Let F and F^* be collections of hierarchical data records, where F^* is obtained via anonymizing F. Let Ψ denote the set of vertices that exist in records in F but do not exist in F^* due to partial or full suppressions of records. Then, the LM cost of F^* is:*

$$LM(F^*) = \frac{\left(\sum\limits_{T_i^* \in F^*} \sum\limits_{v^* \in T_i^*} \sum\limits_{q^* \in v^*_{QI}} LM'(q^*)\right) + \left(\sum\limits_{p \in \Psi} |p_{QI}|\right)}{\sum\limits_{T_i \in F} \sum\limits_{v \in T_i} |v_{QI}|}$$

These cost metrics measure the utility loss due to generalizations and suppressions. LM' is defined on QI values, and asserts a cost according to how general a QI value is. For example, according to Figure 2.1, $LM'(\textit{CS}) = 4/6$, $LM'(\textit{CS2**}) = 1/6$ and $LM'(\textit{CS201}) = 0$. Intuitively, if the output contains *CS* instead of *CS2*** or *CS201*, there is higher ambiguity regarding the initial QI value that was generalized to *CS*. Hence, LM' assigns a higher penalty to more general QIs.

We use LM' to build $LM(F^*)$, a cost metric that is suitable to our setting. In this definition, the anonymization cost is broken down into two factors: The first factor calculates the cost incurred by generalizations of vertices that appear in the published data. The second factor adds the cost of suppressions. The total cost is calculated on the order of labels rather than vertices or trees, to better focus on each individual piece of data lost during anonymization.

One can verify that the LM' cost of a QI is within the range [0, 1], where the root of a DGH receives the highest penalty (1) and leaves receive no penalty (0). Consequently, we ensure that $LM(F^*)$ is also normalized to a value within [0, 1].

We compute the LM cost of anonymizing the two records in Figure 2.7c to provide an example for $LM(F^*)$. Assume that F consists of only the two records in Figure 2.7a, and F^* is the records in Figure 2.7c. Further assume the LM costs of generalizing years of birth *1994* and *1995* to *199** is $1/10$, course IDs *CS306* and *CS305* to *CS3*** is $1/3$, instructors *Prof. Saygin* and *Prof. Nergiz* to *DB Prof.* is $2/7$, and *TA1* and *TA2* to *TA* is $1/2$. Then,

$$LM(F^*) = \frac{(\frac{1}{10} + \frac{1}{3} + \frac{2}{7} + \frac{1}{2}) \cdot 2 + 7}{19} = 0.497$$

Definition 12. ***(QI-isomorphism)*** *Let $T_1(V_1, E_1)$ denote a hierarchical data record with a set of vertices V_1 and edges E_1. A data record $T_2(V_2, E_2)$ is QI-isomorphic to T_1 if and only if there exists a bijection $f : V_1 \to V_2$ such that:*

1. *For $x, y \in V_1$, there exists an edge $e_i \in E_2$ from $f(x)$ to $f(y)$ if and only if there exists an edge $e_j \in E_1$ from x to y.*
2. *The root vertex is conserved; i.e., denoting the root of the first tree as $r_1 \in V_1$ and the root of the second tree as $r_2 \in V_2$, $f(r_1) = r_2$.*
3. *For all pairs (x, x'), where $x \in V_1$ and $x' = f(x)$, x and x' are union-compatible and $x_{QI} = x'_{QI}$.*

Definition 13. ***(Equivalence class of hierarchical records)*** *We say that records $D = \{T_1, .., T_k\}$ are k-anonymous and form an equivalence class, if for all i, j where $1 \leq i, j \leq k$, the pair (T_i, T_j) is QI-isomorphic.*

Two records are QI-isomorphic if they appear to be completely same when all sensitive values are deleted from both. In other words, they are indistinguishable in terms of labels and structure. There is a clear analogy between the traditional definition of equivalence classes in tabular k-anonymity and our definition for hierarchical records: Both state that an equivalence class is a set of records that are indistinguishable with respect to their QIs.

Definition 14. ***(ℓ-diverse equivalence class)*** *We say that records $\{T_1, .., T_k\}$ form an ℓ-diverse equivalence class, if and only if:*

1. *$\{T_1, .., T_k\}$ constitute an equivalence class.*

2. *For all $1 \leq i \leq k-1$, let f_i be a bijection that maps T_1's vertices to T_{i+1}'s vertices, as in QI-isomorphism. Let T_1 have n vertices, labeled arbitrarily as $v_1^1, v_2^1, v_3^1, .., v_n^1$. Then, there should exist a set of bijections $\{f_1, f_2, .., f_{k-1}\}$ such that $\forall x \in \{1, 2, .., n\}$, the set of vertices $V = \{v_x^1, f_1(v_x^1), f_2(v_x^1), .., f_{k-1}(v_x^1)\}$ is ℓ-diverse.*

ℓ-diversity proposes the following extension to k-anonymity: Given a set of k-anonymous records, we are certain that they are pairwise QI-isomorphic, and it is possible to generate a set of bijections $\{f_1, f_2, .., f_{k-1}\}$ to *match* their vertices that are equivalent in terms of structure and QIs. Matching vertices should be ℓ-diverse (i.e., Definition 7) so that for every piece of QI or structure-wise knowledge, the corresponding vertices yield a sensitive value with probability no more than $1/\ell$.

We should point out that multiple bijections between two records' vertices are possible if they contain multiple union-compatible sibling vertices with identical QIs. In such cases, it is too restrictive to require that all possible bijections satisfy ℓ-diversity, therefore our definition states that it would suffice to have one bijection that does.

Figure 2.7 contains two records together with their 2-anonymous and 2-diverse versions. This is just one way of anonymizing these records, there are also other correct (i.e., fitting the definition of *anonymity* and *diversity*) anonymizations. The quality of these anonymizations, however, depend on how much information is lost (according to an appropriate cost metric). An anonymization that satisfies k-anonymity or ℓ-diversity and yields the lowest information loss is most desirable.

An alternative representation of an equivalence class which we use in later sections is the class representative for a given equivalence class. A class representative $\widehat{T}$ is es-

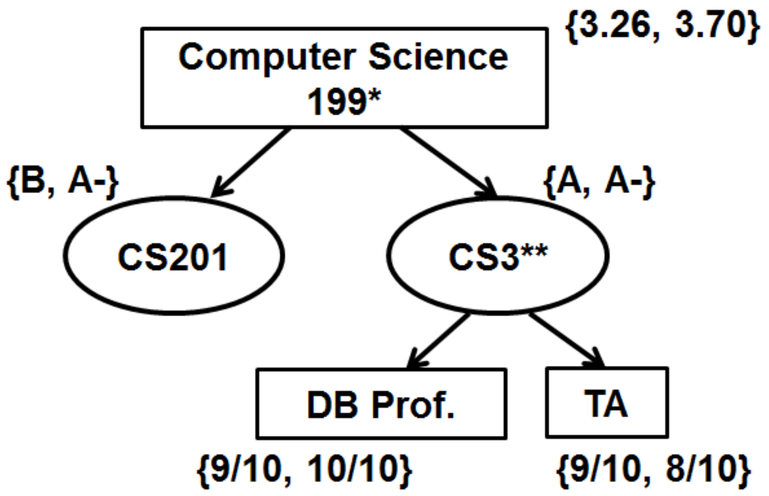


Figure 2.2: A class representative

sentially a hierarchical data record with one extension: If a vertex contains a sensitive attribute, its value is not a single element, but rather a list of elements. ($\forall v \in \widehat{T}$, v_{SA} returns a set rather than a single sensitive value.) We formally define class representative as follows:

Definition 15. ***(Class representative)*** *Given an equivalence class $D = \{T_1, .., T_k\}$ with the corresponding set of bijections $\{f_1, f_2, .., f_{k-1}\}$, we say $\widehat{T}$ is the class representative for D if $\widehat{T}$ is QI-isomorphic to T_1 with a bijection function f and $\forall v \in \widehat{T}$, $v_{SA} = \{f(v)_{SA}, f_1(f(v))_{SA}, \ldots, f_{k-1}(f(v))_{SA}\}$.*

Figure 2.2 shows a representative for the equivalence class given in Figure 2.7c. It is easy to show that a given equivalence class is ℓ-diverse if and only if the corresponding representative is ℓ-diverse, that is $\forall v \in \widehat{T}$, the set v_{SA} satisfies ℓ-diversity.

Definition 16. ***(ℓ-diversity of a database)*** *A collection of records $F^*(T_1^*, ..., T_n^*)$ is ℓ-diverse if every record $T_i^* \in F^*$ belongs to exactly one ℓ_j-diverse equivalence class, and for all ℓ_j, $\ell_j \geq \ell$ holds.*

2.3 Anonymization Algorithm

We designed and implemented a solution to the anonymization problem stated at the Section 1.8. Before moving forward, we would like to underline two important characteristics of our anonymization scheme. First, our approach ensures that the data publisher remains truthful. The output does not contain any information that did not exist originally in the input, i.e., we do not consider adding new vertices, changing QIs of vertices (other than generalizing them), or adding new QIs or SAs to existing vertices. Second, vertices that appear in the output have the same depth, adjacency and parent as they did in the input. That is, the structure of records in the output are consistent with the input. This schema preservation enables easier data mining without any ambiguity.

We present our algorithm in two steps: (1) Given two records, we focus on how to anonymize them with respect to each other so that they become 2-diverse with low information loss. (2) We build a clustering algorithm that employs the previous step and class representatives to anonymize an arbitrary number of records.

2.3.1 Pairwise Anonymization

Converting two records to a 2-diverse pair is pivotal not only because we use it as a building block in our clustering algorithm, but also we employ it as a similarity metric (i.e., to calculate *distance* between two hierarchical data records). In addition, given a fixed pair of records as inputs, the anonymization function should be able to produce a 2-diverse output with as little information loss as possible. Therefore, it relies on finding vertices and subtrees that are *similar* in both records.

We define the following notation: Let $root(T)$ denote the root vertex of the hierarchical data record T, and $subtrees(v)$ denote the subtrees rooted at the children of v (i.e., for each child c_i of v, the hierarchical data record rooted at c_i is included in $subtrees(v)$). Given two QI values X and Y both from the same QI domain, and Z that is the DGH of the QI, we say that function $mrca(X, Y, Z)$ returns the lowest (i.e., most recent) common ancestor of X and Y according to Z. Assume that the function $cost(T)$ returns the cost of anonymization of T, given a pre-defined cost metric CM. An applicable cost metric is LM, and in that case, cost of a record T is:

$$cost(T) = (\sum_{v \in V} \sum_{q \in v_{QI}} LM'(q)) + (\sum_{w \in \Psi} |w_{QI}|)$$

where V denotes the vertices in T that are not suppressed and Ψ denotes the vertices that were in T but are now suppressed. Let $clone(T)$ return a copy of T. Furthermore, given two vertices a and b, let $u\text{-}comp(a, b)$ test the union-compatibility of a and b, and $diverse(a, b)$ have the following behavior:

$$diverse(a,b) = \begin{cases} \text{true} & \text{if } \textit{u-comp(a,b)} \text{ and } a_{SA} \cap b_{SA} = \emptyset \\ \text{false} & \text{otherwise} \end{cases}$$

A function that anonymizes hierarchical records in top-down manner is presented in Algorithm 1. We refer to this function as **diversify**. Without loss of generality, we assume that for the two input hierarchical records T_1 and T_2 (rooted at a and b, respectively), $|$children(a)$| \leq |$children(b)$|$. (Otherwise T_1 and T_2 can be interchanged as the first step.) The algorithm can be studied in several steps. First step checks the union compatibility and diversity of root vertices a and b. If a and b cannot be anonymized, then their trees are suppressed. In the second step (lines 7-10), we generalize the QIs of a and b according to their DGHs. Resulting a and b will be indistinguishable in terms of QIs. In step 3 (lines 11-17), the algorithm checks if further calculation is needed: If a and b both have children, then we need to find a low-cost anonymization of their subtrees. If one does not have any children, then we can safely suppress the children and subtrees of the other. (Otherwise it would be impossible to achieve QI-isomorphism due to structural difference.) When the algorithm reaches line 18, it has dealt with the current level (i.e., checked if root vertices are diverse, anonymized them and ensured that both have children). A low cost pairing (i.e., mapping) between the subtrees rooted at a's children and the subtrees rooted at b's children is returned by the function *FindMapping*. (We will give a detailed explanation of how the mapping is computed in the next section.) Pairs returned by the function are suitable candidates to be anonymized with one another. Hence, *diversify* is run recursively

Algorithm 1 Top-down anonymization of hierarchical records

```
Input: Two hierarchical data records (or class representatives) $T_1$ and $T_2$, anonymization
    cost metric for *cost* calculation, DGHs of QI attributes for finding *mrca*
Require: |children(root($T_1$))| $\leq$ |children(root($T_2$))|, otherwise swap $T_1$ and $T_2$
 1: procedure DIVERSIFY
 2:     $a \leftarrow$root($T_1$)
 3:     $b \leftarrow$root($T_2$)
 4:     if $\neg$diverse($a$,$b$) then
 5:         suppress $T_1$ and $T_2$
 6:         return cost($T_1$) + cost($T_2$)
 7:     for $i = 1$ to $|a_{QI}|$ do
 8:         g $\leftarrow$ mrca($a_{QI}[i]$, $b_{QI}[i]$, DGH of $a_{QIt}[i]$)
 9:         replace $a_{QI}[i]$ with g
10:         replace $b_{QI}[i]$ with g
11:     if subtrees($a$) $= \emptyset$ and subtrees($b$) $= \emptyset$ then
12:         return cost($T_1$) + cost($T_2$)
13:     else if subtrees($a$) $= \emptyset$ and subtrees($b$) $\neq \emptyset$ then
14:         let $E$ be the set of outgoing edges from $b$
15:         for $e \in E$ do
16:             $T_2 \leftarrow \varphi_e(T_2)$
17:         return cost($T_1$) + cost($T_2$)
18:     $P \leftarrow$ FindMapping(subtrees($a$), subtrees($b$))
19:     for each pair $(a_i, b_j) \in P$ do
20:         diversify$(a_i, b_j)$
21:     for $v \in$ subtrees($b$) and $\nexists(x, v) \in P$ for some $x$ do
22:         Let $e$ be the edge from $b$ to $v$
23:         $T_2 \leftarrow \varphi_e(T_2)$
24:     return cost($T_1$) + cost($T_2$)
```

on each pair (lines 19-20). Since we assumed $|\text{children}(a)| \leq |\text{children}(b)|$, all subtrees rooted at a's children will be paired, but some subtrees rooted at b's children might be left-overs (i.e., they remain unpaired). Unpaired subtrees are suppressed (lines 21-23) to achieve QI-isomorphism of T_1 and T_2. Finally, a successful execution of *diversify* always returns the cost of anonymizing its inputs (see the return statements throughout).

2.3.2 Finding a Good Mapping

Recall that *FindMapping* is called using two lists of hierarchical data records S and U (where $|S| \leq |U|$), and the goal is to produce a set of pairs $\{(s, u) \mid s \in S, u \in U\}$ that are similar. We measure similarity as the cost of anonymization. Finding an optimal solution to this problem requires finding all mappings between all elements in S and U, and picking the mapping that yields the lowest cost. However, this is infeasible: Let S have n elements and U have m elements, where $m \geq n$. The number of possible pairings between S and U is $\binom{m}{n} \cdot n!$, which implies exponential complexity. This becomes a significant problem when the branching factor of input data records is large. (Even for toy datasets with average branching factors of 6-7, optimal search took several hours.) We therefore need heuristic strategies for *FindMapping*. Based on this observation, we now describe two different solutions to the problem: one that employs a greedy algorithm, and another that models the problem as an optimization problem using linear programming.

The greedy algorithm. This heuristic traverses S by picking one element at a time, and finds the most suitable candidate in U to pair the element with. A more formal description is given in Algorithm 2. The greedy solution has no guarantees of finding a global optimum, but instead settles for a local optimum in each iteration (i.e., for each element in S).

The procedure in Algorithm 2 works as follows: We pick one record at a time from the first set S and call this record f (line 3). Then, we consider each unpaired element v in the second set U and compute the information loss of anonymizing f with v (lines 6-10). This is done by first making copies of f and v (to make explicit that we do not modify the original records) and then running *diversify* on them. The record that yields

Algorithm 2 Finding a low-cost mapping greedily

Input: Two lists of hierarchical data records S and U, where $|S| \leq |U|$

```
 1: procedure FINDMAPPING-GRD
 2:     $P \leftarrow \emptyset$
 3:     for each $f \in S$ do
 4:         minCost $\leftarrow +\infty$
 5:         match $\leftarrow \emptyset$
 6:         for each $v \in U$ do
 7:             $f' \leftarrow$ clone($f$)
 8:             $v' \leftarrow$ clone($v$)
 9:             c $\leftarrow$ diversify($f', v'$)
10:             if c $<$ minCost then
11:                 minCost $\leftarrow$ c
12:                 match $\leftarrow v$
13:         $P \leftarrow P \cup$ ($f$,match)
14:         $U \leftarrow U-$ match
15:     return $P$
```

the lowest cost wins and gets to pair up with f (lines 10-14). We repeat this procedure until S is exhausted.

An interesting heuristic is to find a strategy to choose f from S in a way such that Algorithm 2 performs better. In other words, can the order in which f is chosen from S affect the final outcome? We tried several heuristics for this, e.g., based on the frequencies of the vertices and QIs. However, our results were not consistent. One strategy performed better in some occasions, but worse in others. Also, the increase or decrease in the utility of the outputs were negligible. Thus, we refrain from building a strategy on top of greedy mapping, but instead propose the approach described next.

Reduction to an assignment problem. We propose a second strategy for *FindMapping*: We model the problem in hand as a linear sum assignment problem (LSAP). LSAP is a famous linear programming and optimization problem [47], where one has n agents that need to be assigned to n tasks. Assigning an agent to a task has a certain cost that depends on the task and the agent performing it. The goal is to find an assignment such that all tasks are performed by assigning one agent to each task, one task to each agent and the total cost of the assignment (i.e., linear sum of task-agent pairs selected) is minimized.

More formally, given an $n \times n$ cost matrix $C = (c_{ij})$ and a binary variable x_{ij} representing the assignment of agent i to task j, a LSAP can be modeled as:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \cdot x_{ij}$$

$$\begin{aligned}
\text{Subject to:} \quad & \sum_{i=1}^{n} x_{ij} = 1 && \text{for } j = 1,2,..,n \\
& \sum_{j=1}^{n} x_{ij} = 1 && \text{for } i = 1,2,..,n \\
& x_{ij} \in \{0,1\} && \text{for } i,j = 1,2,..,n \\
& c_{ij} \geq 0 && \text{for } i,j = 1,2,..,n
\end{aligned}$$

We use the Hungarian algorithm [48] to solve a LSAP, which finds an optimal (i.e., lowest-cost) solution to the problem above in $\mathcal{O}(n^3)$ time. The solution is a collection of x_{ij}s that tell which agent is assigned to which task.

Algorithm 3 Finding a low-cost mapping using a LSAP

```
Input: Two lists of hierarchical data records $S$ and $U$, where $|S| \leq |U|$
1: procedure FINDMAPPING-LSAP
2:     Initialize $|U| \times |U|$ cost matrix $C$
3:     for $i = 1$ to $|S|$ do
4:         for $j = 1$ to $|U|$ do
5:             $f' \leftarrow$ clone($S[i]$)
6:             $v' \leftarrow$ clone($U[j]$)
7:             $c_{ij} \leftarrow$ diversify($f', v'$)
8:     for $i = |S| + 1$ to $|U|$ do
9:         for $j = 1$ to $|U|$ do
10:            $c_{ij} \leftarrow$ cost of suppressing $U[j]$
11:    $X \leftarrow$ solve the LSAP with cost matrix $C$
12:    $P \leftarrow \emptyset$
13:    for each $x_{ij} \in X$ do
14:        if $x_{ij} = 1$ and $i \leq |S|$ then
15:            $P \leftarrow P \cup (S[i], U[j])$
16:    return $P$
```

We now explain how we use LSAPs in *FindMapping*. The process is shown in Algorithm 3. Given two lists of records S and U, we treat the records in S as agents, and the records in U as tasks in a LSAP. We calculate the cost of an agent-task pair by running *diversify* on them, which computes the information loss incurred for anonymizing that pair (lines 3-7). This fills the uppermost $|S|$ rows of the cost matrix with non-negative numbers. In many cases we have $|S| < |U|$ (i.e., number of agents and tasks differ) and hence the LSAP is unbalanced [49]. In these cases we add dummy suppression agents to the cost matrix (lowermost $|U| - |S|$ rows) to mark unmatched elements in $|U|$ which will eventually be suppressed by *diversify*. We capture the costs of suppressing elements in U on lines 8-10. The *cost* method in Section 5.1 can be used for this, and in that case, the cost of suppressing a subtree is equal to the total number of data entries (i.e., QIs) that are deleted from that subtree. On line 11 we solve the LSAP using the Hungarian algorithm, and consequently use this solution to compute the matching pairs of records in S and U that should be returned by *FindMapping*, while removing all dummy assignments (lines 13-15).

2.3.3 ℓ-diverse Clustering

Now that we can make a pair of records 2-diverse, we need to extend our strategy to support $\ell > 2$. We do so using a clustering algorithm. Let c denote a cluster. Each cluster contains:

- A class representative, denoted c_{rep}. This is a summary data structure that depicts the current state of the cluster. A formal definition of class representatives was given in Definition 3.14.
- A set of data records, denoted c_{init}, that are the original (i.e., unmodified) versions of the records in the cluster.

We first explain how we initialize and build one cluster. The procedure for this is given in Algorithm 4. Essentially, we treat a cluster as an equivalence class, and use a clustering algorithm to build ℓ-diverse equivalence classes. A cluster is initialized using one record, and at that point, it is a 1-diverse equivalence class. Then, we iteratively add new records to a cluster one by one, and each record that joins a t-diverse cluster makes it $(t+1)$-diverse. The process is terminated when the cluster becomes ℓ-diverse.

In Algorithm 4, a cluster c is initialized using one record on line 3. In order to satisfy ℓ-diversity, c needs to recruit $\ell-1$ other records, hence the loop on line 4. While recruiting new records, instead of randomly adding records to c, we aim to find the most suitable record in our database F for c. That is, we find T_b in F such that diversify(c_{rep}, T_b) would return the lowest cost (line 5). Once this record is found, it is removed from F and added to c (lines 6-7). Then, c_{rep} is updated on line 8: Generalizations and suppressions are performed, and sensitive values in T_b's vertices are added to matching vertices in c_{rep}.

Algorithm 4 Create ℓ-diverse cluster

```
Input: A collection of n data records $F(T_1, ..., T_n)$,
    parameter $\ell$ for $\ell$-diversity,
    data record $T_a \in F$
1: procedure CREATECLUSTER
2:     $F \leftarrow F - T_a$
3:     Initialize $c$, where $c_{\text{rep}} = T_a$ and $c_{\text{init}} = \{T_a\}$
4:     for $i = 2$ to $\ell$ do
5:         Find $T_b \in F$ s.t. $\text{argmin}_{T_b}$(diversify($c_{\text{rep}}$, $T_b$))
6:         $F \leftarrow F - T_b$
7:         $c_{\text{init}} \leftarrow c_{\text{init}} \cup T_b$
8:         Update $c_{\text{rep}}$ by diversify(clone($T_b$), $c_{\text{rep}}$) and copying sensitive values of
               matching vertices
9:     return $c$
```

We present our main clustering procedure in Algorithm 5 and refer to it as **ClusTree**. It receives a database of hierarchical data records F, a privacy parameter ℓ and two clustering parameters m and s that are both positive floating numbers. It employs Algorithm 4 as follows: It selects records in F to initialize and build new clusters. Based on the *quality* of the clusters that are built, *ClusTree* either accepts or rejects them. Next, we

Algorithm 5 Clustering algorithm

```
Input: A collection of $n$ data records $F(T_1, ..., T_n)$,
    parameter $\ell$ for $\ell$-diversity,
    maximum standard deviation multiplier parameter $m$,
    step size parameter $s$
 1: procedure CLUSTREE
 2:     $R \leftarrow \emptyset$
 3:     $\chi \leftarrow 0$
 4:     while true do
 5:         for each $T_a \in F$ do
 6:             if $|F| < \ell$ then
 7:                 suppress all $T_i \in F$
 8:                 return $R$
 9:             $c \leftarrow$ CreateCluster($F$, $\ell$, $T_a$)
10:             clcost $\leftarrow 0$
11:             for each $T_j \in c$ do
12:                 clcost $\leftarrow$ clcost + cost($T_j$)
13:             if $|R| > 1$ then
14:                 Let $\mu$ be the mean of costs of clusters in $R$
15:                 Let $\sigma$ be the standard deviation of costs of clusters in $R$
16:                 if clcost $> (\mu + \sigma * \chi)$ then
17:                     Add all $T_i \in c_{\text{init}}$ back to $F$
18:                     Discard cluster $c$
19:                 else
20:                     $R \leftarrow R \cup c$
21:             else
22:                 $R \leftarrow R \cup c$
23:             if $|F| = 0$ then
24:                 return $R$
```

```
25:     $\chi \leftarrow \chi + s$
26:     **if** $\chi \geq m$ **then**
27:         $\chi \leftarrow +\infty$
```

describe the details of this procedure.

ClusTree picks a T_a from the input F (line 5) and uses it to create a new cluster using *CreateCluster* (line 9). Once a cluster is formed, its total anonymization cost is calculated (lines 10-12), where the total cost is the sum of individual anonymization costs of all records within that cluster. At this point, we introduce our clustering heuristic. We suggest that the *quality* of a cluster (implied by its cost) depends significantly on the choice of initial record T_a. If T_a happens to be an outlier (e.g., has far less or higher number of vertices than every other record in F, or its QIs are very rare) then even the best T_bs joining T_a's cluster will incur high costs of anonymization. Therefore on lines 13-22 of *ClusTree*, we perform the following check: We compute the mean and standard deviation of previously formed clusters (lines 14-15). If the cost of the newly formed cluster c is significantly higher than the mean, it is discarded and all records in c are inserted back to the input F. Otherwise, c can be added to the output R. We use χ to limit the discrepancy between the cost of c and the mean cost of clusters in R (line 16). χ is initialized to 0 (line 3) and incremented by the step size parameter s (line 25) at each iteration. We run iterations of the clustering procedure until χ goes above m (line 26), and afterwards we run one final pass with $\chi = +\infty$ (line 27) to allow clusters with any cost. The output of the clustering algorithm is a set of ℓ-diverse clusters. Records in F that are not placed in any cluster in R are fully suppressed (lines 6-8). *ClusTree* terminates when less than ℓ records remain in the input F (lines 6-8 and 23-24).

A lower value of χ sets a stricter upper bound on the costs of accepted clusters. The rate at which χ increases is determined by the input clustering parameter s. s should be small enough that expensive clusters are rejected in the first few iterations, but also large enough that clusters which were rejected in the previous iteration have a chance of being accepted in the next iteration. Also, smaller s implies larger number of passes over the input database F (although F is consumed in each iteration) and would hence be more time-consuming. The upper limit parameter m can be determined by experiment. However, if one assumes that clusters' costs will approximately follow a uniform distribution, the probability of a value falling outside $\mu + 3 \cdot \sigma$ is significantly small (e.g., 99.7$\%$ of the samples in a normal distribution lie within 3 standard deviations of the mean). So, even in cases where costs are skewed or randomly distributed, a maximum upper limit of $m = 3$ or $m = 4$ should be reasonable.

2.3.4 Complexity Analysis

In this section we analyze the time complexity of our solution. We start with pairwise anonymization using *diversify* and *FindMapping*. Let our hierarchical data records have branching factor (number of children at each vertex) $b \geq 2$ and height (number of edges on the longest path between the root vertex and a leaf) h. For the sake of simplicity, we'll assume that all *mrca* operations, vertex generalizations and partial and full suppressions are performed in total time t per *diversify* call.

The greedy version of *FindMapping* requires $\frac{b \cdot (b+1)}{2}$ calls to *diversify* when called with two sets of subtrees. To anonymize all pairs of matched subtrees, *diversify* makes b recursive calls (lines 19-20 of Algorithm 1). Hence we obtain the following recurrence relation: $T(h) = \frac{b^2+b}{2} \cdot T(h-1) + b \cdot T(h-1) + t$ where $T(0) = t$. Solving this relation, we find that *diversify* with *FindMapping-GRD* is $\mathcal{O}(t \cdot \frac{b^{2h}}{2^h})$.

The LSAP version of *FindMapping* requires b^2 *diversify* calls to fill its cost matrix with agent-task costs (lines 3-7 in Algorithm 3), when called with two sets of subtrees. Then, finding an optimal solution to the LSAP using the Hungarian algorithm is $\mathcal{O}(b^3)$. Similar to above, *diversify* still makes b recursive calls to anonymize all pairs of matched subtrees. In this case we obtain the following recurrence relation: $T(h) = b^2 \cdot T(h-1) + t + \mathcal{O}(b^3) + b \cdot T(h-1)$ where $T(0) = t$. Solving this relation, we find that *diversify* with *FindMapping-LSAP* is $\mathcal{O}(b^{2h+1} + t \cdot b^{2h})$.

These results are significant in several ways. First, pairwise anonymization is exponential in h. Practical databases in real world, however, often have small h, e.g., $h = 3, 4$.

Therefore this is not a pressing concern. Second, finding an optimal solution to a LSAP comes at the price of introducing an additional $\mathcal{O}(b^{2h+1})$ factor in asymptotic complexity. Third, there is the cost of performing generalizations and suppressions, which we denote by t. The efficiency of these operations is dependent on their implementation. Some operations can be implemented in constant time (e.g., checking if two vertices are 2-diverse, suppressing a given subtree). In our experiments we saw that the factor t has significant impact on execution time, hence efficient implementation of generalizations and suppressions is key to scalability.

The complexity analysis of our clustering algorithm *ClusTree* is as follows: Let n be the number of hierarchical data records in the database, ℓ be the ℓ-diversity parameter, and m and s be the clustering parameters in *ClusTree*. The complexity of pairwise anonymization depends on whether *GRD* or *LSAP* mapping is used, as shown above. We denote it here by $\mathcal{O}$*(diversify)*. We provide a worst-case analysis. The worst case occurs when the first cluster created is the least costly cluster possible, and therefore no cluster is accepted afterwards until the final iteration.

After initializing a cluster with a record, *ClusTree* (Algorithm 5) tries finding $\ell - 1$ other records to join that cluster (Algorithm 4). This requires going over the remaining records in the database $\ell - 1$ times and calling *diversify*. Hence *CreateCluster* is $\mathcal{O}(n \cdot \ell \cdot$ *diversify)*. Calculating a cluster's cost (lines 10-12) can be done cumulatively within *CreateCluster* while the cluster is being formed, and there are online algorithms to compute mean and variance [50] so that computing and updating them when a new cluster is formed can be a constant time operation (lines 13-22). We therefore find that a single pass of *ClusTree* over its input database is $\mathcal{O}(n^2 \cdot \ell \cdot$ *diversify)*. A quick calculation shows that *ClusTree* performs $\lfloor \frac{m}{s} \rfloor + 2$ passes over the data, resulting in a time complexity of $\mathcal{O}((\lfloor \frac{m}{s} \rfloor + 2) \cdot n^2 \cdot \ell \cdot$ *diversify)*.

2.3.5 Proofs of Correctness

We now prove the correctness of the algorithm *ClusTree* (given in Algorithm 5), that is, we prove that the output of the algorithm is an ℓ-diverse anonymization of F. To do this, we first prove the correctness of the algorithm *diversify* (given in Algorithm 1) which acts as a building block in *ClusTree*.

Definition 17. *We say a class representative T is ℓ'-diverse if each vertex in T contains exactly ℓ sensitive values.*

Corollary 1. *If a class representative T is ℓ'-diverse then T is also ℓ-diverse.*

Theorem 1. *Let T_1 and T_2 be ℓ_1' and ℓ_2'-diverse class representatives of equivalence classes D_1 and D_2 respectively. Then diversify on T_1 and T_2 creates a $(\ell_1 + \ell_2)'$-diverse representative for the anonymization of T_1 and T_2.*

Proof. By induction:
Base Case: If the height of T_1 is 0, that is, T_1 is $\emptyset$, T_2 is suppressed. Since we will not have any vertex in T^*, T^* is a valid anonymization of both T_1 and T_2 and satisfies $(\ell_1 + \ell_2)'$-diversity.
Inductive Step: Let us denote a data record with height k as $T^{h=k}$. By inductive hypothesis, we assume *diversify* runs correctly for records with height at most $k - 1$. That is, *diversify* on ℓ_1'-diverse $T_1^{h=i}$ and ℓ_2'-diverse $T_2^{h=j}$ creates an $(\ell_1 + \ell_2)'$-diverse representative for $i, j \in [0, k - 1]$. We now prove the theorem for $T_1^{h=i}$ and $T_2^{h=j}$ where $i, j \leq k$.

We proceed with the proof as follows. We first show that the roots are properly diversified, that is, generalized to a $(\ell_1 + \ell_2)'$-diverse representative vertex (or suppressed if diversification is not possible). We then show that the children of both trees are properly mapped, paired and diversified.

Diversification of the Root: Let a, b, and t be the roots of $T_1^{h=i}$, $T_2^{h=j}$, and T^* respectively. Due to the anonymization process enforced by *diversify*, if $t \neq \emptyset$, we have $t_{SA} = a_{SA} \cup b_{SA}$. If $a_{SA} \cap b_{SA} \neq \emptyset$, then t_{SA} does not satisfy $(\ell_1 + \ell_2)'$-diversity. In such a case, *diversify*, at lines 4-6, suppresses $T_1^{h=i}$ and $T_2^{h=j}$ and subsequently $T^* = \emptyset$. Suppressed T^* is an $(\ell_1 + \ell_2)'$-diverse anonymization. If $a_{SA} \cap b_{SA} = \emptyset$, at lines 7-10, QI values in both roots are generalized into the nearest common parent, thus QI-isomorphism of T^* is ensured at root level. Since $t_{SA} = a_{SA} \cup b_{SA}$, $|a_{SA}| = \ell_1$, and $|b_{SA}| = \ell_2$ then t_{SA} will contain $\ell_1 + \ell_2$ sensitive values, thus satisfies $(\ell_1 + \ell_2)'$-diversity.

Diversification of the Children: Let $C_A = \{A_1, \ldots, A_m\}$, $C_B = \{B_1, \ldots, B_n\}$, $C_T = \{T_1, \ldots, T_m\}$ are the subtrees attached to a, b, and t respectively and $m \leq n$. If C_A is empty, *diversify*, at lines 11-17, suppresses all trees in C_B, consequently $C_T = \emptyset$. In such a case, T^* will be composed of a single already-diversified root, thus satisfies $(\ell_1 + \ell_2)'$-diversity. If C_A is not $\emptyset$, either of the *FindMapping* functions are called. Both algorithms guarantee that every $A_i \in C_A$ is paired with some unique $B_j \in C_B$. *diversify*, at lines 19-20, calls itself recursively on the paired subtrees. Note that due to omission of the root, all subtrees in C_A and C_B have heights less than k. By the inductive hypothesis, for every pair (A_i, B_j) matched, *diversify* correctly returns $(\ell_1 + \ell_2)'$-diverse anonymization T_i of A_i and B_j. If there exists any unpaired subtree, *diversify*, in lines 21-23, suppresses it. Since the root is already-diversified, all vertices in T^* satisfies $(\ell_1 + \ell_2)'$-diversity. □

Theorem 2. *Algorithm CreateCluster, when given records $F = \{T_1, \ldots, T_n\}$, $\ell \leq n$, $T_a \in F$ returns a cluster c where c_{rep} is ℓ'-diverse and $T_a \in c_{init}$.*

Proof. At start, representative c_{rep} is initialized to T_a satisfying 1-diversity. At iteration i of the for loop, $(i-1)'$-diverse c_{rep} is diversified with a 1-diverse record and by Theorem 1, the resulting representative which is assigned to c_{rep} satisfies i-diversity. At the end of the loop, c_{rep} satisfies ℓ'-diversity (thus, ℓ-diversity) and c_{init} contains the associated equivalence class. □

Theorem 3. *ClusTree, when called on records $F = \{T_1, \ldots, T_n\}$, returns an ℓ-diverse anonymization of F.*

Proof. The *ClusTree* algorithm is basically a loop where at each iteration the following is performed:

- At lines 5-22, *ClusTree* scans all records currently in F once and for each record, the function *CreateCluster* creates a single ℓ-diverse cluster. Between lines 16-28, if the quality of the previously-formed cluster is far away from normal parameters, the cluster is discarded. Otherwise, it is moved from F to the result list. Distance threshold on the quality is controlled by the parameter χ.

- χ is incremented, and after reaching m it is set as ∞.

The algorithm halts only when there are less than ℓ records not clustered, in which case these records are suppressed. Due to correctness of *CreateCluster*, if the algorithm terminates, every record in F (except the few suppressed ones) belongs to exactly one cluster (equivalence class). Thus, by Definition 16, the returned clusters and the corresponding equivalence classes give an ℓ-diverse anonymization of the original records.

We conclude by stating that the algorithm always halts, that is, we will eventually have $|F| < \ell$. Note that the distance threshold χ that decides whether to discard a previously-formed cluster is monotonically increasing with each iteration of the while loop. After reaching m, χ is set to ∞. When this happens, no cluster will be discarded, thus every cluster formed by *CreateCluster* function is removed from F. Since *CreateCluster* is called on every record in T_a, we will eventually be left with few enough records in F and the algorithm returns. $\square$

2.4 Experiments

We implemented our algorithms in Java 8 and used MongoDB 2.4 to store our datasets. Experiments were conducted on a commodity machine with Intel Core i7 2.40 GHz CPU and 16 GB RAM.

Evaluation metrics. We use three means of evaluation: *LM cost*, *average query accuracy* and *KL-divergence*. *LM* outputs a numerical value between 0 and 1 that conveys the average cost of generalizations and suppressions over the whole database. Lower *LM cost* implies higher data utility and therefore preferable anonymization.

For measuring *query accuracy*, we randomly generate several aggregate count queries (e.g., "How many students took *CS301*?" or "How many *CS* courses were taken in total?"). We issue these queries on the original dataset (X_i denotes the result of the ith query) and the anonymized dataset (Y_i denotes the result of the ith query). Then, average query accuracy is computed as follows (where N is the total number of queries):

$$\frac{\sum_{i=1}^{N}(1-(\frac{|Y_i - X_i|}{X_i}\cdot 100\%))}{N}$$

As a third metric, we employ the Kullback-Leibler divergence (KL-divergence, in short) as follows: We find the distribution of sensitive values in the original and anonymized dataset. Let us call these probability distributions Q and P, respectively. For example, the distribution of *letter grades* in the original dataset could be 20% A, 25% A- etc., but due to suppressions, this distribution may change to 25% A, 15% A- etc. in the anonymized dataset. The KL-divergence of Q from P is defined as [51]:

$$D_{KL}(P||Q) = \sum_i (P(i) \cdot log\frac{P(i)}{Q(i)})$$

A smaller KL-divergence implies that P and Q are closer to one another, and hence the statistical properties of the data are better preserved after anonymization. Thus, similar to LM cost, it is more desirable to have smaller KL-divergence. We measure the KL-divergence of all sensitive values (e.g., GPA, letter grades and book prices) one by one, and report the average. Since generalizations do not affect sensitive values, only suppressions cause changes in their distribution. Therefore, KL-divergence is a good way to evaluate the effect of suppressions on the statistical properties of a dataset.

Datasets. We report results on three datasets (two *synthetic* and one *real*) obtained from two different universities in Turkey. Both datasets share a similar schema to that in Figure 1.2.

For the *synthetic* datasets, we obtained data regarding students from Sabanci University's Computer Science (CS) program. The data contained the GPA and (partial) course grades of 30 students from this year's graduating class. To test with a meaningful number of data records, we simulated several students based on this sample, with the guidelines explained in the next paragraph.

We assumed that approximately the same number of students graduate every year, and set their current age according to their year of graduation. We simulated GPA values using a normal distribution, where the mean and the standard deviation were determined by the GPA scores of our sample. According to Sabanci University's CS program requirements, we ensured that all students took the obligatory courses. To each student, we randomly assigned a fixed number of courses from the pool of core courses, and a varying number of technical area electives. Students' grades were determined by their GPA and the type of course (e.g., we observed that most students perform better in obligatory courses). We assumed that a student would buy 0 to 2 books for each course.

We created two synthetic datasets, *syntheticT* and *syntheticS*, both containing 1000 students with approximately 20 courses per student. *syntheticS* uses the schema in Figure 1.2, i.e., (major,YoB) $\rightarrow$ courses $\rightarrow$ books. In order to test with an increased height, in *syntheticT* we added an intermediate level between the root and the courses, that depicts the year in which courses were taken, i.e., freshman, sophomore, junior or senior. Therefore the schema in *syntheticT* is: (major,YoB) $\rightarrow$ college years $\rightarrow$ courses $\rightarrow$ books. The division of courses into college years was probabilistic based on whether the course is a pre-requisite for any of the other courses the student took, and the usual timeframe in which the course is actually taken at Sabanci University.

The *real* dataset contains 3162 students together with their years of birth, their GPA, the courses they took and the grades they received. So, records in this dataset have only the first two levels shown in Figure 1.2. Furthermore, instead of the QI attribute *age*, we used *year of birth*. We set DGHs of courses according to their IDs.

Algorithms. We evaluate five approaches, four of which are presented in this thesis. For these, we used the LM metric as the anonymization cost metric in Algorithm 1. We tested *ClusTree* with the greedy and LSAP-based implementations of *FindMapping*. We call the resulting methods *ClusTree-GRD* and *ClusTree-LSAP*, respectively. Regarding the parameters of *ClusTree*, we set $s = 0.5$ since we saw that values below 0.5 did not make an observable difference, and we obtained the best results with $m = 4$.

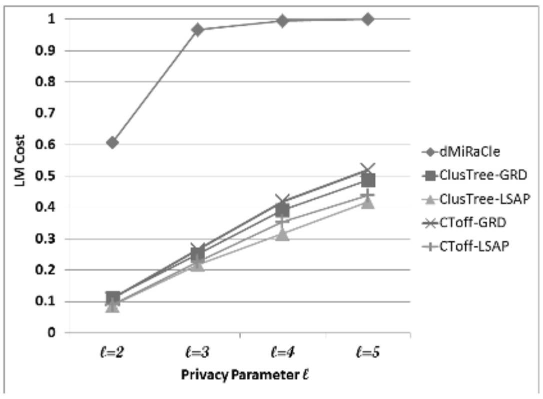
To demonstrate the effectiveness of our clustering heuristic, we implemented a second procedure that does not contain any checks regarding the costs of clusters, i.e., we initialize $\chi = +\infty$ on line 3 of Algorithm 5. We refer to this implementation as *CToff*.

There are also two versions of *CToff*: *CToff-GRD* and *CToff-LSAP*, depending on whether *FindMapping* is greedy or LSAP-based.

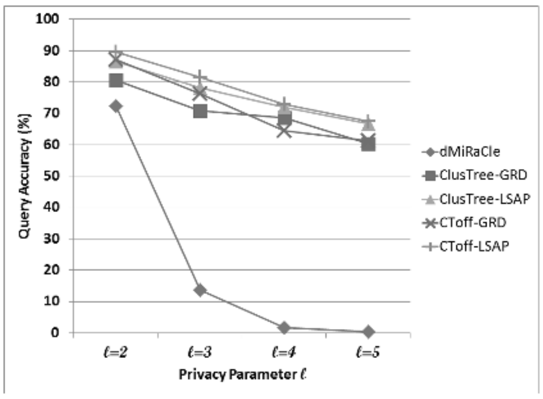
In addition, we implemented the multi-relational k-anonymity algorithm together with its ℓ-diversity extension (*dMiRaCle*) proposed in [6]. To the best of our knowledge, this is the only algorithm that can provide similar privacy guarantees to ours (i.e., ℓ-diversity) in hierarchical data. We converted our datasets into multi-relational databases and ran *dMiRaCle* on them. There are two parameters in *dMiRaCle*: *climit* (a limit on the number of active clusters allowed during the algorithm) and *th* (a distance threshold parameter used in *dMiRaCle*'s clustering phase). In our tests, we exactly mimicked the parameters suggested by the authors: We set *climit* to be 150 and tested with $th \in [0, 1]$ with increments of 0.1, and picked the best-performing result to report in this thesis.

Results and Discussion. We graph our results for varying values of ℓ in Figure 2.3, Figure 2.4 and Figure 2.5 on the synthetic datasets and the real dataset, respectively. In all experiments, we observe that LM cost and KL-divergence increase and query accuracy decreases as privacy requirements get stricter, i.e., ℓ is increased from 2 to 5. Two factors contribute to the loss of data utility when ℓ is increased: (1) The anonymization algorithm needs to find ℓ records for each cluster, i.e., higher ℓ requires more records per cluster. Each record that joins a cluster causes generalizations and/or suppressions. These anonymization operations are never reverted at a later point (e.g., when a new record joins a cluster), therefore the cost of a cluster always accumulates. (2) For large values of ℓ, it is harder to find ℓ different sensitive values per vertex. Consider a case where the instructor of *CS306* decided to grade very generously and all students received either *A* or *A-* from this course. When $\ell = 3$, *CS306* courses may never be matched with each other simply because there are only 2 different grades observed in the database. Hence, either all occurrences of *CS306* have to be suppressed, or they will be generalized with other courses (e.g., *CS3*** courses would be the best candidates) so that they become 3-diverse in the output.

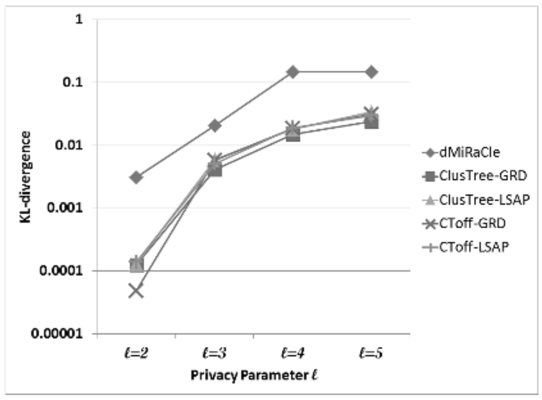
We also observe that our algorithm outperforms *dMiRaCle* by a great margin in every experiment. As explained in Section 2, [6]'s *dMiRaCle* is primarily concerned with k-



(a) LM cost

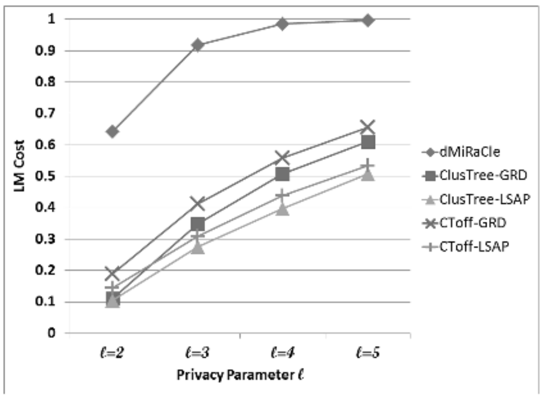


(b) Query accuracy

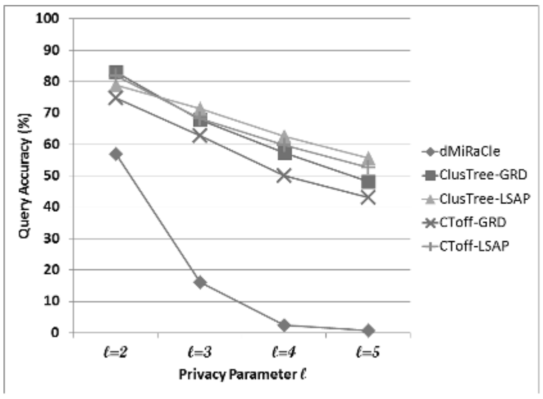


(c) KL-divergence

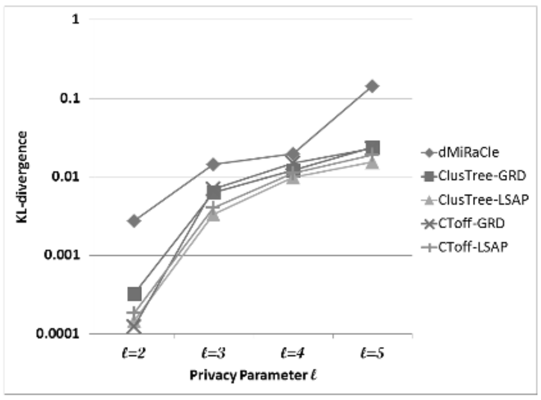
Figure 2.3: Results on the *syntheticS* dataset for $\ell = 2, 3, 4, 5$



(a) LM cost



(b) Query accuracy



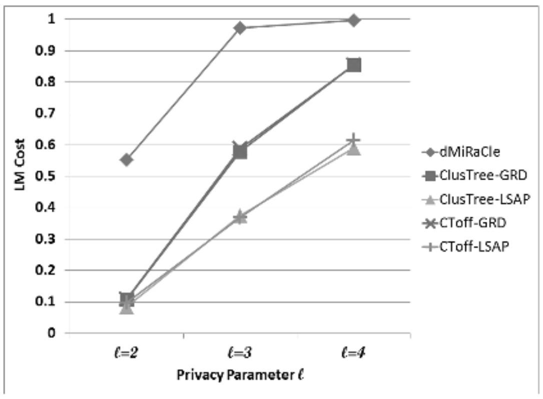
(c) KL-divergence

Figure 2.4: Results on the *syntheticT* dataset for $\ell = 2, 3, 4, 5$

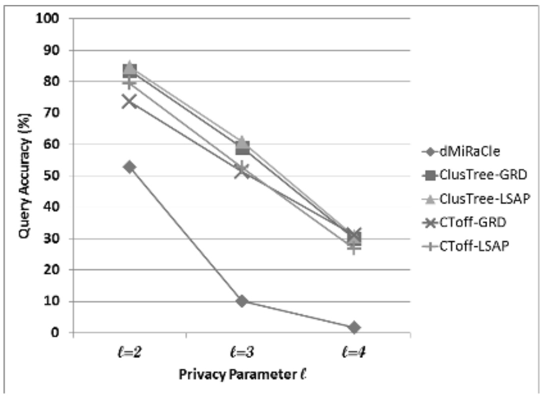
anonymity, and its ℓ-diversity extension depends on k-anonymizing an input dataset first and then finding an ℓ-diverse output. This can be a reasonable strategy when ℓ is small (e.g., $\ell = 2$), since a 2-anonymous equivalence class can, by coincidence, happen to be 2-diverse (or, making it 2-diverse might require very few operations). However, when $\ell = 3$ or 4, if the initial equivalence class is not built with ℓ-diversity in mind, later operations to make it ℓ-diverse will be very costly. Our experiments demonstrate this: There is a sharp increase in LM cost and a sharp decrease in query accuracy (in all three datasets) when ℓ is increased from 2 to 3.

We obtained better results on the synthetic datasets compared to the real dataset. We believe that this is caused by the fact that the real dataset is more sparse (e.g., there are 5000 unique courses, some of which are taken by very few students) and has more variance (e.g., some students took only 1-2 courses, whereas others took 60-70). In contrast, the synthetic datasets are more evenly distributed, e.g., all students are CS majors that take around the same number of courses, most of which are courses in Computer Science or related areas. Also, we obtain roughly 10-15% better results on *syntheticS* compared to *syntheticT*. The probable cause for this is the division of courses into college years in *syntheticT*. For example, consider two students $S1$ and $S2$ that take the elective course *EL101*, but $S1$ takes *EL101* in her freshman year whereas $S2$ takes *EL101* in her senior year. Unless *diversify* decides to match $S1$'s freshman year with $S2$'s senior year (which is a small probability, assuming $S1$'s freshman courses are more similar to $S2$'s freshman courses rather than her senior courses) the *EL101* vertices will not be matched with each other. Instead, they will be matched with other courses or suppressed, due to the top-down nature of *diversify*. The *syntheticS* dataset does not suffer from this problem, since courses are directly children of the root vertices, and are not divided into college years.

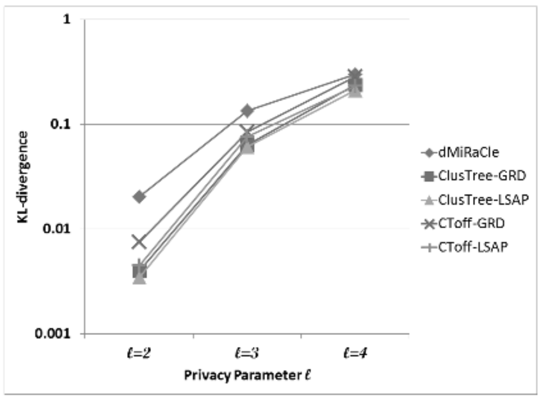
We also would like to study the effects of our heuristics, by comparing (1) *ClusTree* versus *CToff* to validate that our clustering heuristic is useful, and (2) *LSAP* versus *GRD* to validate the effect of using an optimal solution against a greedy solution. In most experiments, we see that *ClusTree* outperforms *CToff* and *LSAP* outperforms *GRD*, as expected. The difference between *LSAP* and *GRD* is usually more evident when ℓ is large,



(a) LM cost



(b) Query accuracy



(c) KL-divergence

Figure 2.5: Results on the *real* dataset for $\ell = 2, 3, 4$

apart from the $\ell = 4$ case on the *real* dataset, since most of the data in this experiment is destroyed no matter which algorithm is used. Also, although our *LSAP* approach provides an optimal solution to the subtree matching problem, neither *ClusTree* nor *CToff* guarantee optimality in the clustering phase - as in any clustering algorithm. Therefore we cannot claim that *ClusTree-LSAP* or *CToff-LSAP* are optimal or they should outperform their greedy counterparts in all experiments. In most experiments they do, which is intuitive, but there are also a few cases where the *GRD* approach performs almost as good as or somewhat better than *LSAP*. This happens often when ℓ is small, and in particular the KL-divergence experiment on the *syntheticS* dataset.

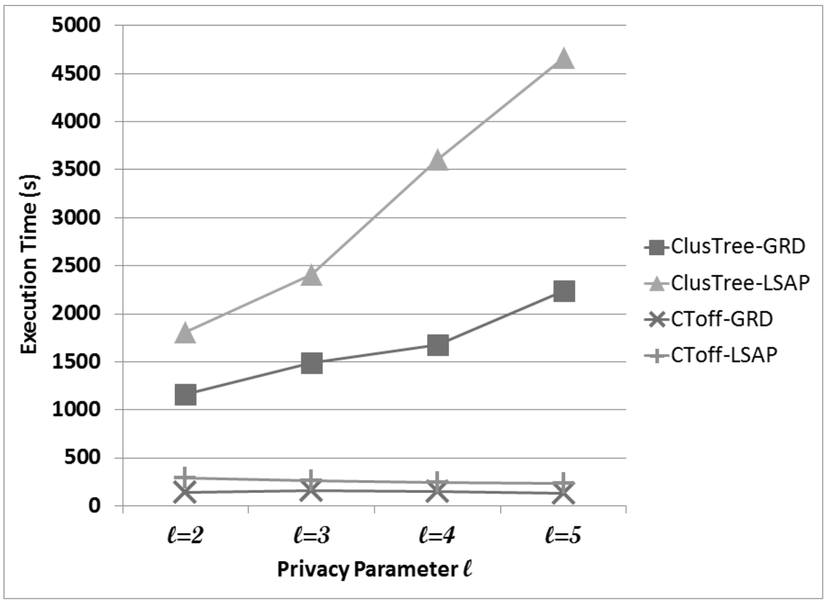


Figure 2.6: Execution time on the *syntheticS* dataset

With regard to efficiency, we obtained the execution times in Figure 2.6 on *syntheticS*. This dataset contains 1000 hierarchical data records (with height = 3) and a total of approximately 42000 vertices. *CToff* is significantly faster than *ClusTree*, since it performs a single pass over the data. For all ℓ values, it took 2-3 minutes to run *CToff-GRD* and 3-4 minutes to run *CToff-LSAP*. Since *ClusTree* performs multiple passes over the data (with increasing χ), it turns out to be roughly 10-15 times slower than *CToff*. The differences between execution times become more significant as ℓ is increased. All of these results are in line with our complexity analyses.

Finally, we would like to emphasize the trade-offs between data utility and efficiency. The choice of using *ClusTree* over *CToff* and *LSAP* over *GRD* both increase data utility, but come at the cost of increased execution time. On average, the best-performing algorithm (in terms of query accuracy and LM cost) is *ClusTree-LSAP*, which also happens to be the slowest.

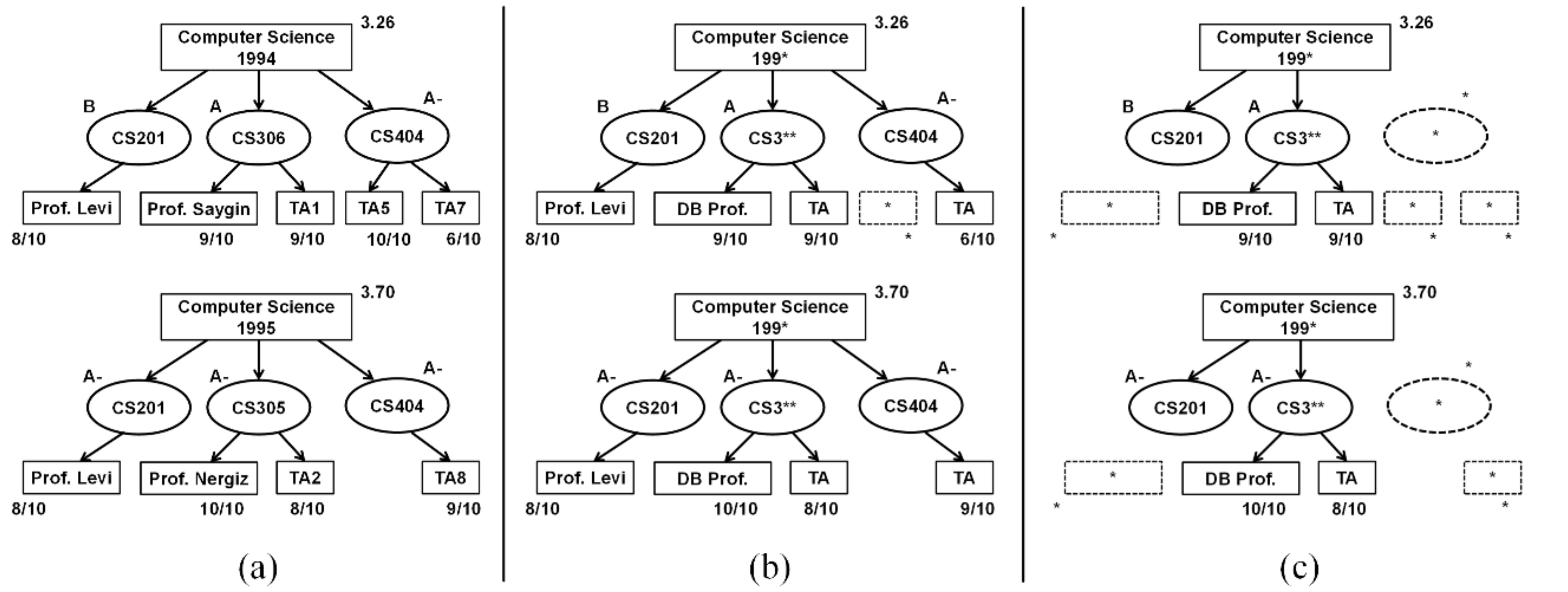


Figure 2.7: (a) Hierarchical data records for two sample students. (b) A 2-anonymous version of these records. (c) A 2-diverse version of these records

Chapter 3

Privacy Preserving Anatomization of Hierarchical Data

3.1 Overview

In Chapter 2, we used the method of generalization to protect the privacy of individuals in shared hierarchical databases. While the proposed technique is effective, we still face two major issues not adequately addressed by ℓ-diverse generalization of hierarchical data. In this chapter, we develop solutions for each of these issues as follows:

Utility of the released data: Over-generalization incurs heavy loss of utility especially when a natural domain generalization hierarchy is not present or difficult to find. The loss of utility is further amplified due to the high dimensional data.

How to preserve utility while protecting privacy has been widely studied for the anonymization of tabular data. One effective technique in this domain is the anatomy method which guarantees better utilized anonymizations compared to the traditional generalization techniques. Instead of applying value generalizations, anatomy breaks the link between QI and sensitive attributes associated with individuals in a given group. As an example, in Table 3.1, we show the private Table T, its 2-diverse generalization T^*, and its 2-diverse anatomization T^A. Note that T^A is composed of a QI table and an associated SA (sensitive attribute) table. While both tables can be joined via the group id

Table 3.1: Generalization and anatomization on sample tabular data

(a) Private data table T

Id	**Age**	**Sex**	**Nationality**	**Disease**
1	42	M	Ukrainian	**Hearth Disease**
2	46	M	Romanian	**Flu**
3	54	F	Turkish	**Bronchitis**
4	51	F	Italian	**Dyspepsia**
5	24	M	German	**Pneumonia**
6	22	F	German	**Gastritis**

(b) ℓ-diverse generalization T^*

Id	**Age**	**Sex**	**Nationality**	**Disease**
1	[40-50]	M	Eastern European	**Hearth Disease**
2	[40-50]	M	Eastern European	**Flu**
3	[50-55]	F	Mediterranean	**Bronchitis**
4	[50-55]	F	Mediterranean	**Dyspepsia**
5	[20-25]	M	German	**Pneumonia**
6	[20-25]	F	German	**Gastritis**

(c) QI table for anatomization T^A

Age	**Sex**	**Nationality**	**Group Id**
42	M	Ukrainian	**1**
46	M	Romanian	**1**
54	F	Turkish	**2**
51	F	Italian	**2**
24	M	German	**3**
22	F	German	**3**

(d) SA table for anatomization T^A

Group Id	**Disease**	**Count**
1	**Hearth Disease**	1
1	**Flu**	1
2	**Bronchitis**	1
2	**Dyspepsia**	1
3	**Pneumonia**	1
3	**Gastritis**	1

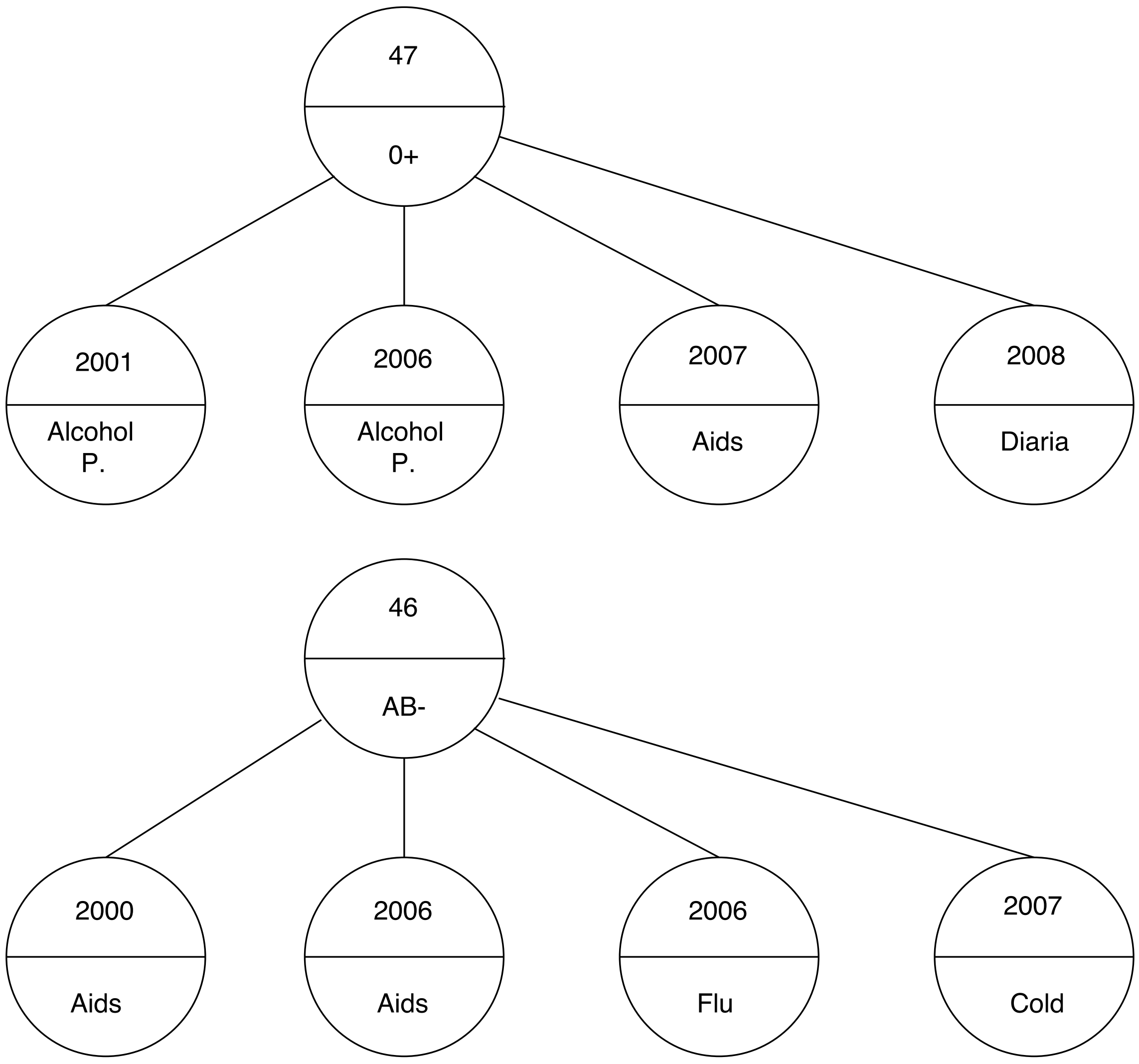


Figure 3.1: Example tree data

attribute, within a group, which QI row is associated with which SA row is hidden. For example, an adversary knowing that his 42 years old, male, Ukranian friend is in the released anatomization can at best link him to hearth disease or flu with equal probabilities. Thus, the released data respects 2-diversity. Note that 2-diverse anatomization T^A offers a richer information content than 2-diverse generalization T^* which groups the same set of individuals as T^A does.

Despite the apparent advantage of anatomy on utility, no work has been done to apply anatomization to hierarchical data. In this chapter, we show how one can create privacy-preserving anatomization of a given hierarchical dataset and experimentally demonstrate

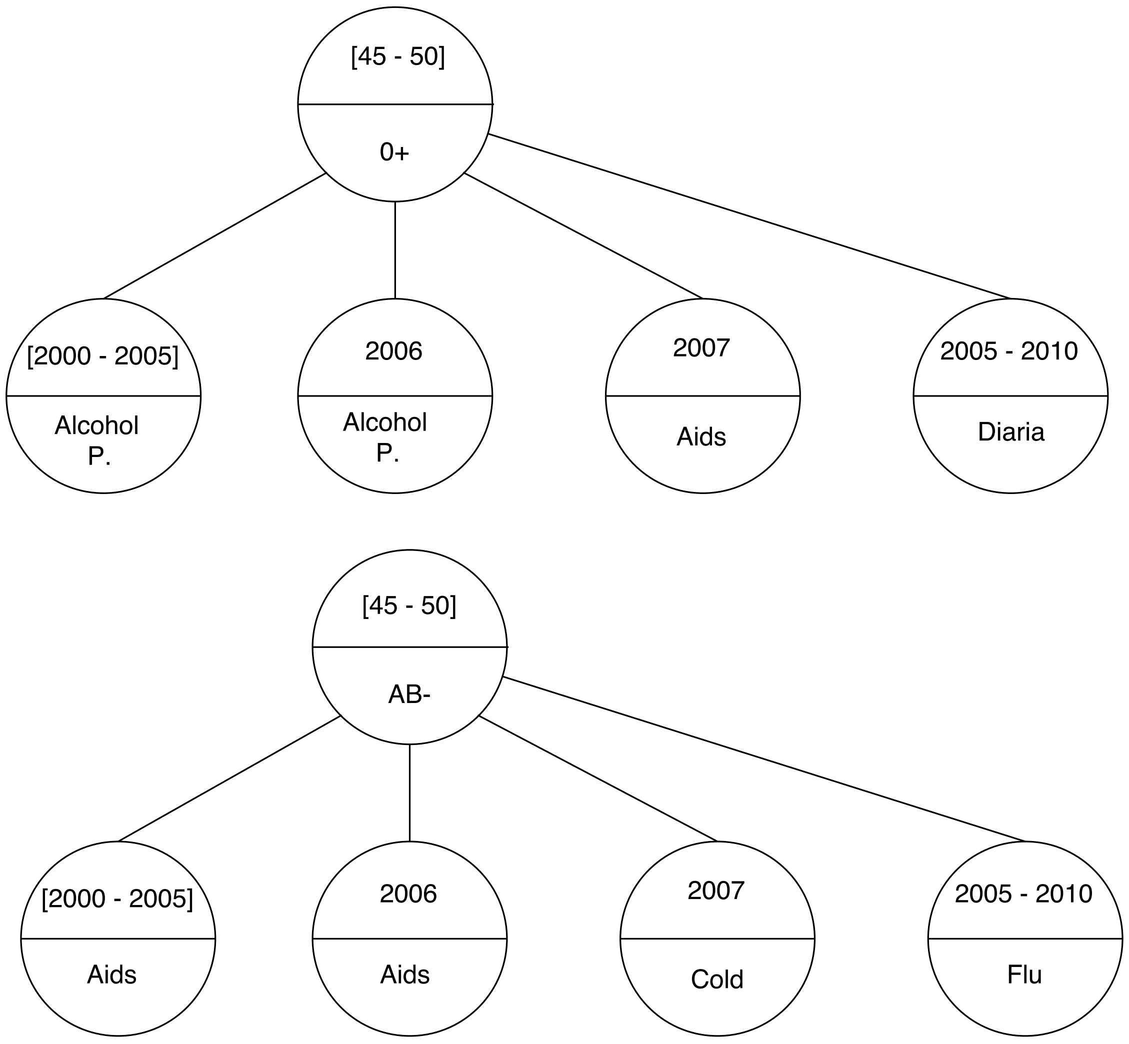


Figure 3.2: ℓ-diverse result

that anatomy can significantly increase the utility of released hierarchical databases.

Multi-sensitive values linked to individuals: ℓ-diversity is initially designed for tabular private databases in which each individual is associated with a single sensitive value. However, hierarchical data belonging to an individual may exhibit multiple sensitive value from the same domain. This introduces a unique privacy attack even if the released hierarchical data respects ℓ-diversity. To demonstrate the attack, consider the private database, in Figure 3.1, showing the blood types and health records of two patients in a given hospital. (E.g., the second tree belongs to an 46 years old patient with AB- blood type. The patient has been diagnosed as having AIDS twice in 2000 and 2006; while diagnosed as suffering from Flu and Cold in 2006 and 2007, respectively.) We also show, in Figure 3.2, a 2-diverse generalization of both patients. Note that an adversary knowing that her 46 years of friend visited the hospital in 2006 can at best map him to one of the generalized trees. The generalization respects 2-diversity in that the respective diagnosis in 2006 can be either Alcohol poisoning or AIDS with equal probability 0.5. While the diagnosis in each distinct visit demonstrates the required diversity, unfortunately the same cannot be said when we consider multiple visits. Note that, both trees contain AIDS, a disease not treatable. Regardless of when the diagnosis happened, the attacker will conclude that his friend suffers from AIDS.

Due to inadequate privacy offered by classical ℓ-diversity, in this chapter, we propose a new privacy metric (p, m)*-privacy* that limits an attacker's confidence in linking at most m instance of a particular sensitive attribute to an individual over multiple nodes of anonymous trees. We propose an anatomization algorithm to enforce the new metric and experimentally show that (p, m)-privacy can be achieved with high utility in released hierarchical databases.

3.2 Anatomization of Hierarchical Data

ℓ-diversity states that the observed frequency of any sensitive value in a multiset or group must be bounded by $1/\ell$. We note that SAs can be categorical or continuous: The domains of categorical SAs consist of discrete values, and the definition above applies trivially. Continuous SAs require an intermediate discretization step, in which their domains are divided into non-overlapping buckets. A uniform discretization is often assumed, however, we designed our algorithms to accommodate arbitrary discretizations that meet the preferences of the data publisher.

As demonstrated in Section 3.1, despite ℓ-diversity, the nature of hierarchical data allows linkage attacks in which: (i) an individual can be linked to a sensitive value with high probability, and (ii) an individual can be linked to having multiple occurrences of a sensitive value. For some sensitive values the latter may not constitute a privacy risk, e.g., linking Alice to *flu* with high probability, or Alice being diagnosed with flu multiple times may not be a problem. However, some sensitive values, such as *AIDS* or *alcohol poisoning* are discriminatory by nature. Therefore we need to limit an adversary's confidence in linking a discriminatory value with an individual.

We introduce the (p, m)-privacy principle for this purpose. First, since anatomy works by disassociating QIs and SAs, reconstruction of the data and therefore linkage attacks are probabilistic (hence the parameter p). Second, for some discriminatory sensitive values, their number of occurrence is important (hence the parameter m). For example, a single occurrence of *AIDS* might be sufficient to discriminate against an individual, but it might take more than two occurrences of *alcohol poisoning* to deduce alcohol addiction problems and consequently discriminate.

Definition 18. ***(Discriminatory rule)*** *A discriminatory rule is a triplet* (s_i, m, p) *where* s_i *is a sensitive value within the domain of a SA,* m *is the maximum number of allowed occurrences of* s_i *in any tree, and* p *is a probability (confidence threshold)* $0 < p \leq 1$.

For example, the verbal rules given above could be written as: $(\textit{AIDS}, 1, 1/2)$ and $(\textit{alcohol poisoning}, 2, 1/3)$.

Definition 19. ***((p,m)-privacy)*** *We say that a publication of the private data* F *satisfies* (p, m)*-privacy if for all discriminatory rules* (s_i, m, p) *and an adversary observing the published data:*

$$\forall T \in F, \quad Pr[\textit{Adversary links } T \textit{ to } \geq m \textit{ occurrences of } s_i] \leq p$$

Informally, (p, m)-privacy states that the published data should not cause an adversary to discriminate against an individual with confidence greater than p. For example, for any $T \in F$, $Pr[T$ is linked to AIDS$] \leq 1/2$ and $Pr[T$ is linked to at most 2 occurrences of alcohol poisoning$] \leq 1/3$. Note that, per the definition above, the confidence of 1 occurrence of alcohol poisoning can be $> 1/3$, and this would not yield a violation of (p, m)-privacy.

3.3 Anatomization Techniques

We developed two techniques based on the disassociation of QIs and SAs for privacy preserving publishing of hierarchical data records. Per the naming in the seminal work of Xiao and Tao studying disassociation on tabular data [41], we refer to our techniques as anatomization techniques.

Our two techniques are called *tree-by-tree anatomy* (*t-t anatomy*) and *vertex-by-vertex anatomy* (*v-v anatomy*) due to their grouping and publishing strategy. Both techniques satisfy ℓ-diversity and (p, m)-privacy, but the application and enforcement of these privacy notions are inherently different. Hence, as the experiments in Section 3.5 will show, one technique may outperform the other with respect to the information loss in its output, based on the values of parameters ℓ, p and m.

3.3.1 t-t Anatomy

The general idea behind t-t anatomy is to first distribute each record $T \in F$ into one of the many groups. We discuss how groups are formed in coming sections. Each group has a unique group ID, and it is modified to enforce ℓ-diversity and (p, m)-privacy. When a group is published, the records in that group are divided into two separate trees: one for QIs (called *QI-trees*) and one for SAs (called *SA-trees*).

We illustrate this process in Figure 3.3 and introduce some notation. Let the trees in Figure 3.1 be located in the group with group ID G_0. To enforce ℓ-diversity and (p, m)-privacy, we perform some modifications (suppressions and vertex reordering) on the trees. Before publication, we separate the QIs and SAs by dividing G_0's trees into QI-trees and SA-trees as shown. We use the *projection* operation from the database literature to denote this operation, e.g., $\pi_{QI}(T)$ outputs the QI-tree of T, and $\pi_{SA}(T)$ outputs the SA-tree of T. Note that from the published version, an adversary equipped with arbitrary information regarding an individual's QIs is not able to locate that individual's SA-tree (and hence sensitive values) with probability larger than $1/|G_0|$, where $|G_0|$ is the cardinality of G_0. Given the published QI-trees and SA-trees, reconstruction of the original data is probabilistic, using the permutation operation which we denote by $\bowtie$. Note that in the general case, for a group cardinality of g, there exist $g!$ different permutations. For any one individual T, the probability that the adversary correctly links the QIs and SAs of an individual through reconstruction is $1/g$.

Enforcement of ℓ-diversity requires structural isomorphism as well as sufficient diversity in sensitive values. An adversary with complete knowledge of his victim's QIs will only be able to link an individual to a group ID (say G_0) under which there must exist at least ℓ SA-trees as a pre-requisite for ℓ-diversity. Then, for each vertex, we check if the matching vertices in the SA-trees (i.e., vertices that appear in the same index of the ordered tree traversal) satisfy ℓ-diversity. This is formalized in the following definition.

Definition 20. ***(ℓ-diversity for t-t anatomy)*** *Let G be a t-t anatomous publication of a group of QI-trees and SA-trees, $\mathcal{R}(T_i) = \{v_1^i, v_2^i, .., v_m^i\}$ denote the breadth-order traversal of SA-tree T_i, and $\mathcal{I}_j = \bigcup_{i \in [1,|G|]} v_j^i$ denote the set of vertices at the j'th index of the traversal, as in the definition of isomorphism. G satisfies ℓ-diversity, if:*

1. *The number of QI-trees and SA-trees is equal, and $\geq \ell$.*
2. *QI-trees are structurally isomorphic, and so are SA-trees.*
3. *For all $j \in [1, m]$, the set consisting of the sensitive values (v_{SA}) of the vertices in $\mathcal{I}_j$ satisfies ℓ-diversity.*

As an example, consider the 2-diverse anatomization given in Figure 3.3. Rule 1 and 2 can easily be verified as there are exactly two structurally isomorphic trees in QI and SA groups. For rule 3, I_1 represents the root nodes of the trees. Corresponding set of sensitive {0+, AB-} satisfy 2-diversity. Similarly, for I_2, I_3, and I_4, the corresponding sets are {Alcohol P., AIDS}, {Alcohol P., Flu}, and {Diaria, Cold} respectively. All sets respects 2-diversity, thus rule 3 holds as well.

Enforcement of (p,m)-privacy. Let $\mathcal{S}$ denote the list of all discriminatory rules. To enforce (p, m)-privacy, we check each rule $(s_i, m, p) \in \mathcal{S}$ one by one. First, we find the number of SA-trees in G which would violate the rule if they were linked with an individual: Let $c(T, s_i)$ denote the number of observances of s_i in SA-tree T. Then, we define $\mu(G, s_i, m)$ as the number of SA-trees $T \in G$ such that $c(T, s_i) \geq m$. Second, recall that an adversary's confidence in linking an individual to a particular SA-tree is $^{1}/_{|G|}$. Linking to any one of the μ trees causes a privacy violation. Therefore, we arrive at the following: If $^{\mu(G, s_i, m)}/_{|G|} \leq p$, the discriminatory rule is satisfied; else, the rule is violated. This is formalized below.

Definition 21. ***((p,m)-privacy for t-t anatomy)*** *Let G be a t-t anatomous publication of a group of QI-trees and SA-trees, and $\mathcal{S}$ denote a list of discriminatory rules. G satisfies (p, m)-privacy, if for each rule $(s_i, m, p) \in \mathcal{S}$, we have:*

$$\frac{\mu(G, s_i, m)}{|G|} \leq p$$

As an example, t-t anatomization in Figure 3.3 satisfies (AIDS, 1, 0.5) rule. Note that there is only 1 out of 2 SA trees having at least one occurrence of AIDS. Similarly, the anatomization also satisfies (Alcohol P., 2, 0.5) rule.

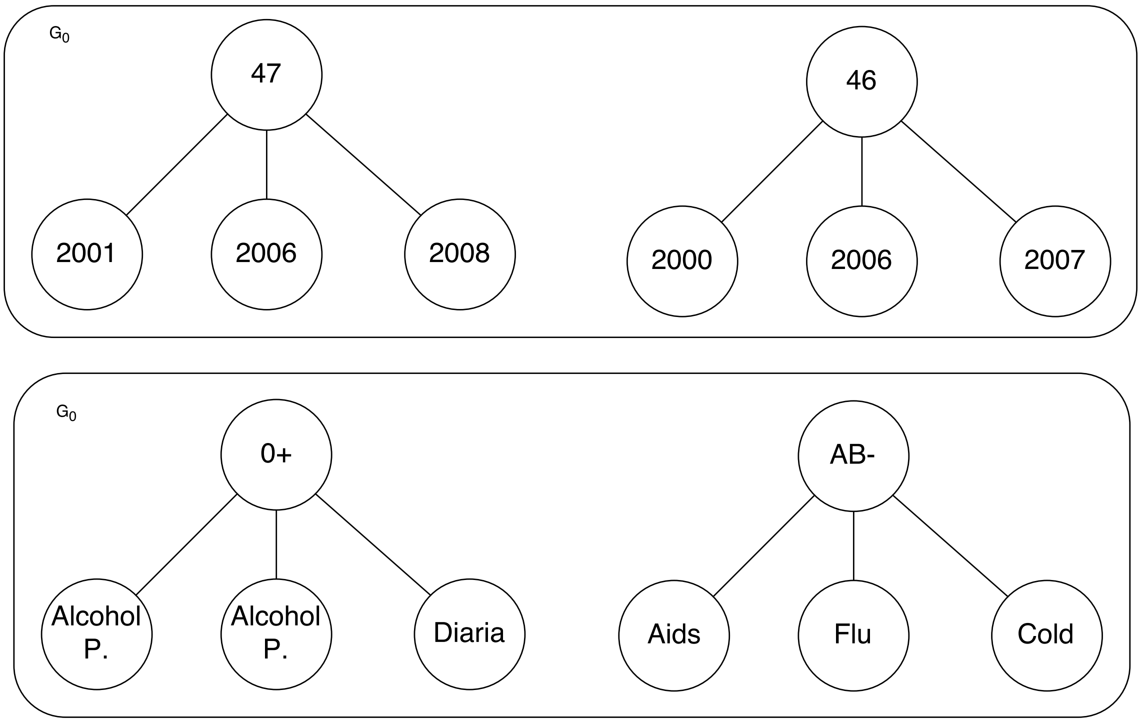


Figure 3.3: t-t anatomy result, QI and SA trees

Finally, we say that a t-t anatomous publication of the whole data is privacy preserving, if for all groups G_i containing QI-trees and SA-trees, ℓ-diversity and (p, m)-privacy are satisfied.

3.3.2 *v*-*v* Anatomy

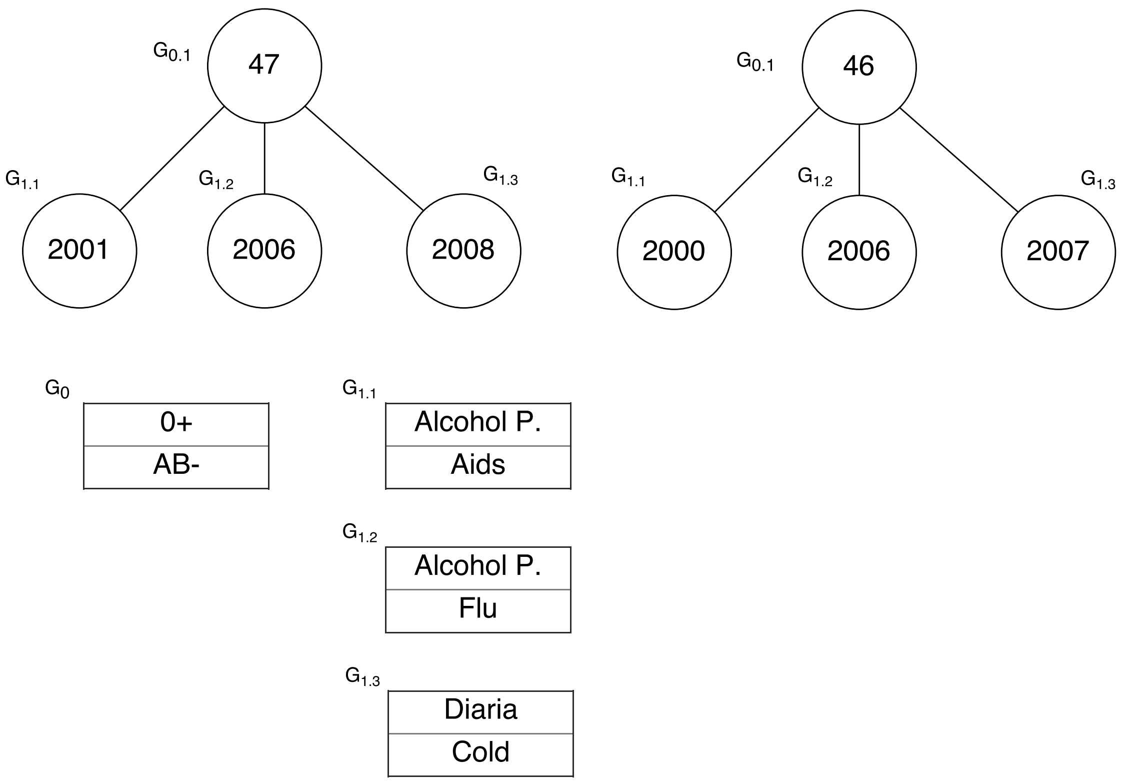


Figure 3.4: v-v anatomy result, QI trees and SA groups

In t-t anatomy, both QI-trees and SA-trees follow a tree structure where each piece of information is explicitly linked to the other with tree edges. This can be regarded as both an advantage and a disadvantage. Its advantage is that there is no ambiguity in the output regarding which sensitive values occur together. Hence, data reconstruction and interpretation is easier: Given the QI-trees and SA-trees in a group, the potential hierarchical data records can be reconstructed using a permutation operation $\bowtie$ and we can run any data mining task on these outputs as if we ran them on the original (private, unpublished) records. Another advantage is that joint distributions and correlations between sensitive values are preserved in the published data.

However, explicit links between sensitive values can also become a disadvantage. The general assumption in the privacy preserving data publishing literature is that an adversary has his victims' QI information, but no knowledge of SAs. The previous works on hierarchical data publishing also employ this assumption [6, 52]. However, as [53] points out, the unique nature of hierarchical data may invalidate this assumption. Let us assume a stronger adversary, who not only has QI information, but also one sensitive value. For example, for the publication in Figure 3.3, let the adversary know that Alice's diagnosis in 2007 was Cold. Such knowledge is realistic, e.g., the adversary was a co-worker of Alice and Alice might have given this information thinking Cold is common and harmless. Then, since all other SAs are linked to Cold, the adversary immediately learns that Alice was also diagnosed with AIDS in 2000. This shows that in hierarchical data, direct links between SAs can help an adversary infer discriminatory sensitive values given that he knows a non-discriminatory value. A second disadvantage is that explicit links makes enforcing (p,m)-privacy harder which we will cover in coming sections.

We designed v-v anatomy to overcome this problem. v-v anatomy prevents this attack at the expense of no longer preserving correlations between sensitive values. The publishing strategy of v-v anatomy is roughly the same for QIs, i.e., QIs are still published as tree structures. However, SAs are published in independent groups rather than trees.

We illustrate v-v anatomy using Figure 3.4. In contrast to QI-trees in t-t anatomy, we use *augmented QI-trees (AQI-trees)* in v-v anatomy. AQI-trees extend QI-trees by adding an explicit sensitive value identifier associated with each vertex, e.g., $G0.1$, $G1.1$, $G1.2$. For each AQI-tree published in one group, we ensure that the vertices which appear in the same index of the breadth-order traversal have the same identifier. (Formally, for the traversal of AQI-tree T_i in group G, i.e., $\mathcal{R}(T_i) = \{v_1^i, v_2^i, .., v_m^i\}$, for all j, the set of vertices $\mathcal{I}_j = \bigcup_{i \in [1,|G|]} v_j^i$ contain the same sensitive value identifier.) Furthermore, each identifier is unique, i.e., it is never repeated within the same tree. Each unique identifier corresponds to one *sensitive attribute group (SA-group)*. An SA-group is a collection (multiset) of sensitive values. The key difference between v-v anatomy and t-t anatomy is that v-v anatomy employs SA-groups instead of SA-trees.

For a vertex v in an augmented QI-tree, we use $v.SAG$ to denote the corresponding SA group and $v.I$ to denote the vertices that has the same sensitive value identifier.

We now discuss how reconstruction works in v-v anatomy and how the aforementioned privacy attack is prevented. Consider the v-v anatomization given in Figure 3.4. First, let Bob be the 47 years old victim whose QIs are known by the adversary in Figure 3.4. Then, the adversary can precisely locate Bob's QI-tree. For each of Bob's sensitive values, the adversary observes an SA-group identifier, e.g., $G1.2$ containing $\{$Alcohol P.,Flu$\}$. However, none of Bob's actual sensitive values can be inferred with probability larger than $^1/_\ell$, as long as each SA-group satisfies ℓ-diversity. Second, let Alice be the 46 years old victim whose QIs are known by the adversary, and in addition, one more of her SAs are also known, say the diagnosis *Cold* in 2007. Notice that the unknown SAs cannot be inferred, e.g., even if *Cold* is known by the adversary, a second diagnosis, $\{$Alcohol P.,Flu$\}$ cannot be discovered with confidence larger than $^1/_\ell$ because it occurs in an independent SA-group $G1.2$. (Of course, in this stronger adversarial setting, ℓ-diversity and (p, m)-privacy lose their meaning for the sensitive value that is already known by the adversary.)

The permutation operation (i.e., $\bowtie$) is more involved: Let T be an AQI-tree with $|V|$ vertices. For each vertex, given that the publication satisfies ℓ-diversity, there exists an SA-group of cardinality at least ℓ that stores potential sensitive values for that vertex. Therefore the $\bowtie$ operation has at least $(\ell!)^{|V|}$ outputs, as opposed to $\ell!$ in t-t anatomy. The probability that an individual's *all* sensitive values are correctly reconstructed is $\ell^{-|V|}$, as opposed to ℓ^{-1} in t-t anatomy. Typical values for $\ell \in [2, 5]$, hence it is possible to enumerate all outputs of $\bowtie$ in t-t anatomy. On the other hand, we often have trees with $|V| \geq 20$, which causes the output of $\bowtie$ to be incomputable in the case of v-v anatomy. Hence, when reconstructing v-v anatomous data, we need to resort to answering queries probabilistically. This limitation in practicality is the price we have to pay to overcome the aforementioned attack.

Enforcement of ℓ-diversity has 3 requirements: (1) Structural isomorphism of AQI-trees prohibits linkage attacks based on structural differences. (2) The existence and uniqueness of SA-groups within each tree guarantees that an individual's information is not under-represented or over-represented in SA-groups. (3) Since SA-groups are multisets by construction, application of the standard ℓ-diversity definition (in Def. 6) is sufficient to ensure ℓ-diversity of the published information. These 3 requirements are formalized in the definition below.

Definition 22. ***(ℓ-diversity for v-v anatomy)*** *Let G be a v-v anatomous publication of a group of AQI-trees and SA-groups. G satisfies ℓ-diversity, if:*

1. *The number of AQI-trees in the group is $\geq \ell$, and all AQI-trees are structurally isomorphic.*
2. *Each vertex in an AQI-tree is associated with an SA-group, and no SA-group is repeated twice in the same AQI-tree.*
3. *Each SA-group satisfies the multiset ℓ-diversity definition given in Def. 6.*

Enforcement of (p,m)-privacy. Let (s_i, m, p) be the discriminatory rule under consideration. First, we observe that after ℓ-diversity is satisfied, the frequency of occurrence of s_i in any SA-group must be less than or equal to $1/\ell$. From a worst-case perspective, assume that the frequency is equal to $1/\ell$. Let $\eta(G, s_i)$ denote the number of SA-groups in G that contain one or more occurrences of s_i. The probability that an adversary infers exactly m occurrences of s_i in any reconstructed hierarchical data record in G follows a Binomial distribution:

$$\begin{aligned} Pr[X = m] &= Pr[m; \eta(G, s_i), \frac{1}{\ell}] \\ &= \binom{\eta(G, s_i)}{m} \cdot (\frac{1}{\ell})^m \cdot (1 - \frac{1}{\ell})^{\eta(G,s_i)-m} \end{aligned}$$

where X is a random variable capturing the number of occurrences of s_i in any reconstructed record in G. From Def. 19, we need to ensure: $Pr[X \geq m] \leq p$. It follows that:

$$1 - Pr[X < m] \leq p$$

$$Pr[X < m] \geq 1 - p$$

Following the cumulative distribution function of the Binomial distribution, together with the notation above, the LHS term can be written as follows:

$$Pr[X < m] = \sum_{j=0}^{\lfloor m-1 \rfloor} \binom{\eta(G, s_i)}{j} \cdot (\frac{1}{\ell})^j \cdot (1 - \frac{1}{\ell})^{\eta(G,s_i)-j}$$

Definition 23. ***((p,m)-privacy for v-v anatomy)*** *Let G be a v-v anatomous publication of a group of AQI-trees and SA-groups, and $\mathcal{S}$ denote a list of discriminatory rules. G satisfies (p, m)-privacy, if for each rule $(s_i, m, p) \in \mathcal{S}$, the following inequality holds:*

$$\sum_{j=0}^{\lfloor m-1 \rfloor} \binom{\eta(G, s_i)}{j} \cdot (\frac{1}{\ell})^j \cdot (1 - \frac{1}{\ell})^{\eta(G,s_i)-j} \geq 1 - p$$

where j is an integer.

The v-v anatomization given in Figure 3.4 satisfies (p, m)-privacy with rules (AIDS,1,½) and (Alcohol P.,2,¼). For the latter, we have two SA-groups containing Alcohol P. (e.g., $\eta(G, \text{Alcohol P.}) = 2$). We have $\binom{2}{0} \cdot (1/2)^2 + \binom{2}{1} \cdot 1/2 \cdot 1/2 = 3/4 \geq 1 - 1/4$, thus the rule holds. Note that even though the groups and the corresponding node matchings are the same in v-v and $t - t$ anatomizations in Figures 3.4 and 3.3, v-v anatomization satisfies a stronger (p, m)-privacy standard. While this is so, $t - t$ anatomization has the ability to better preserve the explicit links between sensitive values.

Finally, we say that a v-v anatomous publication of the whole data is privacy preserving, if for all groups containing AQI-trees and SA-groups, ℓ-diversity and (p, m)-privacy are satisfied.

3.4 Anatomization Algorithms for Hierarchical Data

In this section, we propose an algorithm for hierarchical data that creates t-t or v-v anatomizations satisfying ℓ-diversity and (p, m)-privacy. The pseudocode for the algorithm *Anatomize* is given in Algorithm 6.

Algorithm 6 Anatomize

```
Require: A collection of trees $F$, privacy parameter $\ell$ and a set of discriminatory rules $R$.
Ensure: The returned dataset is an anatomization of $F$ and respects $\ell$-diversity and $(m, p)$-privacy for the given rules.
 1: let $F^A$ be an empty set of augmented trees
 2: let $LP$ be an empty set of trees
 3: let $FP = \{F_1, \ldots, F_{np}\}$ where each $F_i$ is an empty set of trees.
 4: $FP$ = partition$(F, np)$
 5: $FP = FP + LP$
 6: for all $F_i \in FP$ do
 7:     for all $T \in F_i$ do
 8:         if $|F_i| < \ell$ then
 9:             $LP = LP \cup F_i$
10:             break
11:         $F_i = F_i - T$
12:         let $T^A$ be an augmented tree structurally isomorphic to $T$.
13:         for all vertex $v^A \in T^A$ do
14:             let $v$ be the vertex with the same breadth-first order in $T$
15:             $v^A.SAG = \{v.sa\}$
16:             $v^A.I = \{v\}$
17:         for $j \leftarrow 2, \ell$ do
18:             $T' \leftarrow$ argmin$_{T \in F_i}$COST(Merge$(T^A, T, R)$)
19:             $T^A \leftarrow$ Merge$(T^A, T')$
20:             $F_i = F_i - T'$
21:         $F^A = F^A + T^A$
22: generate and return $t$-$t$ or $v$-$v$ anatomization respecting SA groups and matched vertices (e.g., $I$) in $F_A$.
```

As the first step on line 4, the algorithm splits the data into equal size partitions where the number of partitions np is an input parameter. To partition, we order the collection of trees according to number of vertices in the trees and then place them into new sets in such a way that each partition contains trees with similar number of vertices. The algorithm almost always groups trees belonging to the same partition. Partitioning makes the algorithm more efficient by only searching for groups within a much smaller population at a small cost to utility. We claim that the cost to utility is minimal. An optimal algorithm without partitioning would avoid grouping trees with many vertices with those with few vertices since such grouping would cause many vertices in the former to be suppressed. Thus, for most trees best possible group candidates reside in the same partition.

On the next step, the algorithm iterates over each partition F_i on line 6 and tries to group trees in F_i. Each group is represented as an augmented tree so that information about linked vertices and associated SA groups can be maintained. ℓ-diversity is enforced in an incremental fashion. For instance, if ℓ is 3, algorithm will first pick a tree T from F and find the closest tree that can be *merged* with minimum cost so that the resultant group satisfies 2-diversity. (we discuss algorithm merge shortly) Then, another tree is merged with the group in such a way that the final group satisfies 3-diversity. Finally the group is then put in the result set F^A on line 21. Note that any group created by the algorithm contains exactly ℓ data trees. At the end of the loop, those $< \ell$ many trees that cannot be diversified are moved to a temporary partition LP. Trees in LP are revisited once all partitions are processed.

Algorithm 7 Merge

```
Require: augmented tree $T^A$, tree $T$ and a set of discriminatory rules $R$
Ensure: the returned augmented tree is a merge of the two input trees and does not violate
    the discriminatory rules.
1: $\hat{T}^A$ = clone($T^A$)
2: MergeVertices({root($\hat{T}^A$)},{root($T$)},$\hat{T}^A$,$R$)
```

Algorithm *Merge* (given in Alg. 7) gets an associated j-diverse augmented tree for the existing group and a tree T, and merges them into a new $(j + 1)$-diverse augmented tree respecting discriminatory rules. The merge of the trees is performed recursively in a top-down fashion. At each level of the trees, the algorithm *MergeVertices* (given in Alg. 8) maps each vertex v^A in the augmented tree with exactly and uniquely one vertex v in the data tree only if the following conditions are met. First, the sensitive value $v.sa$ associated with v should not appear in the SA group $v^A.SAG$. Recall that groups contain exactly ℓ trees, thus any repeating sensitive attribute would violate ℓ-diversity. Second, adding $v.sa$ in $v^A.SAG$ should not violate any of the discriminatory rules. Recall that given the augmented tree T^A, this check can be performed via Definition 23 or 21 depending on the desired type of the anatomization. If both conditions are met, the vertices are linked and the algorithm is called on the children of vertices recursively on line 8.

Algorithm 8 MergeVertices

```
Require: set of augmented vertices $V^A$, set of vertices $V$, associated augmented tree $T^A$,
    and a set of discriminatory rules $R$
Ensure: $V^A$ and $V$ are merged into $V^A$ in such a way that SA groups in $T^A$ respects
    diversity and $(p, m)$-privacy
1: set all $v^A \in V^A$ and $v \in V$ as unmatched.
2: for all $v^A \in V^A$ do
3:     for all unmatched $v \in V$ do
4:         if $v.sa \notin v^A.SAG$ then
5:             $v^A.SAG = v^A.SAG + v.sa$
6:             $v^A.I = v^A.I + v$
7:             if $T^A$ satisfies $(m, p)$-privacy rules $R$(see Def. 23 and 21) then
8:                 MergeVertices(children of $v^A$, children of $v^A$, $T^A$, $R$)
9:                 set $v^A$ and $v$ as matched.
10:                break
11:            else
12:                $v^A.SAG = v^A.SAG - v.sa$
13:                $v^A.I = v^A.I - v$
14:    if $v^A$ is unmatched then
15:        suppress the subtree rooted by $v^A$ in $T^A$
```

3.5 Experiments

We now present experimental results regarding anatomization algorithms we proposed for hierarchical databases. We implemented our algorithms in Java 8. Experiments were conducted on a machine with Intel Xeon E3 @ 3.60 GHz CPU and 32 GB RAM. In order to evaluate our work on (p,m)-privacy, we did extensive experiments which show the effects of varying the privacy parameters p, m and ℓ. These experiments include measurement of suppression and query accuracy. At the end of this section, we compare our technique with our previous work described in Chapter 2. We use the same dataset *syntheticS* detailed in Section 2.4 in all of these experiments.

We use two quality metrics to evaluate the utility of released hierarchical data. First metric we use is suppression accuracy. Suppression accuracy can be calculated as

$$\frac{\sum_{T_i \in F} \frac{|\{v \mid v \in T_i^A\}|}{|\{v \mid v \in T_i\}|}}{|F|} \cdot 100$$

where T_i^A is the anatomized tree linked to the same individual as T_i and query accuracy can be calculated with the same formula given in 2.4

Suppression accuracy

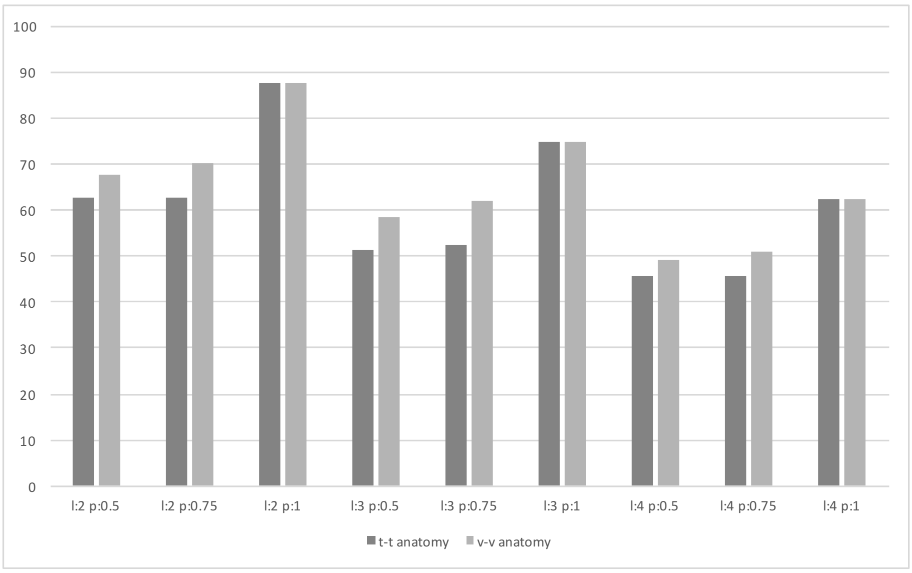


Figure 3.5: Suppression accuracy at m = 3 ℓ-p varying

In Figure 3.5, we fix m at 3 and demonstrate the change in suppression accuracy with respect to various ℓ and p. It's clearly seen that p and accuracy are directly proportional, e.g., as p increases, so does the accuracy. Utility and privacy requirements are inversely proportional. As ℓ increases, so does the number of trees per group. Each additional tree requires more suppression, thus the utility drops. Note that when p is 1, t-t anatomy yields the same suppression accuracy as v-v anatomy. This is an expected behaviour since for $p = 1$ the discriminatory rules are not enforced.

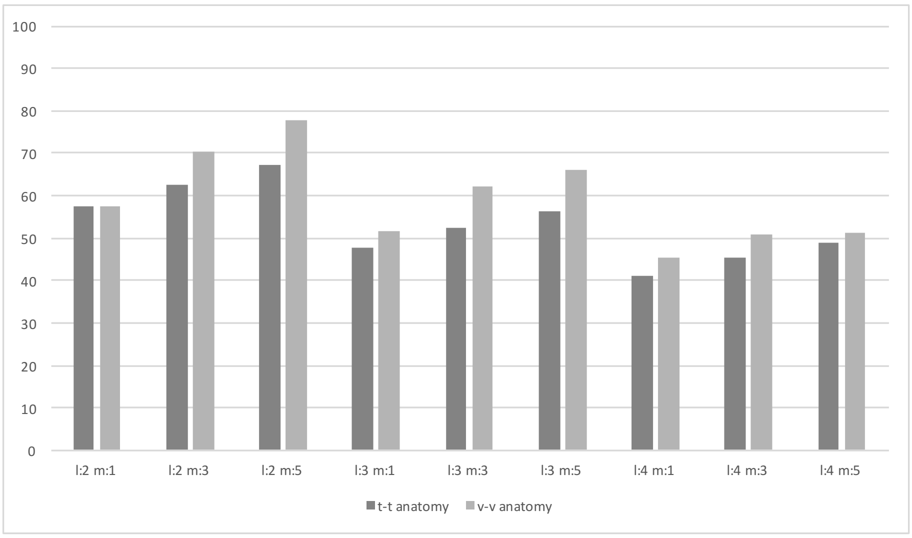


Figure 3.6: Suppression accuracy at p = 0.75 ℓ-m varying

In Figure 3.6 we fix p at 0.75 and we investigate the change in suppression accuracy versus change in ℓ and m. As m increases, the standard tolerates more occurrences of the same sensitive values appearing in an anatomized tree, this in return decreases the number of suppressed vertices per anatomized tree.

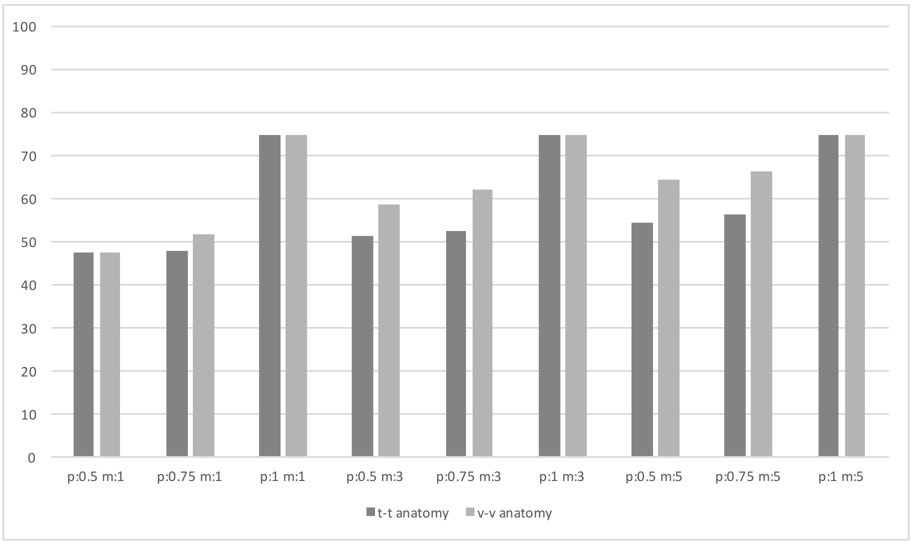


Figure 3.7: Suppression accuracy at ℓ = 3 p-m varying

In Figure 3.7 we fix ℓ at 3. We observe change in suppression accuracy due to change in m and p. Similar conclusions can be derived here, increase in m and p results in better accuracy.

In all three experiments, we achieve $> 60\%$ accuracy for $\ell = 2$. This drops to $> 50\%$ and $> 40\%$ for $\ell = 3$ and $\ell = 4$. We also observe the overhead of achieving (p,m)-privacy. For example, the accuracy increases to $> 80\%$, $> 70\%$, and $> 60\%$ when $p = 1$. Another important point is that in all our suppression tests with $p \neq 1$ v-v anatomy almost always performs better than t-t anatomy. Recall that a grouping achieving (p,m)-privacy for t-t anatomy also satisfies (p,m)-privacy for v-v anatomy. However, the reverse is not the same. Thus, there likely exist groupings with lower number of suppressions satisfying (p,m)-privacy for v-v anatomy but not t-t anatomy.

Query accuracy

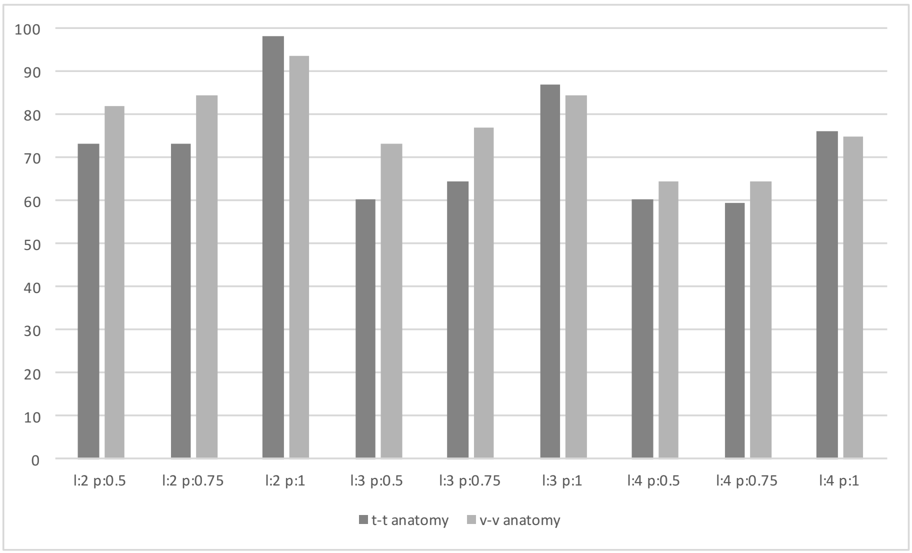


Figure 3.8: Query accuracy at m = 3 ℓ-p varying

In Figure 3.8, we examine query accuracy with varying ℓ and p when m is fixed at 3. We observe a slightly different behaviour compared to the results in Figure 3.5, Figure 3.6 and Figure 3.7. As in suppression accuracy experiments, v-v anatomy out-performs t-t anatomy in all settings $p \neq 1$. When $p = 1$, however, we see t-t anatomy results in anatomizations with better query accuracy. This is an expected behaviour. Note that a grouping achieves ℓ-diversity for t-t anatomy if and only if it also does for v-v anatomy.

As discussed in Section 3.3.1 and 3.3.2, for the same grouping, t-t anatomy preserves explicit links between sensitive values, thus provides better query accuracy. Note that the information content within explicit links are not captured by the suppression accuracy metric.

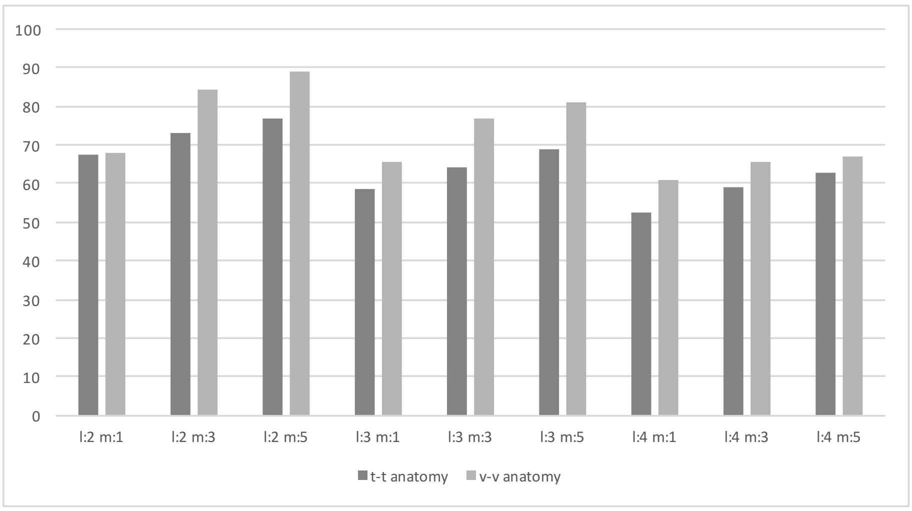


Figure 3.9: Query accuracy at p = 0.75 ℓ-m varying

In Figures 3.9 and 3.10, we observe a similar behaviour as in Figures 3.6 and 3.7.

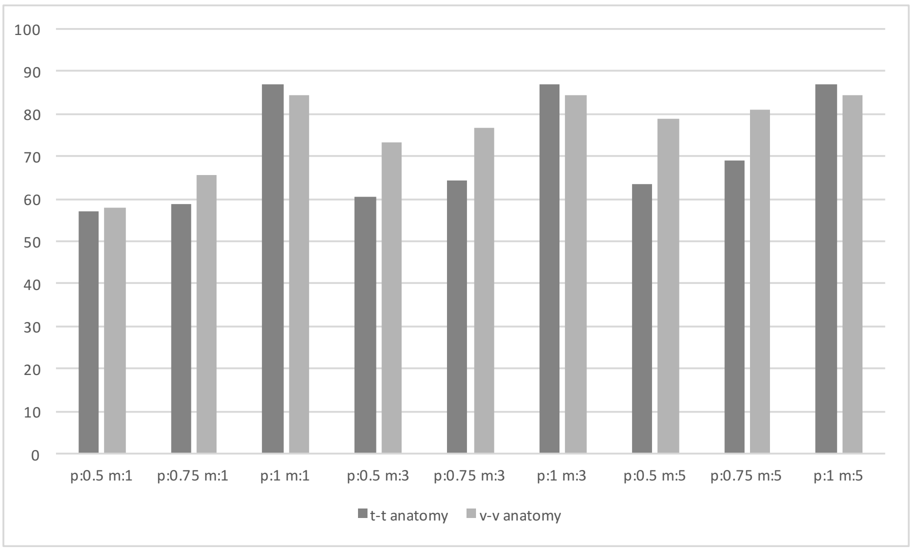


Figure 3.10: Query accuracy at ℓ = 3 p-m varying

We now investigate how query type affects utility by clustering queries into partitions (e.g., query families). We have created 10 different query families which are based on the count of SAs and QIs in different depths of the trees and their combinations. It includes queries such as:

- The count of students having A- or B+ from a Math or CS course.
- The count of students with age 26 and has taken CS 201.
- The count of students that has taken Math 101 and got grade B- in Math 101 and also has bought Math 101 textbook.

In Figure 3.11, Figure 3.12 and Figure 3.13, we show accuracy versus queries (denoted with their family id numbers). We have different settings of ℓ, p and m in each figure so that it enable us to compare t-t anatomy and v-v anatomy. In all these figures, we can clearly see that when the depth of the query increases, the query accuracy reduces which is an expected behaviour. Because as we go down during top down merging of trees when upper level parent node is suppressed, children nodes that are connected to the parent node are also suppressed.

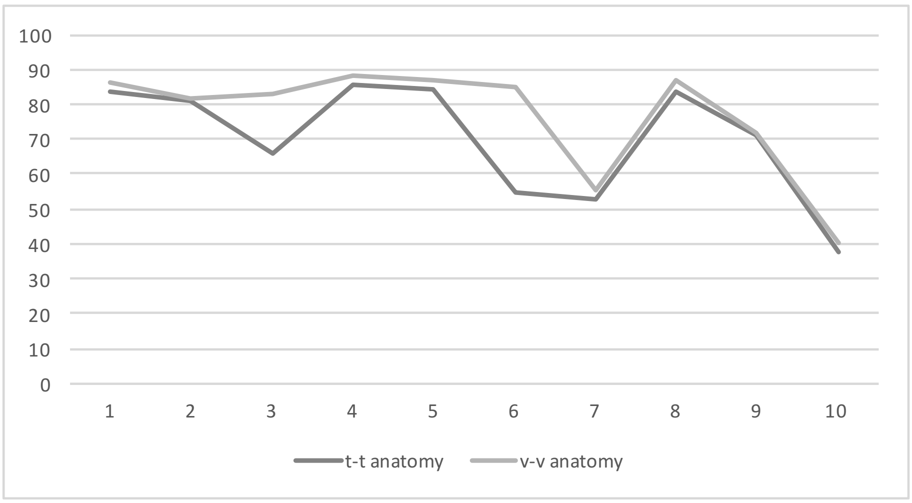


Figure 3.11: Query family accuracy at ℓ = 2, m = 2, p = 0.5

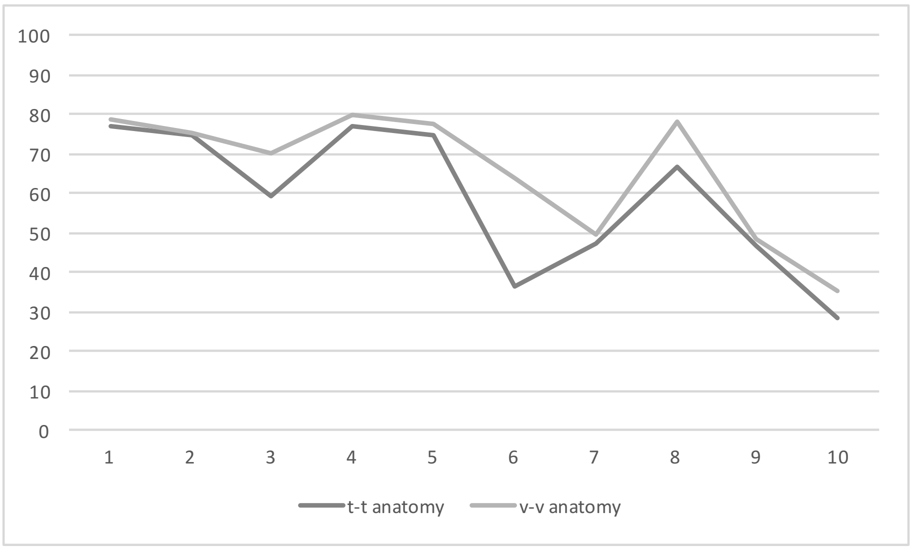


Figure 3.12: Query family accuracy at $\ell = 3$, $m = 2$, $p = 0.33$

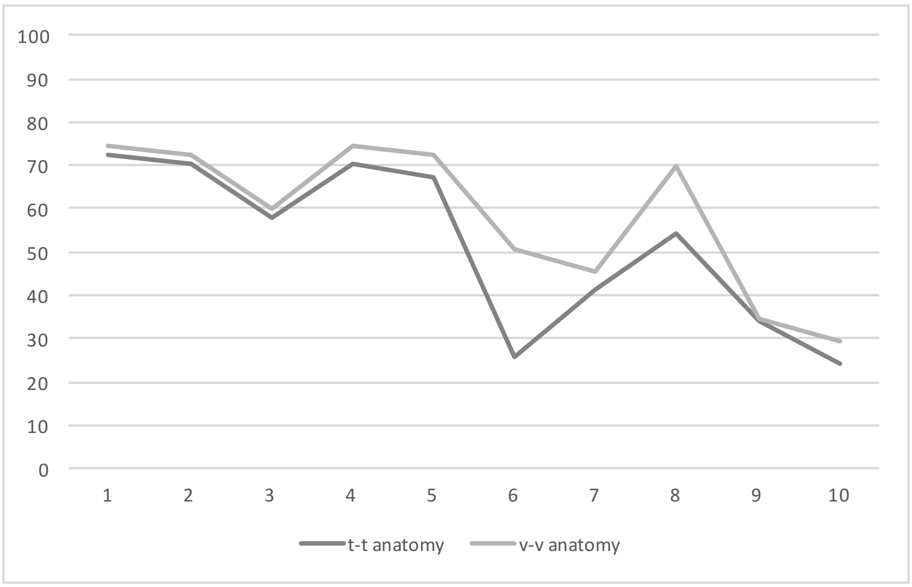


Figure 3.13: Query family accuracy at $\ell = 4$, $m = 2$, $p = 0.25$

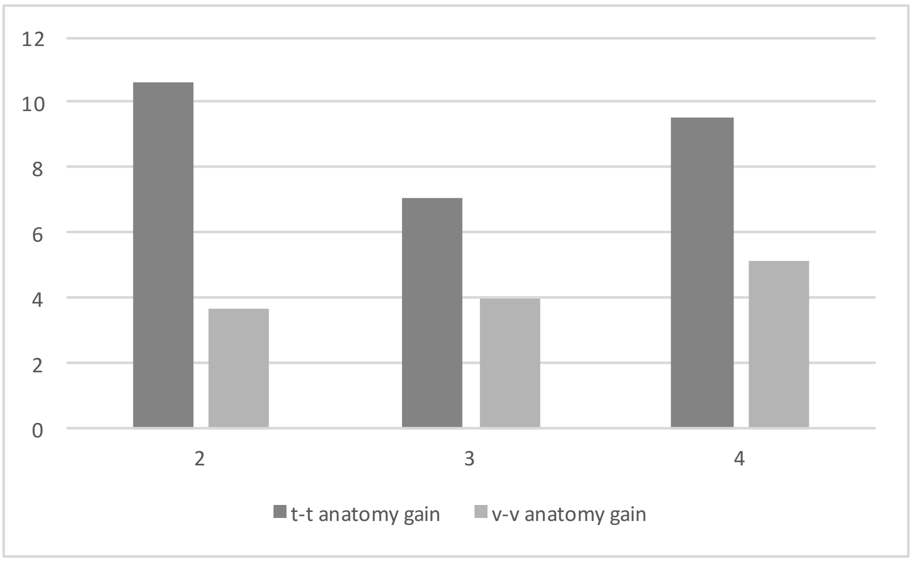


Figure 3.14: Accuracy gain in percentage vs ℓ-diversity

Comparison to Previous Work

In order to evaluate our anatomy technique with generalization technique proposed in Chapter 2, we run the same query accuracy tests on ℓ-diverse generalization, ℓ-diverse t-t and v-v anatomization with no discriminatory rules. In Figure 3.14, x-axis shows ℓ values and y-axis shows accuracy gain (in percentage) against ℓ-diversity. We observe that t-t-anatomization offers 6-10% more query accuracy compared to generalization. This drops to 3-5% better accuracy for v-v-anatomy.

Effect of partitioning

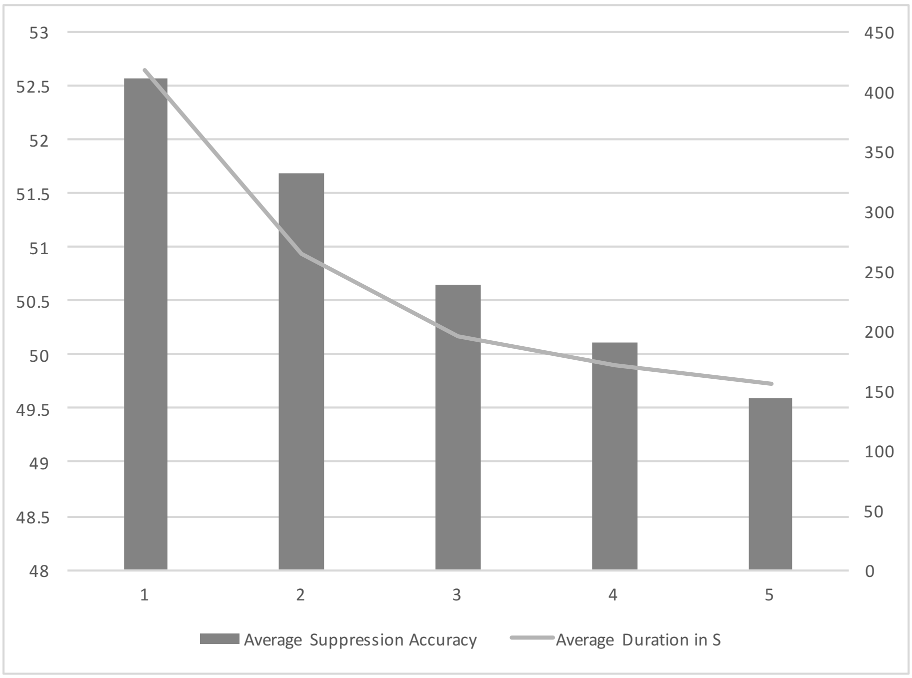


Figure 3.15: t-t anatomy running time over number of partitions

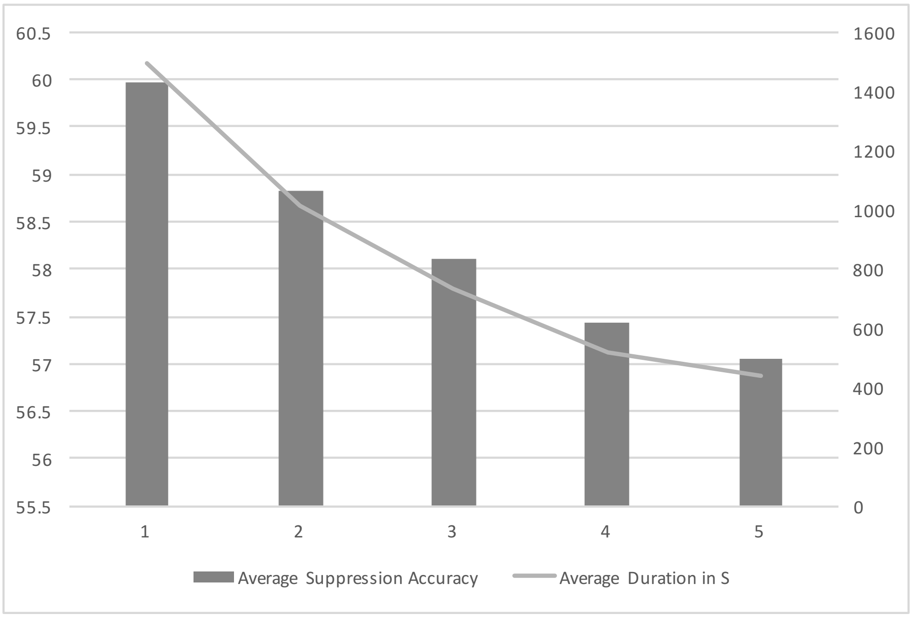


Figure 3.16: v-v anatomy running time over number of partitions

In Figure 3.15 and Figure 3.16, we investigate the effect of partitioning over utility. In both figures, horizontal axis denotes the number of clusters and vertical axis denotes the average utility and right vertical axis denotes the average duration in seconds. For both t-t anatomy and v-v anatomy, one can improve the running time of the anatomization process by increasing the number of partitions. Normally the size of the clusters decreases as the number of partitions increases. Hence the speedup is based on comparing less number of trees in the same cluster. For instance, increasing the number of partitions up to 5 results in reduction of running time to one third of using a single partition. This incurs only a $\%3$ loss of utility which a good trade-off between utility and efficiency. One may prefer this method for a rapid preview or when the efficiency is a concern as in streaming databases.

Chapter 4

Conclusions

In this thesis we addressed the problem of privacy in hierarchical data publishing. We discussed how popular privacy notions such as k-anonymity, ℓ-diversity, as well as other new standards can be applied to hierarchical data with generalization and anatomy techniques. In Chapter 2 we designed an algorithm that produces ℓ-diverse generalizations of collections of hierarchical data records. Even though we use the LM metric in Chapter 2, our approach is suitable for other monotonic cost metrics. For example, one can use a metric that penalizes certain levels in the hierarchical schema more than others (e.g., to apply more emphasis on courses than evaluations). Other domain-specific heuristics can also be employed. To fight sparsity of high-dimensional data and provide flexibility, our solution uses local recoding. We also address negative knowledge as well as positive knowledge: For every piece of information an adversary has (e.g., student has taken course X and/or has not taken course Y), there are at least ℓ records in the anonymized output that fit this description. Therefore, the adversary's confidence regarding a particular sensitive value of his victim is always bounded by $1/\ell$.

There are also certain limitations of our approach in Chapter 2. For example, if all records in an equivalence class contain the letter grade A for different courses, an adversary may learn with probability $> 1/\ell$ that his victim has received an A from *some* course, even though he cannot be certain *which* course it was. In certain cases such disclosures might be unacceptable, e.g., adversary learns that his victim has AIDS from an anonymized medical data set. To aid in this problem, in Chapter 3 we presented new standard called (p,m)-privacy which bounds the probability of being linked to more than m occurrences of any sensitive values by p, through defining discriminatory rules for different sensitive values. Instead of utility-expensive generalizations, we enforced (p,m)-privacy with anatomization which de-associates QIs and SAs to increase utility in shared databases. We have proposed two different types of de-association (publishing techniques), t-t anatomy and v-v anatomy, each of which is better suited in a different scenario. Both proposed techniques ensure ℓ-diversity and (p,m)-privacy.

4.1 Future Work

Anonymization and anatomization techniques are extensively studied in the literature. Current techniques are mostly developed for tabular data. We believe that hierarchical data is also as important as tabular data. With growing number of devices and combining various sources of information available, near future will witness more hierarchical data sets than ever, which will bring the challenge of privacy preserving data transformation of hierarchical data for data owners.

This thesis brings a novel solution to this problem. We can discuss several directions as future work. Since our anonymization and anatomization strategy does not allow noise in the output. One could try to see whether data utility can be improved by adding noise and counterfeits like differential privacy [54], which relies on noise addition. This question may yield alternative techniques for reaching higher data utility. Also one may improve the current partitioning technique by altering the introduced partitioning scheme (i.e. introducing new partitioning strategies). Moreover one may also come up with new publishing methods other than t-t and v-v anatomy which should be investigated in future.

In addition in Chapter 3 our proposed techniques results in exactly ℓ number of trees in a group. One may try to add more than ℓ trees in a group, as it's perfectly okay to have more than ℓ trees in group as long as (p,m)-privacy and/or ℓ-diversity are satisfied.

Furthermore there are numerous tools and engines that process hierarchical data. In particular, XML streams and query engines are widely used in today's world. An interesting area of research is how our definitions of privacy can be applied in these contexts (e.g., XML data streams) [55].

Finally, hierarchical data anonymization and anatomization is open to develop further techniques to work for different privacy definitions and expectations.

Bibliography

[1] US Congress, Sponsor: Sen. Brian Schatz , “S.2852, open government data act,” 2016.

[2] US Department of Health and Human Services, National Institutes of Health , “Nih data sharing policy and implementation guidance,” 2016.

[3] Council of European Union, “Directive 95/46/ec protection of individuals with regard to the processing of personal data on the free movement of such data, oj l 281, 23/11/95, pp 31-50,” 1995.

[4] US Department of Health and Human Services, Health Information Privacy, “The hipaa privacy rule,” 2002.

[5] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[6] M. E. Nergiz, C. Clifton, and A. E. Nergiz, “Multirelational k-anonymity,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 8, pp. 1104–1117, 2009.

[7] X. Yang and C. Li, “Secure xml publishing without information leakage in the presence of data inference,” in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004)*, pp. 96–107, VLDB Endowment, 2004.

[8] A. H. Landberg, K. Nguyen, E. Pardede, and J. W. Rahayu, "δ-dependency for privacy-preserving xml data publishing," *Journal of Biomedical Informatics*, vol. 50, pp. 77–94, 2014.

[9] O. Gkountouna and M. Terrovitis, "Anonymizing collections of tree-structured data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, 2015.

[10] S. D. Warren and L. D. Brandeis, "The right to privacy," *Harvard Law Review*, vol. 4, no. 5, pp. 193–220, 1890.

[11] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[12] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[13] T. M. Truta and B. Vinay, "Privacy protection: p-sensitive k-anonymity property," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006) Workshops*, p. 94, IEEE, 2006.

[14] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," tech. rep., SRI International, 1998.

[15] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, pp. 25–25, IEEE, 2006.

[16] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS 2004)*, pp. 223–228, ACM, 2004.

[17] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, pp. 217–228, IEEE, 2005.

[18] V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pp. 279–288, ACM, 2002.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *Proceedings of the 24th ACM International Conference on Management of Data (SIGMOD 2005)*, pp. 49–60, ACM, 2005.

[20] M. E. Nergiz, A. Tamersoy, and Y. Saygin, "Instant anonymization," *ACM Transactions on Database Systems*, vol. 36, no. 1, p. 2, 2011.

[21] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.

[22] B. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[23] X. Xiao, K. Yi, and Y. Tao, "The hardness and approximation algorithms for l-diversity," in *Proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010)*, pp. 135–146, ACM, 2010.

[24] J. Liu and K. Wang, "On optimal anonymization for l+-diversity," in *Proceedings of the 26th International Conference on Data Engineering (ICDE 2010)*, pp. 213–224, IEEE, 2010.

[25] J. Cheng, A. W.-C. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *Proceedings of the 29th ACM International Conference on Management of Data (SIGMOD 2010)*, pp. 459–470, ACM, 2010.

[26] K. Liu and E. Terzi, “Towards identity anonymization on graphs,” in *Proceedings of the 27th ACM International Conference on Management of Data (SIGMOD 2008)*, pp. 93–106, ACM, 2008.

[27] E. Zheleva and L. Getoor, “Preserving the privacy of sensitive relationships in graph data,” in *Proceedings of the 2nd International Workshop on Privacy, Security and Trust in KDD (PinKDD 2008)*, pp. 153–171, Springer, 2008.

[28] B. Zhou, J. Pei, and W. Luk, “A brief survey on anonymization techniques for privacy preserving publishing of social network data,” *ACM SIGKDD Explorations Newsletter*, vol. 10, no. 2, pp. 12–22, 2008.

[29] G. Ghinita, P. Kalnis, and Y. Tao, “Anonymous publication of sensitive transactional data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 2, pp. 161–174, 2011.

[30] Y. He and J. F. Naughton, “Anonymization of set-valued data via top-down, local generalization,” *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 934–945, 2009.

[31] M. Terrovitis, N. Mamoulis, and P. Kalnis, “Privacy-preserving anonymization of set-valued data,” *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 115–125, 2008.

[32] M. Terrovitis, N. Mamoulis, and P. Kalnis, “Local and global recoding methods for anonymizing set-valued data,” *The VLDB Journal*, vol. 20, no. 1, pp. 83–106, 2011.

[33] A. E. Cicek, M. E. Nergiz, and Y. Saygin, “Ensuring location diversity in privacy-preserving spatio-temporal data publishing,” *The VLDB Journal*, vol. 23, no. 4, pp. 609–625, 2014.

[34] M. E. Nergiz, M. Atzori, and Y. Saygin, “Towards trajectory anonymization: a generalization-based approach,” in *Proceedings of the 2008 ACM SIGSPATIAL In-*

ternational Workshop on Security and Privacy in GIS and LBS, pp. 52–61, ACM, 2008.

[35] M. Terrovitis and N. Mamoulis, "Privacy preservation in the publication of trajectories," in *Proceedings of the 9th International Conference on Mobile Data Management (MDM 2008)*, pp. 65–72, IEEE, 2008.

[36] A. H. Landberg, J. W. Rahayu, and E. Pardede, "n-dependency: dependency diversity in anatomised microdata tables," *Logic Journal of IGPL*, vol. 19, no. 5, pp. 679–702, 2011.

[37] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007)*, pp. 106–115, IEEE, 2007.

[38] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, "Specifying and enforcing access control policies for xml document sources," *World Wide Web*, vol. 3, no. 3, pp. 139–151, 2000.

[39] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "A fine-grained access control system for xml documents," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 2, pp. 169–202, 2002.

[40] I. Fundulaki and M. Marx, "Specifying access control policies for xml documents with xpath," in *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, pp. 61–69, ACM, 2004.

[41] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006)*, pp. 139–150, VLDB Endowment, 2006.

[42] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques.* Elsevier, 2011.

[43] L. Sweeney, "Uniqueness of simple demographics in the us population," tech. rep., Carnegie Mellon University, 2000.

[44] E. Bertino, D. Lin, and W. Jiang, "A survey of quantification of privacy preserving data mining algorithms," in *Privacy-preserving data mining*, pp. 183–205, Springer, 2008.

[45] X. Xiao and Y. Tao, "Personalized privacy preservation," in *Proceedings of the 25th ACM International Conference on Management of Data (SIGMOD 2006)*, pp. 229–240, ACM, 2006.

[46] A. Tamersoy, G. Loukides, M. E. Nergiz, Y. Saygin, and B. Malin, "Anonymization of longitudinal electronic medical records," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 3, pp. 413–423, 2012.

[47] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[48] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[49] D. W. Pentico, "Assignment problems: A golden anniversary survey," *European Journal of Operational Research*, vol. 176, no. 2, pp. 774–793, 2007.

[50] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[51] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[52] I. Ozalp, M. E. Gursoy, M. E. Nergiz, and Y. Saygin, "Privacy-preserving publishing of hierarchical data," *ACM Trans. Priv. Secur.*, vol. 19, pp. 7:1–7:29, Sept. 2016.

[53] M. E. Gursoy, A. Inan, M. E. Nergiz, and Y. Saygin, "Privacy-preserving learning analytics: Challenges and techniques," *IEEE Transactions on Learning Technologies*, vol. 10, no. 1, pp. 68–81, 2017.

[54] C. Dwork, “Differential privacy: A survey of results,” in *Theory and applications of models of computation*, pp. 1–19, Springer, 2008.

[55] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia, “Continuous privacy preserving publishing of data streams,” in *Proceedings of the 12th International Conference on Extending Database Technology (EDBT 2009)*, pp. 648–659, ACM, 2009.