SECURING THE INFORMATION DISCLOSURE PROCESS

by

Lei Zhang

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

[illegible]	Dr. Sushil Jajodia, Dissertation Co-director
[illegible]	Dr. Alexander Brodsky, Dissertation Co-director
[illegible]	Dr. Duminda Wijesekera, Committee Member
[illegible]	Dr. Songqing Chen, Committee Member
[illegible]	Dr. Daniel Menascé, Senior Associate Dean
[illegible]	Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: 06/10/2010

Summer 2010
George Mason University
Fairfax, VA

Securing the Information Disclosure Process

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Lei Zhang
Master of Engineering
Tsinghua University, 2004
Bachelor of Engineering
Tsinghua University, 2001

Co-director: Sushil Jajodia, Professor
Center for Secure Information Systems

Co-director: Alexander Brodsky, Associate Professor
Department of Computer Science

Summer 2010
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my wife Ying Wu and my parents.

Acknowledgments

I would like to express my sincere gratitude to my advisors, Professor Sushil Jajodia and Professor Alexander Brodsky, for their guidance, encouragement, and support in every stage of my graduate study. I am deeply grateful to Professor Sushil Jajodia, who gave me the opportunity to work in the prestigious Center for Secure Information Systems (CSIS) and guided me through the journey of becoming a scholar. His knowledge, vision, and discipline provided me lifetime benefits. I am also very grateful to my co-advisor, Professor Alexander Brodsky. He is always willing to help, both on academic development and on personal issues.

I thank my committee members, Professor Duminda Wijesekera and Professor Songqing Chen, for their wonderful class materials and their support and time. I thank Professor Angelos Stavrou for his great help on my research and his precious friendship.

I thank my colleagues who have graduated from George Mason University: Dr. Lingyu Wang, Dr. Xinwen Zhang, Dr. Shiping Chen, and Dr. Sankardas Roy.

I also thank the members of our research center for their suggestions: Dr. Pierangela Samarati, Dr. Xiaoyang Wang, and Dr. Bo Zhu. I also want to thank Jiang Wang for his helpful discussions with me.

Finally, I owe everything to my wife and my parents for their support.

Table of Contents

List of Tables

List of Figures

Abstract

SECURING THE INFORMATION DISCLOSURE PROCESS

Lei Zhang, PhD

George Mason University, 2010

Dissertation Directors: Sushil Jajodia and Alexander Brodsky

In the problem of information sharing, two goals must be met to fulfill the requirements of both information providers and information consumers. That is, the information providers have constraints of data security/privacy protection, while the information consumers are interested in particular information and want to acquire such information as much as possible.

To solve this problem, disclosure algorithms are applied in the information disclosure process so information providers can compute what to share.

However, in a typical information disclosure process, the applied disclosure algorithm is constructed to check the goals solely on the exact disclosed data of the algorithm's output. This leads to serious security problems in that a malicious information consumer, or namely, the adversary, may be able to acquire additional information from the disclosure algorithm itself that violates the security/privacy constraints of the information providers.

This dissertation presents a number of techniques for answering basic questions about the problem of information sharing: how secure is an information disclosure process, when the disclosure algorithm is known to the public, and if it is not secure, how can we make it so?

This dissertation starts by extending an existing solution to the problem of online query auditing, i.e., whether a posed information request from the information consumer should be permitted or not.

In the problem of online query auditing, an adversary may acquire more precise information than what has been disclosed by the information providers based on the knowledge he or she obtained from the fact that some information requests have been denied. The existing solution, called simulatable auditing, does solve the problem partially by achieving the first goal, which is guaranteeing the security constraints of the information provider. However, it fails to achieve the second goal. That is, many information requests from information consumers will be denied unnecessarily to cause a significant data availability downgrade. This dissertation proposes a new solution that achieves both of the goals by identifying a sufficient and necessary condition for guaranteeing the data protection of the information providers.

This dissertation then studies a more relaxed problem, the problem of micro-data disclosure, in which the disclosure algorithm has to choose what to disclose from multiple candidate data/datasets. The problem of checking whether the security/privacy protection of the information provider has been violated turns out to be much harder in this case, i.e., the general case is an NP-complete problem. This problem has not been given enough attention, and most existing solutions suffer from a failure of the desired data security/privacy protection. This dissertation presents a new model to design safe disclosure algorithms that at least guarantee the data protection of the information providers. Heuristic algorithm design is also proposed to achieve an acceptable good performance for real-life data applications due to the hardness of the problem.

Finally, this dissertation addresses an open problem of how to restore the data security/privacy when it has already been compromised by incidental data disclosure, which is unavoidable when multiple information providers are disclosing the same set of information without collaboration or centralized control. This dissertation shows that, under certain conditions, this can be accomplished by applying a statistical approach.

Chapter 1: Introduction

1.1 Problem Statement

The problem of information sharing has drawn much attention in recent years. To support sharing information on a large scale, we need to ensure privacy while providing as much data utility as possible to users. This problem can be regarded as a game played between two parties, the information provider and the information consumer.

Information providers will hold a set of constraints that the disclosed data/dataset must satisfy. Typically, in a large scale database, the information providers may want to prevent the value of each single item from being disclosed to the information consumers. Therefore, sometimes the term privacy will be used instead of security when these data items refer to the information of individuals.

The information consumers can be divided into two categories: active information consumers and passive information consumers. Active information consumers will generate queries for any information of interest to the information providers. Passive information consumers simply transmit their target information or interests to the information providers, who have full control of what and how the related information is disclosed.

Upon request, the information providers must decide, for either active information consumers or passive information consumers, what information should be disclosed. This process can be represented by the execution of an algorithm, namely, the disclosure algorithm or a disclosure monitor.

The applied disclosure algorithm will take into account the information providers's security constraints, as well as the information consumer's data utility interests, and compute the data items to be disclosed. Regardless of how the data is disclosed, the disclosed data can be represented by a set (could be infinite) of instances, each of which represents a possible state of the entire data set of the information providers. When the disclosure algorithms are non-deterministic, this set will also be associated with a defined PDF. [1]

The typical and well-accepted definition of “safe” is defined by the satisfaction of the security/privacy constraints in this disclosed instance set. And then “safe” disclosure algorithms are compared by the data utility they can provide through the disclosed instance set w.r.t. different interests of the information consumers in different data applications.

However, this dissertation will reveal that this typical view of “safe” in the problem of information disclosure is problematic. To compute the disclosed information is only the second step in the process of disclosing information. There is always a hidden first step of decision making, i.e., the information consumers may have different choices of how to disclose information and the information disclosure algorithm has to decide a proper one and then compute the disclosed data in the second step.

In different data applications, the difficulty of decision making processes varies widely. In this dissertation I focus on two data applications: numerical data disclosure and category data disclosure, respectively.

1.1.1 Online Query Auditing of Numerical Data

Consider the information provider holds a numerical database and the information consumer will pose customized aggregation queries (SUM, MAX, and so on) of a subset of the stored data. The information provider has two choices for a posed query: give precise answer, or simply deny it. The online query auditing problem refers to the following question being asked of the information provider: given a posed query and a sequence of previous received and processed queries, how to choose, such that (1) when the information provider chooses to answer the query, the information consumer cannot determine the value of any individual piece of data; (2) the information provider chooses to deny the query only when (1) is not possible.

[1]Output of a deterministic disclosure algorithm may be also considered as a set associated with a PDF, which is uniform.

Although this is the simplest case among all decision-making processes in the problem of information disclosure, one may still be puzzled by the complexities hidden under it. Assume the information provider uses a straightforward solution as follows: the information provider will first compute the answer for a posed query, and then try to determine whether an individual piece of data can be resolved from the answer and the answers to all the previous queries; the information provider will deny the query if he succeeds in doing so and answer the posed query otherwise.

One may think this solution is easy to understand and should be safe. However, as I have discussed above, by doing so the information provider will fall into the trap of the typical definition of "safe" that is not safe. I will discuss this problem in detail in Chapter 3.

1.1.2 Micro-Data Disclosure Through Generalization

In the problem described above, the information provider only has two choices during the decision-making process. The situation becomes worse when information provider has to choose the way to disclose among many more possibilities.

Consider that the information provider holds a database that contains micro-data about people's medical condition. The information consumer may ask for a set of people's medical conditions with their ages, sex, and living locations. The information provider holds the privacy constraints such that any individual's medical condition cannot be determined by the information consumer. Therefore, the information provider may choose to remove the names of individuals from the disclosed data and generalize tuples into groups by generalizing values of other attributes into higher granularity.

Note that in this case, the information consumer does not have a particular query and the information provider can choose any different generalization of the original data, including disclosing nothing. The inherent rule is that the information provider will try to disclose as much information as possible under the condition that the privacy constraints are satisfied. The related work of how to define the privacy constraints and the data utility property will be given in the next chapter. The idea here is that, even with these properties precisely defined, the information provider still has to choose from all candidate disclosures, the "best" one w.r.t. the data utility and privacy constraints.

Similar to what I have mentioned above, existing solutions to the problem of micro-data disclosure always fall into the same trap where they consider only an unsafe "safe" definition. Therefore, every one of them should be re-studied based on the framework discussed in this dissertation.

Unfortunately, the underlying complexity in this multi-choice case leads to the fact that, unlike the problem of query auditing, this micro-data disclosure problem cannot have computationally feasible optimal solutions. Furthermore, even a heuristic solution is hard to find when guaranteeing a certain level of data utility. Detailed discussion will be given in Chapters 4 and 5.

1.1.3 Multi-Source Micro-Data Disclosure

The typical information disclosure algorithms could also be problematic when considering a multi-source information disclosure. That is, when more than one information provider is disclosing the same set of information without collaboration or centralized control, a desired security/privacy constraint is most likely not able to be maintained. In this dissertation, I will introduce a special kind of information disclosure algorithm to handle this kind of problem that other solutions cannot handle. Detailed discussion will be given in Chapter 6.

1.2 Summary of Contributions

The contributions of this dissertation are summarized as follows.

For the Problem of Query Auditing in Numerical Databases

In Chapter 3, I propose a new model called *simulatable binding*, extending the existing simulatable auditing model for the problem of online query auditing in numerical databases. The intuition is to find a set of database states and bind them together when "denied" is the answer to a posed query, under the condition that the binding set is “large” enough to protect the privacy of its elements.

Both the proposed simulatable binding model and existing simulatable auditing model guarantee privacy constraints for the information providers. However, the proposed simulatable binding model is able to answer the query q_2 in more cases than the simulatable auditing model does.

I should emphasize that the fundamental differences between the proposed model and the simulatable auditing model are as follows:

- Under the simulatable auditing model, a decision whether to answer a newly posed query is made solely on the knowledge that has already been disclosed.

- Under the simulatable binding model, selection of a safe binding for a newly posed query is made solely on the knowledge that has already been disclosed; however, the decision whether to answer the newly posed query is based on not only the disclosed knowledge and the safe binding, but also the secret and true database state.

It is clear that the simulatable binding model provides a more relaxed condition for answering queries. However, I will prove that, the condition provided by this simulatable binding model is not only sufficient, but also necessary to guarantee database privacy in the online auditing problem. As a consequence, I prove that the algorithms applying the simulatable binding model always provide better utility than algorithms applying the simulatable auditing model. It is worth noting that the simulatable binding model is independent of the concerned privacy property, i.e., any privacy property can be applied to the simulatable binding model, which makes the simulatable binding model widely applicable.

In order to show the practicality and efficiency of the simulatable binding model, I present two algorithms for max query online auditing and sum query online auditing, respectively. Each of those algorithms is built in a way that is comparable to an algorithm applying the simulatable auditing model considering the same privacy property. I conduct experiments to show the improved data utility of the simulatable binding model over the simulatable auditing model.

For the Problem of Micro-Data Disclosure

In Chapter 4, I first model disclosed information under the assumption that the malicious information consumer, i.e., the adversary, has the knowledge of what I call a *deterministic disclosure function* (DDF), which is a formal notion of a function defined by a disclosure algorithm. This is done by introducing a formal definition of a *disclosure set.* Intuitively, all the adversary can infer from the disclosed data and the knowledge of the DDF is that a true database state is one of the database states in the *disclosure set.*

Second, given a *safety* predicate p representing the security/privacy constraint of the information provider, I define the notion of *p-safety* for a DDF. Intuitively, it means that for any true database state, the DDF returns an answer, such that the *disclosure set* inferred by the adversary (not just the disclosed answer!) satisfies the safety predicate p. I also define the notion of *p-optimality* for a DDF. Intuitively, it means that, in addition to *p-safety*, there does not exist a *locally better* DDF, in terms of data utility, that is also *p-safe.*

Third I prove that *p-optimal* DDF is computable, although the problem of deciding whether a DDF is *p-optimal* is NP-hard. I then introduce two specific conditions under each of which I prove that the problem of whether a DDF is *p-optimal* is P-time in the size of the set of all possible database states and the size of the generalization sequence. I do this by developing polynomial algorithms to compute a *p-optimal* DDF. Note, however, that the size of all possible database states may be exponential, in the worst case, in the size of a single database state. Clearly, computing *p-optimal* DDFs in such a general setting would not be practical beyond a restricted number of considered database states.

Fourth, I turn to developing a disclosure algorithm for a specific setting of microdata disclosure. For this case, the *locally better* relation on DDFs in terms of data utility is provided by a sequence of quasi-identifier generalizations. I develop a *p-safe* algorithm that is *weakly p-optimal*, and also polynomial with respect to the size of the original table and the generalization lattice.

However, the heuristic algorithms in this family turn out not to have enough good performances. I further discuss the problem of how to design an efficient heuristic algorithm that also has a relatively good performance in Chapter 5.

I propose a novel concept of coincident generalization as an efficient solution to micro-data disclosure using public algorithms. I derive necessary and sufficient conditions for any generalization algorithm to be coincident in order to facilitate the design of such algorithms. As the next step, I instantiate the concept by devising two classes of efficient generalization algorithms: pair-based generalization algorithm, and interleaved generalization algorithm, and prove them to be coincident. The effectiveness of the proposed algorithms is independent of privacy properties, which makes them applicable to a wide range of applications. Experiments show that those algorithms lead to good data utility and performance as well.

For the Problem of Multi-Source Information Disclosure

In Chapter 6, I first propose a new property, called γ-Privacy, for privacy protection in a micro-data disclosure problem when multiple views are disclosed. Given the disclosed views and publicly available information, the set PIS of "all possible worlds"(*i.e., possible tables that would yield the same disclosure results*) is defined. γ-Privacy intuitively means that in a randomly (uniformly) selected instance, the probability of any individual to be associated with a sensitive value is at most γ. I then prove that, for the case of a single disclosed view, γ-Privacy is equivalent to the property of Recursive $(\frac{\gamma}{1-\gamma}, 2)$-Diversity. This means that the property is a "natural" extension of l-Diversity, which is defined only for a single disclosed view, to multiple views.

Second, I prove that deciding on whether γ-Privacy is satisfied by a set of disclosed views is #P-complete. Third, to mitigate the high computational complexity, I relax the property of γ-privacy to be satisfied with (ϵ, θ) confidence, i.e., that the probability of disclosing a sensitive value of an individual be at most $\gamma + \epsilon$ with statistical confidence θ, where ϵ is an arbitrary small positive constant. I propose a Monte Carlo-based algorithm to check the relaxed property in $O((\lambda\lambda')^4)$ time for constant ϵ and θ, where λ is the number of tuples in the original table and λ' is the number of different sensitive values in the original table.

Finally, I turn to the problem of restoring compromised privacy. Namely, given a set of disclosed views that violates γ-Privacy, can we extend it to a superset of views that jointly satisfy γ-Privacy. I propose heuristic polynomial time algorithms which are based on enumerating and checking additional disclosed views. I conduct a preliminary experimental study on heart disease records taken from the UCI data repository ([1]), which demonstrates that the proposed polynomial algorithms restore privacy in up to 60% of compromised disclosures. I also discuss how to apply the proposed technique under different assumptions when the adversary is also aware of the proposed technique.

Chapter 2: Related Work

To solve the problem of protecting the data privacy in a statistical database [2], different methods have been introduced. One way is to perturb the data in the answers for the posed queries [3–5]. Another choice is to perturb the data in the database itself before answering the posed queries [6–10].

Besides these noise-addition based approaches, work on the auditing problem, where responses to queries are either true answers or "denied," has also been proposed. An *off-line auditing problem* [11–15] is to decide whether the database privacy has been breached based on a set of posed queries and their answers. Complexity analyses on specific problems on max/min [11, 14] and sum [12–14] queries are given.

In [16–18], the authors target the *online auditing problem*, that is, to decide whether the database privacy will be breached by giving a true answer to a newly posed query based on a set of posed queries and their answers. [18] also provides a logic-oriented model for the online auditing problem that combines modified answers and denials to enforce the database privacy.

Recently, in [19], the authors uncover a fact that the database privacy may be breached by an attacker with the help of information leaked in the online auditing process, i.e., to decide how to answer a posed query. The authors also provide a model called *simulatable auditing* [19,20] to prevent information leakage in an auditing process. However, because the conditions provided by the simulatable auditing model are far from necessary to guarantee the database privacy, the huge data utility loss in their solutions inspired my work.

The problem of micro-data disclosure has also been extensively studied [21–25], where the security issue discussed in this dissertation is largely ignored. In particular, data swapping [26–28] and cell suppression [29] both aim to protect micro-data released in census tables. However, the amount of privacy is usually not measured in those earlier works. Miklau et al. presents an interesting measurement of information disclosed through tables based on the perfect secrecy notion by Shannon [30]. The important notion of k-anonymity is proposed as a model of privacy requirement [31], which has received extensive studies in recent years. To achieve optimal k-anonymity (with the most utility) is shown to be computationally infeasible [32].

A model based on the idea of blending individuals in a crowd was proposed in [33]. A personalized requirement for anonymity was studied in [34]. In [35], the authors approached the issue from a different perspective, where the privacy property is based on generalization of the protected data and could be customized by users. Many efforts have been made around developing efficient k-anonymity algorithms [31, 36–39], whereas the security of the k-anonymity model is assumed. Two exceptions are the l-diversity notion proposed in [40] and the t-closeness notion proposed in [41], which address the deficiency of k-anonymity of allowing insecure groups with a small number of sensitive values. Algorithms developed for k-anonymity can be extended to l-diversity and t-closeness, but they still share the same security issue addressed in this dissertation because they do not take into account an adversary's knowledge about generalization algorithms. When such knowledge is assumed, most existing generalization algorithms become insecure. In [42], the authors pointed out the above problem and proposed a model for the adversary's knowledge, but did not give any efficient solution for the general micro-data disclosure problem. In [43], the authors studied the problem of how k-anonymity is preserved in multiple views, which first addresses the fundamental problem of Chapter 6 in this dissertation.

Chapter 3: Optimized Online Query Auditing

3.1 Introduction

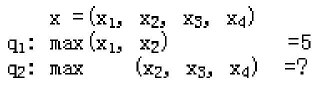
Let $X = \{X_1, X_2, \ldots, X_n\}$ be a statistical database consisting of n variables. Generally, X_i are all real numbers. I use the vector $x = (x_1, x_2, \ldots, x_n)$, where $x_i \in R (1 \leq i \leq n)$, to denote a database state. All queries over X take the form $q : R^n \to R$.

The following problem is known as the *online query auditing problem* ([16–18]): Suppose that a set of queries $q_1, q_2, \ldots, q_{T-1}$ has already been posed and the corresponding answers $a_1, a_2, \ldots, a_{T-1}$ have been given, where each answer $a_i, 1 \leq i < T$, is either the true answer to the query or "denied." Given a new query q_T, the database should give the true answer if the privacy of the database is not breached (i.e., an x_i cannot be determined); otherwise, it should give "denied" as the answer.

3.1.1 The Failure of a Simple Strategy

A simple strategy, denoted as A_{simple}, is to deny q_T if database privacy may be breached when a true answer of q_T is given, and provide the true answer otherwise. Surprisingly, as observed recently by Kenthapadi, Mishra, and Nissim [19], the privacy of the database could still be breached if A_{simple} is applied.

To see this, consider the following example: we have a database consisting of four variables $x = (x_1, x_2, x_3, x_4)$, all of which are integers. Suppose that the first query $q_1 : max(x_1, x_2)$ has been posed and the true answer, say $a_1 = 5$, has been given. Suppose the next query is $q_2 : max(x_2, x_3, x_4)$. Based on the strategy A_{simple}, if the true answer happens to be $a_2 \geq 5$, then the database will return the true answer a_2 for q_2 because the database privacy will not be breached. On the other hand, if the true answer is $a_2 < 5$, the database will deny q_2 because by giving a_2, the database will disclose the the true value of x_1.



```
    x =(x1, x2, x3, x4)
q1: max(x1, x2)            =5
q2: max    (x2, x3, x4)  =?
```

Figure 3.1: Answering Two Max Queries

Unfortunately, this is not enough to protect the database privacy. The problem is that if q_2 gets “denied,” an outside adversary can still determine that $x_1 = 5$. This is because the only reason for the denial of q_2 under such condition is that $a_2 < 5$, which leads to the fact that $x_1 = 5$.

3.1.2 The Idea of Simulatable Auditing

Intuitively, the reason that the simple strategy A_{simple} fails is that it does not take into consideration the information flow from the true database state to the auditing decision. The *simulatable auditing* model proposed by Kenthapadi, Mishra, and Nissim [19] guarantees that the decision making process does not leak any information. This is because whether to answer a newly posed query is decided based on knowledge that an adversary has already acquired, including all posed queries and all given answers.

In general, the strategy of a *Simulatable Auditor*, denoted as A_{sa}, can be stated as the following: Given a set $\mathcal{X}$ of all possible database states that are consistent with previously posed queries $(q_1, q_2, \ldots, q_{T-1})$ and their answers $(a_1, a_2, \ldots, a_{T-1})$, a newly posed query q_T will be denied if:

$$\exists x' \in \mathcal{X}, \text{ the privacy of } x' \text{ will be breached if the true answer } a_T \text{ of } q_T(x') \text{ is given.}$$

If we apply A_{sa} to the example shown in Figure 3.1, q_2 will always be denied no matter what possible answer a_2 is.

3.1.3 Unnecessary Loss of Utility in Simulatable Auditing

Although strategy A_{sa} prevents the information flow from the true database state to the decision making process during auditing, it does so at a large cost of data utility, much of which is unnecessary. In the example shown in Figure 3.1, in order to protect the data privacy, A_{sa} will always deny q_2. Later I will show that we should be able to do much better than this.

The situation gets much worse when the simulatable auditing model is applied to sum queries. When there exist some database constraints, which happens in most cases, A_{sa} could become a strategy that will refuse all queries [19]. For example, in a database $x = (x_1, x_2, \ldots, x_n) \in R^n$, if we have the database constraints $x_i \geq 0, (1 \leq i \leq n)$, for any sum query $q : \sum_{i=0}^{n} b_i x_i, (b_i \in \{0, 1\})$, we will give a "denied" answer based on A_{sa} because there always exists a possible database state $x' = (0, 0, \ldots, 0)$ that if the true answer $a = q(x')$ is given, some x_i will be disclosed.

The problem of information disclosure can be regarded as an optimization problem, i.e., to maximize the data utility of the disclosed data under the constraint that data privacy must be guaranteed. This key observation leads to the problem of finding the sufficient and necessary condition to guarantee database privacy, which is the main focus of this chapter.

Indeed, the model of simulatable auditing model only provides a sufficient condition that is far from necessary. Such a sufficient condition may lead to significant and unnecessary data utility loss, as discussed above. I will show that, by applying the technique proposed in this chapter, such unnecessary utility loss can be completely avoided.

Contribution

In this chapter, I propose a new model called *simulatable binding*. The idea is to find a set of database states and bind them together when "denied" is the answer to a posed query, under the condition that the binding set is "large" enough to protect the privacy of its elements.

To illustrate, consider the example shown in Figure 3.1 I have discussed. The strategy A_{simple} fails to guarantee the database privacy because the denial of q_2 will lead to the fact that $a_1 = 5$. The strategy A_{sa} of the simulatable auditing model deny q_2 unconditionally to protect the privacy of all possible database states. However, what we can do is to deny q_2 not only when the true answer of q_2 satisfies $a_2 < 5$, but also when $a_2 = 5$. Therefore,

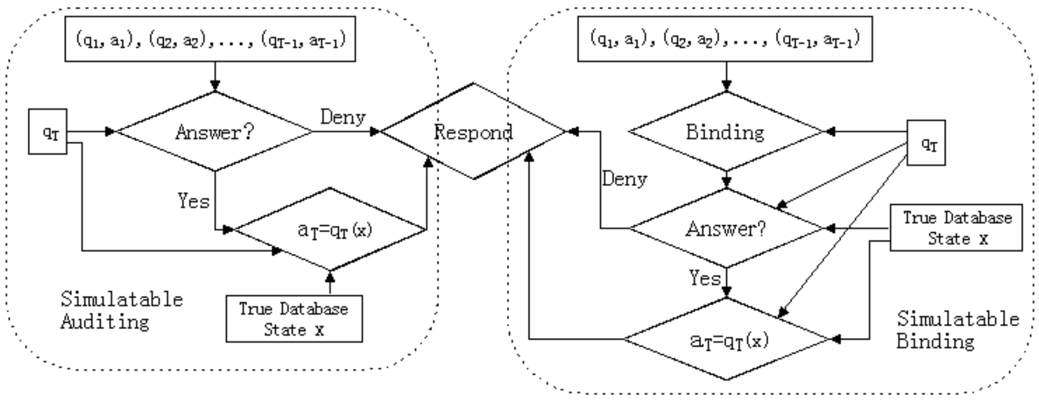


Figure 3.2: Comparison of Simulatable Binding and Simulatable Auditing Frameworks

an outside adversary cannot determine any of the $\{x_1, x_2, x_3, x_4\}$ if he gets "denied" when posing query q_2.

The result of such a strategy, denoted as A_{sb}, is shown in the following table with comparisons:

a_2	A_{simple}	A_{sa}	A_{sb}
$a_2 < 5$	Deny	Deny	Deny
$a_2 = 5$	Answer	Deny	Deny
$a_2 > 5$	Answer	Deny	Answer
Privacy Guarantee?	No	Yes	Yes

Both A_{sa} and A_{sb} guarantee privacy. However, A_{sb} is able to answer the query q_2 in more cases than A_{sa} does. Note that although a binding set can protect the privacy of its element database states, the decision process for determining such a set may contain additional information about the true database state.

Therefore we need to make sure this binding decision is made based on the knowledge that is already known to the outside adversary, leading to the name *simulatable binding.* Conceptually, the framework of the simulatable binding model compared to the simulatable auditing model is shown in Figure 3.2. I should emphasize that the fundamental differences between this model and the simulatable auditing model are as follows:

- Under the simulatable auditing model, a decision whether to answer a newly posed query is solely based on the knowledge that has already been disclosed.

- Under the simulatable binding model, selection of a safe binding for a newly posed query is solely based on the knowledge that has already been disclosed; however, the decision whether to answer the newly posed query is based on not only the disclosed knowledge and the safe binding, but also the secret and true database state.

It is clear that the simulatable binding model provides a more relaxed condition for answering queries. However, I will prove that the condition provided by this chapter's simulatable binding model is not only sufficient but also necessary to guarantee database privacy in the online auditing problem. I also prove that the algorithms applying the simulatable binding model always provide better utility than algorithms applying the simulatable auditing model. It is worth noting that the simulatable binding model is independent of the concerned privacy property, i.e., any privacy property can be applied to the simulatable binding model, which makes the simulatable binding model widely applicable.

In order to show the practicality and efficiency of the simulatable binding model, I present two algorithms for max query online auditing and sum query online auditing, respectively. Each of those algorithms is built in a way that is comparable to an algorithm applying the simulatable auditing model considering the same privacy property. I conduct experiments to show the improved data utility of the simulatable binding model over the simulatable auditing model.

3.2 The Model of Simulatable Binding

To build intuition, I introduce the *simulatable binding* model through the following simple graphical example:

- Consider a database state as a point and the current knowledge of an outside adversary as a set of points $\mathcal{X}$. Intuitively, $\mathcal{X}$ contains exactly all database states that are possible from the point of view of an outside adversary.

- I state the database privacy requirement as that an outside adversary cannot determine any point in $\mathcal{X}$ to be the true database state. Therefore, a set of possible database states $\mathcal{X}'$ is safe if $|\mathcal{X}'| > 1$. I assume that $\mathcal{X}$ is safe at the beginning, which means that $|\mathcal{X}| > 1$.

- A newly posed query q over the database is a partition of $\mathcal{X}$, $q = \{s_1, s_2, \ldots, s_n\}$. The true answer to q is $s_i, 1 \leq i \leq n$ such that $x \in s_i$. The database auditor has to decide whether to give the true answer for q, or to deny it.

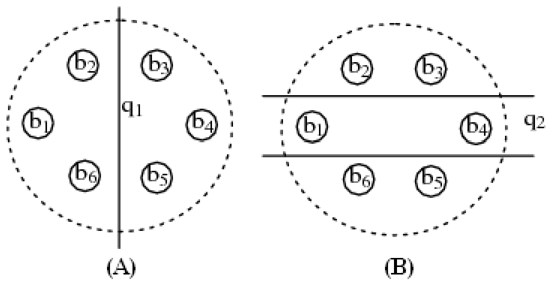


Figure 3.3: Queries as Partitions

Note that, in this toy example, we consider a very naive privacy requirement. In the general problem setting, the privacy property can regarded as a predicate p and we say a set of possible database states $\mathcal{X}$ is safe if $p(\mathcal{X}) = true$.

As an example, let $\mathcal{X} = \{b_1, b_2, \ldots, b_6\}$. Figure 3.3(A) shows a potential query $q_1 = \{\{b_1, b_2, b_6\}, \{b_3, b_4, b_5\}\}$. In (B), $q_2 = \{\{b_1, b_4\}, \{b_2, b_3\}, \{b_5, b_6\}\}$ denotes another potential query.

If we follow the traditional simple strategy for the online auditing problem, we will give the true answer for a newly posed query q if the size of the true answer, being a set, is safe. In this case, the simple strategy can be stated as:

> A_{simple}: Given $\mathcal{X}$, x and a newly posed query q, let a denote the true answer for q: (1) a "denied" answer is given if the set $\mathcal{X} \cap a$ is not safe; (2) a is given otherwise.

In this example, the safety of a set means its size is larger than 1. Clearly, in both cases (A) and (B), if the true answer of the newly posed query is given, an outside adversary still cannot determine the true database state. Thus, we will give the true answer for q_1 and q_2 in the two cases, respectively.

The database privacy will certainly be guaranteed in any case where A_{simple} returns the true answer for the newly posed query. In fact, all algorithms I discuss in this chapter will meet this requirement.

However, the safety of a strategy requires guaranteeing privacy not only when the true answer is given, but also when "denied" is given. Let $\eta_{(A,\mathcal{X},q)}$ denote the subset of possible database states in $\mathcal{X}$ where q will be denied by a strategy A. I define the following:

Definition 1 (deny-safe)**.** *A strategy A is said to be* deny-safe *if:*

$$\forall \mathcal{X}, \forall q, |\mathcal{X}| > 1 \Rightarrow \eta_{(A,\mathcal{X},q)} = \phi \vee \eta_{(A,\mathcal{X},q)} \ is\ safe$$

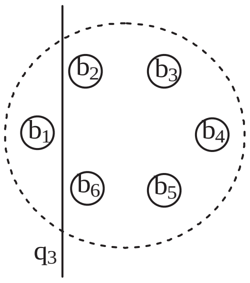


Figure 3.4: A Query that A_{simple} Cannot Answer Safely

Unfortunately, A_{simple} is not a *deny-safe* strategy. Figure 3.4 shows a situation, where $|\eta_{(A_{simple},\mathcal{X},q_3)}| = 1$. In this case, if the newly posed query q_3 is denied, an outside adversary would determine the true database state to be b_1.

Now we consider the strategy of the simulatable auditing model:

> A_{sa}: Given $\mathcal{X}$, x and a newly posed query q: (1) a "denied" answer is given if $\exists s \in q, \mathcal{X} \cap s$ is not safe; (2) the true answer of q is given otherwise.

It is clear that, in the situation in Figure 3.4, q_3 will always be denied no matter which point represents the true database state.

Theorem 1. *A_{sa} is* deny-safe*.*

This is an obvious result because the definition of A_{sa} naturally implies that $\eta_{(A,\mathcal{X},q)} = \phi$ or $\eta_{(A,\mathcal{X},q)} = \mathcal{X}$, where $\mathcal{X}$ is safe as assumed. But A_{sa} denies more queries than what is necessary. Instead, a better way is described as follows.

- we select another query (partition) $q' = \{\{b_1, b_6\}, \{b_2, b_3\}, \{b_4, b_5\}\}$ on $\mathcal{X}$;
- Let a and a' denote the true answer for q_3 and q', respectively. We will deny q_3 not only if "a *is not safe*" but also if "$a' \cap \eta_{(A_{simple},\mathcal{X},q_3)} \neq \phi$."

As illustrated in Figure 3.5, by applying the above strategy, if q_3 gets denied, an outside adversary still cannot determine whether b_1 or b_6 is the true database state. This illustrates the idea of a binding strategy. And the safety of such a strategy depends on the selected query q'.

Definition 2 (safe binding)**.** *Given a set $\mathcal{X}$ and a query q, a safe binding q' is a partition of $\mathcal{X}$ such that:*

(1) $\forall s' \in q'$, s' is safe, and

(2) $\forall s \in q$, if s is safe, then $s \backslash s^*$ is either ϕ or a safe set, where $s^* = \bigcup_{s' \in q', s' \cap \eta_{(A_{simple},\mathcal{X},q)} \neq \phi} s'$.

A binding strategy can be stated as follows:

A_b: Given $\mathcal{X}$, x, a safe binding q', and a newly posed query q, let a' denote the true answer for q':

(1) a "denied" answer is given for q if $a' \cap \eta_{(A_{simple},\mathcal{X},q)} \neq \phi$

(2) the true answer for q is given otherwise.

Note that a safe binding q' protects the privacy of the true database state because $\eta_{(A_b,\mathcal{X},q)}$ is determined by "safe" elements of q'. However, in order to prevent inference about the true database state, we still need to be aware of the potential information flow from the true database state to the selection of q'.

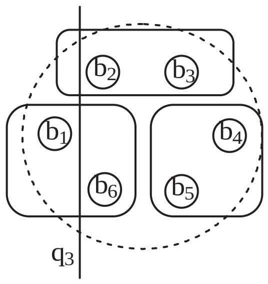


Figure 3.5: A Safe Binding for Query q_3

Definition 3 (simulatable binding)**.** *A binding strategy is said to be* simulatable*, if the safe binding is selected based on $\mathcal{X}$ and the newly posed query.*

Theorem 2. *If the desired safety property satisfies:*

$$\forall \mathcal{X}' \subseteq \mathcal{X}'', \mathcal{X}' \; is \; safe \Rightarrow \mathcal{X}'' \; is \; safe$$

A simulatable binding*, denoted as A_{sb}, is* deny-safe*.*

Proof: For any set $\mathcal{X}$, any query q, let q' denote the safe binding of A_{sb}. Suppose that $\eta_{(A_{sb},\mathcal{X},q)} \neq \phi$. Then there exists a possible database state $b \in \mathcal{X}$ such that $b \in \eta_{(A_{sb},\mathcal{X},q)}$. Based on the definition of a binding strategy, there exists $s \in q'$ such that $b \in s \wedge s \subseteq \eta_{(A_{sb},\mathcal{X},q)}$. Because q' is a safe binding, we have that s is safe. Therefore $\eta_{(A_{sb},\mathcal{X},q)}$ is safe.

I slightly extend this as follows:

Definition 4 (ultimate simulatable binding)**.** *A simulatable binding is said to be* ultimate *if any selected safe binding q' based on $\mathcal{X}$ and q satisfies:*

$$\exists s \in q', \eta_{(A_{simple},\mathcal{X},q)} \subseteq s$$

The following theorem is straightforward:

Theorem 3. *An* ultimate simulatable binding*, denoted as A_{usb}, is* deny-safe*.*

Until now, I have proven that both A_{sa} and A_{sb} are *deny-safe*. Also, they are both safe in the case when true answers are given for a posed query. For A_{sb}, this is guaranteed by condition (2) in the definition of a "safe binding."

However, the data utility provided by these two models is quite different. Next, I prove that any A_{sb} provides more data utility than A_{sa} by means of the following theorem.

Theorem 4. *Given $\mathcal{X}$ and a query q:$\eta_{(A_{sb},\mathcal{X},q)} \subseteq \eta_{(A_{sa},\mathcal{X},q)}$.*

Proof: Because $\eta_{(A_{sa},\mathcal{X},q)}$ is either an empty set or the set $\mathcal{X}$ itself, it is sufficient to prove that $\eta_{(A_{sa},\mathcal{X},q)} = \phi \Rightarrow \eta_{(A_{sb},\mathcal{X},q)} = \phi$. This is a natural implication by the definition of A_{sa}. Because if $\eta_{(A_{sa},\mathcal{X},q)} = \phi$, there will be no $b \in \mathcal{X}$ for which A_{simple} will output "denied." Thus the condition for A_{sb} to deny q will never be satisfied. Thus we will have $\eta_{(A_{sb},\mathcal{X},q)} = \phi$.

In the example shown in Figure 3.5, A_{sa} will deny to answer q_3 no matter what the true database state is and A_{sb} will deny to answer q_3 only if the true database state is $x = b_1$ or $x = b_6$. Furthermore, I prove that:

Theorem 5. *Given any strategy A, $\mathcal{X}$, and a newly posed query q, if for any $x \in \mathcal{X}$, to disclose the answer for q based on A does not violate the safety of the database, then:*

(1) there exists a safe binding q', such that A is identical to a binding strategy A_b based on q' with respect to the knowledge that any outside adversary could obtain, and

(2) q' is independent of the true database state.

Proof: To prove (1), construct q' as the follows:

- Let $s_1 = \eta_{(A,\mathcal{X},q)}$ and $s_1 \in q'$;
- Let $s^* = \mathcal{X} \setminus s_1$;
- While $s^* \neq \phi$:
  - Select $x \in \mathcal{X} \setminus s_1$, let s_x be the set of all possible database states that an outside adversary could obtain after the true answer for query q based on A is given, and $s_x \in q'$;
  - Let $s^* = s^* \setminus s_x$;

Because A is safe, it is clear that q' is a safe binding and (1) is true. (2) is also clear because the following condition must be satisfied when considering an adversary's knowledge: $\forall x, x' \in \mathcal{X}, x' \in s_x \Leftrightarrow x \in s_{x'}$

With Theorem 5, I have shown that the simulatable binding model provides not only a sufficient, but also a necessary condition to guarantee the database privacy for the online auditing problem.

However, we can see that the process of selecting a safe binding is only required to be simulatable, which means that the safe binding that can be selected is not unique. Therefore, the performance of a simulatable binding can be influenced by the way that a safe binding is "simulatably" selected. The problem is that "what is the best safe binding given $\mathcal{X}$ and q" cannot be uniformly defined. Because a "safe binding" naturally creates a dependency between different possible database states within the set $\mathcal{X}$, such preference in dependency really depends on the real-time application and the users' requirements.

Generally, there are two different ways to select a "safe binding" in a simulatable binding with respect to the treatment to the set $\eta_{(A_{simple}, \mathcal{X}, q)}$:

- Bind the entire set $\eta_{(A_{simple}, \mathcal{X}, q)}$ together with other selected possible database states in $\mathcal{X} \setminus \eta_{(A_{simple}, \mathcal{X}, q)}$.
- Bind the elements in the set $\eta_{(A_{simple}, \mathcal{X}, q)}$ separately with different selected possible database states in $\mathcal{X} \setminus \eta_{(A_{simple}, \mathcal{X}, q)}$.

The first way represents the preference that tries to remain the data utility that can be safely provided by the simple strategy A_{simple} for the elements in the set $\mathcal{X} \setminus \eta_{(A_{simple}, \mathcal{X}, q)}$. The second way represents the preference that tries to obtain a more balanced result, i.e., to provide better utility for the possible database states in the original none-guaranteed set $\eta_{(A_{simple}, \mathcal{X}, q)}$ than the first way, by sacrificing a little more data utility for other possible database states.

In next section, I discuss applications of these two different ways in two particular online auditing problems.

3.3 Two Practical Algorithms

In this section, I provide algorithms based on the *simulatable binding* model for the problems of auditing max query and auditing sum query, respectively. Note that, when considering a practical problem, two changes are noticeable: 1) the set $\mathcal{X}$ is no longer a static set, but is defined by all the previously posed queries and their answers; 2) the privacy property may be quite different and involved in different applications. In fact, I consider two different privacy properties for the two cases.

3.3.1 A Simulatable Binding Algorithm for Max Query Auditing

I consider the following online auditing problem of max query. This is the same problem setting used in [19].

1. The database consists of n variables, all of which are real-valued, where a database state is denoted as a vector $x = (x_1, x_2, \dots, x_n) \in R^n$;

2. The database privacy is said to be guaranteed if an outside adversary cannot determine the value of any variable in the database. I say that a set of possible database states $\mathcal{X}'$ is safe if: $\forall i, 1 \leq i \leq n, \exists x', x'' \in \mathcal{X}', x'_i \neq x''_i$. Note that $\forall \mathcal{X}' \subseteq \mathcal{X}''$, therefore $\mathcal{X}'$ is safe implies that $\mathcal{X}''$ is safe.

3. A query q over the database is to ask the maximum value of a set of the variables in the database. Let Q denote the corresponding set, the true answer a for q is computed as $a = max\{x_i | x_i \in Q\}$.

4. The problem is: Given a set of queries $q_1, q_2, \dots, q_{T-1}$ and their answers, how to answer a newly posed query q_T, providing as much data utility as possible, while the database privacy is also guaranteed.

Theorem 6. *There exists a simulatable binding algorithm for the max query auditing problem above that runs in $O(T \sum_{i=1}^{T} |Q_i|)$ time where $|Q_i|$ represents the number of variables in q_i.*

```
Algorithm 3.1: MAX simulatable binding
    For $i = 1$ to $2T + 1$ do
      Let $a_T$ be $\alpha[i]$;
      If $a_T$ is consistent with previous answers $a_1, a_2, \ldots, a_{T-1}$ AND
      there exists $1 \leq j \leq n$ such that $x_j$ is uniquely determined (using [11])
        then set $\beta[i]$ to "true";
    If all the $\beta[i], (1 \leq i \leq 2T + 1)$ have been set to "true"
      then return "denied";
    Select the least $k \in [1..2T + 1]$ such that $\beta[k] =$"false";
    Let $a_T$ be the true answer for $q_T$;
    If there does not exist $1 \leq j \leq n$ such that $x_j$ is uniquely determined (using [11])
    AND $a_T \neq \alpha[k]$
      then return the true answer $a_T$;
    If all the $\beta[i], (1 \leq i \leq 2T + 1)$ remain "false"
      then return the true answer $a_T$;
    Record the the answer for $q_T$ as $\alpha[k]$ (*);
    Return "denied";
*: This record serves only for future auditing.
```

Figure 3.6: Algorithm 3.1

I prove this theorem by providing Algorithm 3.1, which is built comparable to the algorithm proposed in [19] for the same problem setting. I adopt the process that is proposed in the work [11], which has already solved the off-line max auditing problem. Some of the parameters are set as follows: Let $q'_1, q'_2, \ldots, q'_{T-1}$ be the previous queries with $a'_1 \leq \cdots \leq a'_{T-1}$ as the corresponding true answers. Let $a'_L = a'_1 - 1$ and $a'_R = a'_{T-1} + 1$. Let $\alpha[1..2T+1]$ be an array that takes the value of the sequence $(a'_L, a'_1, \frac{a'_1+a'_2}{2}, a'_2, \frac{a'_2+a'_3}{2}, a'_3, \ldots, a'_{T-2}, \frac{a'_{T-2}+a'_{T-1}}{2}, a'_{T-1}, a'_R)$. Let $\beta[1..2T + 1]$ be a boolean array with default values of "false."

In [19], the authors have already shown that in order to check whether $\eta_{(A,\mathcal{X},q)} \neq \phi$ for such a problem setting, it is sufficient to check whether it is not safe to answer q_T under the condition that the true answer for q_T is any one of the values listed in the array $\alpha[1..2T+1]$. Thus, the algorithm they proposed based on the simulatable auditing model is to deny q_T if $\eta_{(A,\mathcal{X},q)} \neq \phi$.

Comparably, the Algorithm 3.1 that I propose, based on the simulatable binding model, provides a way to select a safe binding that binds the entire set $\eta_{(A,\mathcal{X},q)}$ together with the set of possible true database states, which is consistent with $\mathcal{X}$ and the least "safe" answer $\alpha[k]$ for q_T. For other possible database states, the true answer will be given for the query q_T. Clearly, this applies the first kind of strategy to select a safe binding as I have discussed in the previous section.

Note that by selecting a safe binding in this way, I need to log the returned answer of q_T as $\alpha[k]$ (in Algorithm 3.1) while the actually returned answer is "denied." The logged answer is used to define a new $\mathcal{X}'$ representing the knowledge contained in the denial for future auditing. In order to guarantee the safety of the binding in future auditing, the answers have to be consistent not only with $q_1, q_2, \ldots, q_{T-1}$ and their answers, but also with q_T and its answer $\alpha[k]$.

The selection of the $\alpha[k]$, bound with "denied," is based on the order of α and independent from the true database state. Thus I have that:

Lemma 1. *Algorithm 3.1 is a* simulatable binding.

Proof Sketch: Clearly, the selection of binding in Algorithm 3.1 is based on $q_1, \ldots, q_T$ and $a_1, \ldots, a_{T-1}$. It suffices to prove that the binding in Algorithm 3.2 is a safe binding. Algorithm 3.1 tends to bind the entire denial set of A_{simple} with a set that already satisfies the privacy requirement. Therefore, the proof of safe binding requires: (1) the desired safety property satisfies that $\forall \mathcal{X}' \subseteq \mathcal{X}''$, $\mathcal{X}'$ is safe implies that $\mathcal{X}''$ is safe, and (2) it is sufficient to check values in the array $\alpha[1..2T+1]$ to determine $\eta_{(A_{simple}, \mathcal{X}, q_T)}$. (1) is clear and (2) has already been proved in the work [19]. Besides, the complexity of Algorithm 3.1, $O(T \sum_{i=1}^{T} |Q_i|)$ can be computed from the algorithm itself including the process from [11].

In fact, I have illustrated the result of Algorithm 3.1 in Section 3.1.4, when applying to the example I discussed in the introduction.

3.3.2 A Simulatable Binding Algorithm for Sum Query Auditing

I consider the following online auditing problem of sum query:

1. The database consists of n variables, all of which are real-valued, where a database state is denoted as a vector $x = (x_1, x_2, \dots, x_n) \in R^n$.

2. There is a set of constraints $\mathcal{C}$: for each x_i: $c_{i,l} \le x_i \le c_{i,r}, (c_{i,l}, c_{i,r} \in R, c_{i,r} - c_{i,l} \ge 1)$.

3. I adopt the similar privacy requirement used in [44]. The database privacy is said to be guaranteed if an outside adversary cannot determine the value of any variable within an interval of 1. I say that a set of possible database states $\mathcal{X}'$ is safe if: $\forall i, (1 \le i \le n), \exists x', x'' \in \mathcal{X}', |x'_i - x''_i| \ge 1$. Note that $\forall \mathcal{X}' \subseteq \mathcal{X}''$, therefore $\mathcal{X}'$ is safe implies that $\mathcal{X}''$ is safe.

4. A query q over the database asks the sum value of a set of the variables in the database. Let Q denote the corresponding set, the true answer a for q is computed as $a = sum\{x_i | x_i \in Q\}$.

5. The problem is: Given a set of queries $q_1, q_2, \dots, q_{T-1}$ and their answers, how to answer a newly posed query q_T, providing as much data utility as possible, while the database privacy is also guaranteed?

Theorem 7. *There exists a simulatable binding algorithm for the sum query auditing problem above that runs in polynomial time w.r.t. n, T.*

I provide Algorithm 3.2 based on the following two tests:

Test 1: Given a set of sum queries $q_1, q_2, \dots, q_{T-1}$ and their corresponding answers $a_1, a_2, \dots, a_{T-1}$, a set of constraints $\mathcal{C} = \{c_{i,l} \le x_i \le c_{i,r}, (1 \le i \le n)\}$, and a new query q_T, a real value $r \in R$ satisfies Test 1 if: there exists $x \in R^n$ such that x satisfies $\mathcal{C}$ and x is consistent with the set of sum queries $q_1, q_2, \dots, q_T$ and their corresponding answers $a_1, a_2, \dots, a_T$, where $a_T = r$.

Test 2: Given a set of sum queries $q_1, q_2, \dots, q_{T-1}$ and their corresponding answers $a_1, a_2, \dots, a_{T-1}$, a set of constraints $\mathcal{C} = \{c_{i,l} \le x_i \le c_{i,r}, (1 \le i \le n)\}$, and a new query q_T, a real value $r \in R$ satisfies Test 2 if each of the values $r-1$, r, $r+1$ satisfies Test 1, given the same setting.

As I have mentioned in the introduction, the problem of deciding whether a variable can be determined by a sequence of answered sum queries under the condition that there is no constraint on the database state, has been solved in [14]. For the sake of simplicity, in Algorithm 3.2, I adopt the process proposed in [14] when checking the same problem. On the other hand, when constraints exist, any simulatable auditing algorithm that guarantees the safety of the entire database will never answer a single query.

In Algorithm 3.2, I apply this chapter simulatable binding model using the following way to select a safe binding: I bind the set of possible database states that is consistent with an "unsafe" answer r', i.e., it does not satisfy Test 2, with another set of possible database states that is consistent with a "safe" answer $r' + 1$ or $r' - 1$, i.e., it does satisfy Test 2. Note that when r' is an "unsafe" answer, at most one of $r' + 1$ or $r' - 1$ could possibly be a "safe" answer. Similar to Algorithm 3.1, I need to log the returned answer of q_T as $r' + 1$ or $r' - 1$ (corresponding to r in Algorithm 3.2) while the actually returned answer is "denied." The logged answer is used to define a new $\mathcal{X}'$ representing the knowledge contained in the denial for future auditing. That is, in order to guarantee the safety of the binding, in future auditing, the answers have to be consistent not only with $q_1, q_2, \ldots, q_{T-1}$ and their answers, but also with q_T and its answer as logged.

Also, it is clear that this selection applies the second kind of strategy to select a safe binding as I have discussed in the previous section. And I have that:

Lemma 2. *Algorithm 3.2 is a* simulatable binding.

Proof Sketch: Clearly, the selection of binding in Algorithm 3.2 is based only on $q_1, \ldots, q_T$ and $a_1, \ldots, a_{T-1}$. It suffices to prove the binding in Algorithm 3.2 is a safe binding. As shown in Algorithm 3.2, in the deny set $\eta_{(A_{sb}, \mathcal{X}, q_T)}$, where $\mathcal{X}$ is defined by $q_1, q_2, \ldots, q_{T-1}$ and their corresponding answer $a_1, a_2, \ldots, a_{T-1}$, there always exist two set of database states and a real value r'' such that (1) one set of database states is consistent with $\mathcal{X}$, q_T and its corresponding answer r'' and (2) the other set of database states is consistent with $\mathcal{X}$, q_T and its corresponding answer $r''+1$. In each set, because the database privacy has not been breached before q_T is posed, regardless to q_T and its answer, none

Algorithm 3.2: SUM simulatable binding

```
Use [14] to decide $q_T$ without considering $\mathcal{C}$:
  If [14] denies $q_T$ then return "denied";
Let $r$ be the true answer for $q_T$ computed from $x$;
If $r$ satisfies Test 2 do
  If both $r-1$ and $r+1$ satisfy Test 2, do
    Let $a_T = r$, return $a_T$;
  Else do
    Record the answer for $q_T$ as $r$ (*);
    Return "denied";
Else do
  If $\exists r', r' = r-1 \vee r' = r+1$, such that $r'$ satisfies Test 2, do
    Record the answer for $q_T$ as $r'$ (*);
    Return "denied";
  Else do
    Return "denied";
```

*: This record serves only for future auditing.

Figure 3.7: Algorithm 3.2

of x_i can be determined within an interval of 1. Thus, for any i, there always exist two possible database states, $x' = (x'_1, x'_2, \ldots, x'_n)$ and $x'' = (x''_1, x''_2, \ldots, x''_n)$, in the two sets above, respectively, such that $|x''_i - x'_i| = 1$.

The complexity of Algorithm 3.2 depends on Test 1, which is a linear programming problem, and the process I adopt from [14]. Both of them have been proved to be polynomial w.r.t. n, T.

3.4 Experiments

I conducted experiments to show the better performance of the above two algorithms than that of algorithms based on the simulatable auditing model.

For MAX query, Algorithm 3.1: The size of the database, n, is selected from 1000 to 2000. Each variable, as an integer, is randomly selected from $[1..n]$ with uniform distribution. The sequence of queries $q_1, q_2, \ldots$ is also randomly selected from all possible max queries such that the corresponding set $|Q_i| \le \frac{n}{10}$. In Figure 3.8, I show the large performance improvement of Algorithm 3.1, compared to the reference algorithm based on the simulatable auditing model in an average of 50 tests (performance of the unsafe A_{simple} is also listed as reference). In Figure 3.8,

- Plot (A) shows the number of answered queries before the first denial.
- Plot (B) shows the number of answered queries for total 200 posed queries.
- Plot (C) shows the number of answered queries when different number of queries are posed (from 50 to 250), in a data set with the fixed size 1000.

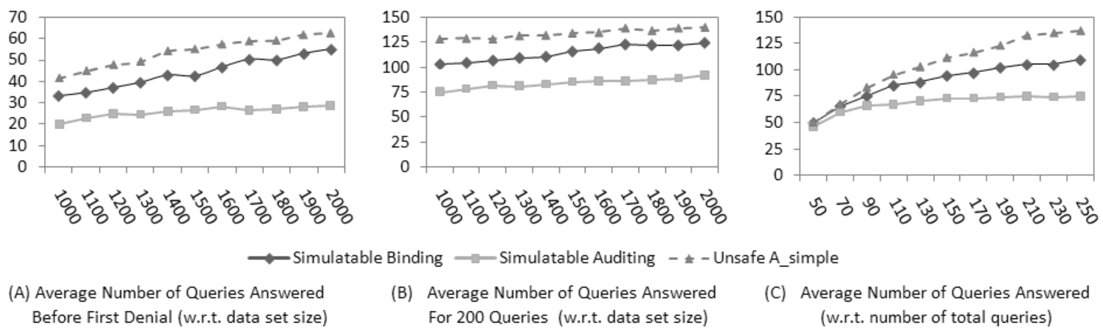


Figure 3.8: Performance of Algorithm 3.1 (Max Query Auditing)

In the above experiments, more than 50% sacrificed data utility (unnecessarily denied queries) by the algorithm based on the simulatable auditing model is regained by Algorithm 3.1 based on the simulatable binding model.

For SUM query, Algorithm 3.2: The size of the database, n, is selected from 1000 to 2000. Each real number variable is randomly selected from $[0, 10]$ with uniform distribution. The database constraint is a set $\{0 \leq x_i \leq 10, (1 \leq i \leq n)\}$. The sequence of queries $q_1, q_2, \dots, q_{n-1}$ is randomly selected from all possible sum queries such that all these $n-1$ queries are independent and can be answered by the algorithm in [14], without considering the existence of the constraints.

In this case, because an algorithm based on the simulatable auditing model cannot even answer a single query, Figure 3.9 shows that the number of answered queries by Algorithm 3.2 based on the simulatable binding model, is very close to the upper bound determined by the original A_{simple}, which is safe in no-constraint cases, but not safe in this case.

One may argue that I should compare Algorithm 3.2 with the algorithm proposed in [19], based on their compromised privacy definition: $(\lambda, \delta, \alpha, T)$-Private. However, I claim that:

(1) The utilities of all these algorithms are bound by the utility provided by the unsafe A_{simple}.

(2) More importantly, the algorithm based on the aforementioned probabilistic private definition, is not a safe algorithm. It is not able to guarantee the safety of all the possible database states, and therefore not comparable to the safe algorithm with data utility.

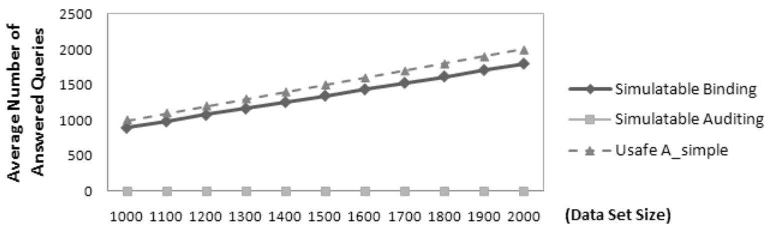


Figure 3.9: Performance of Algorithm 3.2 (Sum Query Auditing)

3.5 Summary

I address the fundamental issue in an online auditing problem that the decision on how to reply to a posed query may leak information about the true database state. The newly proposed *simulatable auditing* model can get around the problem, but does have a huge data utility loss when applied. I suggest that it would be much better to control this information leakage instead of totally denying it.

I propose a new model, called *simulatable binding*, which controls the information leakage and is proved to provide a not only sufficient but also necessary condition to guarantee the database privacy. Two practical *simulatable binding* algorithms are also given for max query and sum query, respectively. Related experimental results are provided to show the regaining, by the proposed algorithms, of great and unnecessary utility loss by previous models. As future work, I believe that the simulatable binding model can be applied to many other online auditing problems, including more sophisticated queries, than those I have discussed in this chapter.

Chapter 4: Securing the Micro-Data Disclosure Process

4.1 Introduction

The problem of information disclosure has drawn much attention in recent years. To support sharing information on a large scale, we need to ensure privacy while providing as much data utility to users as possible. Typically, privacy requirements are expressed by a formal *safety* condition. For example, the property l-diversity (e.g., [40]) is used to ensure privacy in the micro-data disclosure.

There has been much work on maximizing data utility subject to the satisfaction of a safety property. A typical approach is to enumerate a sequence of data generalizations $\mathcal{T} = (T_1, T_2, \ldots, T_n)$ in a non-increasing order of data utility. The first data generalization T_i in the sequence that satisfies the desired *safety* property is disclosed. Because the first such generalization is disclosed, intuitively, maximal data utility is achieved, while the *safety* property is satisfied at the same time. This enumeration approach is used in many algorithms for microdata disclosure, including [36–38, 40].

Unfortunately, this approach does not take into account the fact that users may know not only the disclosed data and the safety property it satisfies, but also the disclosure algorithm used to produce the disclosed data. In conjunction with this knowledge, I claim that the adversary can "break" the safety property.

To illustrate the problem, consider an example of microdata disclosure in Table 4.1 which contains medical records of six patients. The first three columns of the table are assumed to be public knowledge. To protect patient privacy, suppose that we do not wish that a condition of any patient can be determined from the disclosed data and the public knowledge. Certainly, removing all the information of the *Name* attribute is not enough. If the attributes *Age*, *Sex* and *Condition* are released, the patient who has a particular

Table 4.1: A Patient Table

Name	**Age**	**Sex**	**Condition**
Alan	Old	M	Heart Disease
Bob	Old	M	Viral Infection
Clark	Middle	M	Cancer
Diana	Middle	F	Cancer
Ellen	Young	F	Flu
Fen	Young	F	Ulcer

Age	**Sex**	**Cond**
O	M	HD
O	M	VI
M	M	Ca
M	F	Ca
Y	F	Fl
Y	F	Ul

(a)

Age	**Sex**	**Cond**
O	$*$	VI
O	$*$	HD
M	$*$	Ca
M	$*$	Ca
Y	$*$	Fl
Y	$*$	Ul

(b)

Age	**Sex**	**Cond**
$*$	M	Ca
$*$	M	VI
$*$	M	HD
$*$	F	Fl
$*$	F	Ul
$*$	F	Ca

(c)

Age	**Sex**	**Cond**
$*$	$*$	VI
$*$	$*$	Ca
$*$	$*$	HD
$*$	$*$	Fl
$*$	$*$	Ul
$*$	$*$	Ca

(d)

Figure 4.1: Sequence of Disclosure Tables

condition can be still determined from the attributes *Sex* and *Condition*, which are so-called quasi-identifiers [31]. For example, from the fact that there is only one patient who is *Middle Age* and also *Male*, Clark's medical condition can be determined.

Assume that we would like to release the patient table over (possibly generalized) quasi-identifier and the sensitive attribute *Condition*, and guarantee the *safety* property of *entropy 2-diversity* [40], which in this case means, intuitively, that each patient has at least two equally likely choices of sensitive attribute values. Assume also that we traverse the generalization lattice[1] for the quasi-identifiers in this order: (Age, Sex), $(Age, *)$, $(*, Sex)$ and $(*, *)$, in which case we generate corresponding disclosure tables, as shown in Figure

[1]A generalization lattice consists of all the possible generalizations of quasi-identifiers and a partial order defined by the generalization function.

4.1.

The first table in the sequence that satisfies Entropy 2-diversity is table (c). (In fact, table (c) even satisfies entropy 3-diversity, more than the required 2-diversity.) Therefore, a standard algorithm will disclose table (c). However, with the knowledge of the selection algorithm and the generalization sequence, an adversary can infer that both *Clark* and *Diana* have *Cancer* from the disclosed table, which is a violation of their privacy.

Indeed, an adversary can reason as follows: From the public knowledge, the adversary knows that *Alan* and *Bob* are of *old* age, *Clark* and *Diana* are of *middle* age, and *Ellen* and *Fen* are of *young* age. Furthermore, the adversary knows that the generalization (*Age*, ***), (corresponding to table (b)) was not selected because it does not satisfy entropy 2-diversity. Therefore, at least one pair (*Alan* and *Bob*), (*Clark* and *Diana*), or (*Ellen* and *Fen*) must have the same condition. Finally, from the disclosed table (c), the adversary knows that the first 3 rows correspond to Alan, Bob and Clark (who are males), and the last 3 rows correspond to Diana, Ellen and Fen (who are females). Therefore, *Alan* and *Bob* cannot have the same condition, nor can *Ellen* and *Fen*. The only remaining possibility is that *Clark* and *Diana* have the same condition. Since *Clark* and *Diana* belong to two separate groups in table (c), the only common condition they may have is *Cancer*. This violates their privacy (i.e., entropy 2-diversity).

The above example illustrates that the traditional method of selecting the first generalization in the sequence that satisfies the safety property cannot guarantee this safety property when an adversary knows the disclosure algorithm and the generalization sequence. I believe that it is unrealistic to assume otherwise in most cases. This situation is similar in the cryptography area where encryption algorithms are typically assumed to be known by the adversary.

This chapter is concerned with the problem of maximizing data utility subject to satisfaction of a safety property. To the best of my knowledge, this is the first time a model and algorithms that guarantee safety, under the more realistic assumption that the adversary knows the disclosure algorithm and the generalization sequence, have been proposed.

More specifically, the contributions of this chapter are as follows. First, I model disclosed information under the assumption that the adversary has knowledge of what I call a *deterministic disclosure function* (DDF), which is a formal notion of a function defined by a disclosure algorithm. This is done by introducing a formal definition of a *disclosure set*. Intuitively, all the adversary can infer from the disclosed data and the knowledge of the DDF is that a true database state is one of the database states in the *disclosure set*.

Second, given a *safety* predicate p, I define the notion of *p-safety* for a DDF. Intuitively, it means that for any true database state, the DDF returns an answer, such that the *disclosure set* inferred by the adversary (not just the disclosed answer!) satisfies the safety predicate p. I also define the notion of *p-optimality* for a DDF. Intuitively, it means that, in addition to *p-safety*, there does not exist a *locally better* DDF, in terms of data utility, that is also *p-safe*.

Third I prove that *p-optimal* DDF is computable, although the problem of deciding whether a DDF is *p-optimal* is NP-hard. I then introduce two specific conditions under each of which I prove that the problem of whether a DDF is *p-optimal* is P-time in the size of the set of all possible database states and the size of the generalization sequence. I do this by developing polynomial algorithms to compute a *p-optimal* DDF. Note, however, that the size of all possible database states may be exponential, in the worst case, in the size of a single database state. Clearly, computing *p-optimal* DDFs in such a general setting would not be practical beyond a restricted number of considered database states.

Fourth, I turn to developing a disclosure algorithm for a specific setting of microdata disclosure. For this case, the *locally better* relation on DDFs in terms of data utility is provided by a sequence of quasi-identifier generalizations. I develop a *p-safe* algorithm that is *weakly p-optimal*, defined formally in this chapter, and polynomial in the size of the original table and the generalization lattice.

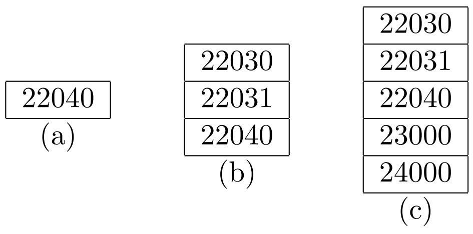


Figure 4.2: Instance Sets for Example 1

4.2 Modeling the Problem

I denote by x the true database state. As an example, x could hold a single value, a vector of values, or a relational database instance. Given a database type, I denote by $\mathbb{D}$ the domain of x; i.e., the set of all possible database states. Furthermore, I denote by D a sub-domain of $\mathbb{D}$ which is the set of states in $\mathbb{D}$ that satisfy given database constraints.

Consider the following *Example 1:* Let the true database state $x = 22040$, i.e., it consists of a single ZIP code. The domain $\mathbb{D}$ is the set of all 5-digit integers. Assume D ={22030, 22031, 22040, 23000, 24000} is a sub-domain that represents the database constraint that each ZIP code must belong to D.

When we do not want to disclose the true database state x, the system can instead provide the user with a set s of database states that contains x. To formalize this, I next introduce the notion of a *disclosure schema*, which defines, for a given true database state x, the set s to be returned to the user.

Definition 5. *A* disclosure schema T *over* $\mathbb{D}$ *is a partition of* $\mathbb{D}$*; i.e.,* $T = \{s_1, s_2, \ldots, s_{n_T}\}$*, where* $\bigcup_{i=1}^{n_T} s_i = \mathbb{D}$ *and* $s_i \cap s_j = \phi \; \forall 1 \leq i < j \leq n$*.*

Intuitively, given a true database state x and a *disclosure schema* T, the "returned" partial information will be a set s such that $x \in s \wedge s \in T$.

In *Example 1:* consider a possible disclosure schema T_i, $1 \leq i \leq 6$, that partitions the sets of all 5-digit zip codes into subsets such that each subset has zip codes that have the same first $6 - i$ digits. Intuitively, the disclosure schema T_i discloses the first $6 - i$ digits of the true database state x.

Many possible *disclosure schemas* can be used to disclose partial information on the true database state to preserve privacy. However, it is important to provide maximal data utility, i.e., disclose as much (”precise”) information on the true database state as possible.

For simplicity, I assume that we are provided with a *candidate disclosure schema sequence* $\mathcal{T} = (T_1, T_2, \ldots, T_n), T_i = \{s^i_1, s^i_2, \ldots, s^i_{n_i}\}, 1 \le i \le n$ where $T_1, \ldots, T_n$ are disclosure schemas, which appear in nonincreasing order of data utility. I always assume that the last disclosure schema T_n is $\{\mathbb{D}\}$. Here $\mathbb{D}$ is the only element of T_n, which, beyond what the user knows from the database constraints, gives the system the choice to disclose nothing, and thus always be able to satisfy the required safety property.

To represent the nonincreasing order of data utility in the candidate disclosure schema sequence $\mathcal{T}$, I write $T_i < T_j$ to denote that T_i appears earlier than T_j in $\mathcal{T}$ (i.e., $i < j$).

Also, by a slight abuse of notation, I will sometimes refer to $\mathcal{T}$ as a set of all elements in the sequence $\mathcal{T}$. Given a specific problem, I call the correlated triple $(D, \mathbb{D}, \mathcal{T})$ the *problem setting.*

In *Example 1:* one possible candidate schema sequence is $\mathcal{T} = (T_1, T_2, \ldots, T_6)$ where the disclosure schema T_i, $1 \le i \le 6$, discloses the first $6 - i$ digits of the true database state x.

Given a specific disclosure schema T_i and the assumption that the user already has the knowledge that $x \in D$, the following definition formalizes the notion of a set of database states that is actually returned.

Definition 6. *Given a database with a domain $\mathbb{D}$, a sub-domain D, and a candidate disclosure schema $\mathcal{T} = (T_1,\ T_2,\ \ldots,\ T_n)$ over $\mathbb{D}$, an* instance set function *t of a true database state x is a mapping $t : \mathcal{T} \times D \to 2^D$ defined by:*

$$t(T_i, x) = D \cap s^i_j$$

such that $s^i_j \in T_i$ and $x \in s^i_j$. I call $t(T_i, x)$ an instance set.

For *Example 1:* given the true database state $x = 22040$ and the candidate schema

```
Input: $D, \mathbb{D}, \mathcal{T}, x$;
Var:
    i:Integer;
Begin
    For $i = 0$ to $n - 1$ Do
        If $|t(T_k, x)| > 1$
            Return $T_k$;
        End If
    End For
    Return $T_n$;
End
```

Figure 4.3: Algorithm 4.1

Table 4.2: The DDF A_1

$d \in D$	$A_1(d)$
22030	T_2
22031	T_2
22040	T_3
23000	T_5
24000	T_5

sequence $\mathcal{T} = (T_1, T_2, \ldots, T_6)$ such that T_i discloses the first $6 - i$ digits of the true database state x, the schema T_1 and T_2, T_3 and T_4, and T_5 and T_6 generate the instance sets shown in Figures 4.2(a), 4.2(b), and 4.2(c), respectively.

Intuitively, given a problem setting $(D, \mathbb{D}, \mathcal{T})$ and a true database state x in D, an algorithm of data disclosure will output one disclosure schema in $\mathcal{T}$. Such an algorithm defines a *deterministic disclosure function* (DDF).

Definition 7. *A* deterministic disclosure function *(DDF) A over $\mathcal{T}$ is a mapping $A : D \to \mathcal{T}$.*

For *Example 1:* consider a variant of the "traditional" disclosure algorithm given in Figure 4.3. *Algorithm* 4.1 defines a DDF A_1 shown in Table 4.2. For example, if the true database state is $x = 22040$, *Algorithm* 4.1 will return T_3.

Intuitively, the schema T_3 carries the information that the first three digits of x is 220. Note that the instance set function t gives an instance set associated with a database state x. In Example 1, the instance set associated with $x = 22040$ is $t(T_3, 22040) =$

$\{22030, 22031, 22040\}$.

If an adversary did not know the algorithm and the disclosure schema sequence it uses, the information in the instance set would be all the adversary could infer from the disclosed data. The knowledge of the algorithm, however, provides an adversary additional information that can be used to further restrict the set of choices in the instance set. More specifically, a DDF A over $\mathcal{T}$ returns T_i as output, i.e., $A(x) = T_i$, the adversary will know that the true database state x must be in $A^{-1}(T_i)$. This consideration gives rise to the notion of the *disclosure set*.

Definition 8. *The* disclosure set *DS of a database state x by a DDF A is a function $DS : \mathcal{A} \times D \to 2^D$, where $\mathcal{A}$ denotes the set of all DDFs over $\mathcal{T}$, defined by:*

$$DS(A, x) = t(A(x), x) \cap A^{-1}(A(x))$$

Intuitively, the disclosure set $DS(A, x)$ is all the adversary can infer about the true database state. That is, adversary knows that (1) $x \in DS(A, x)$ and (2) $DS(A, x)$ is the minimal set that satisfies (1) (i.e., the adversary cannot infer that x is in any proper subset of $DS(A, x)$).

For *Example 1:* as described earlier, the adversary already knows that $x \in \{22030, 22031, 22040\}$. Moreover, because T_3 is returned, he or she also knows that $x \in A_1^{-1}(T_3) = \{22040\}$. Thus the adversary can conclude that x is in $\{22030, 22031, 22040\} \cap \{22040\} = \{22040\}$. Thus, the adversary can precisely determine the true database state x. Note that the intersection above is exactly the disclosure set $DS(A_1, x)$:

$$DS(A_1, 22040) = t(T_3, 22040) \cap A_1^{-1}(T_3) = \{22040\}$$

I assume that there is a *safety predicate* $p : 2^{\mathbb{D}} \to \{true, false\}$. Intuitively, to satisfy a safety property in the information disclosure problem means to satisfy the predicate p on the disclosure set of a database state x.

In Example 1, the safety predicate $p(D') =$ true if $|D'| > 1$. For this p, if a set satisfies the safety predicate, so does any superset. In this case, I say that the safety predicate p is *set-monotonic*:

Definition 9. *A safety predicate p is said to be* set-monotonic *if*

$$\forall D' \subseteq D'', p(D') \Longrightarrow p(D'')$$

Not all safety properties are set-monotonic. For example, as I discuss later in Section 4.4, the entropy l-diversity property is not.

Definition 10. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$ and a safety predicate p, a DDF A is said to be p*-safe *if*

$$\forall x \in D, p(DS(A, x)) = true$$

A deterministic algorithm is said to be p-safe *if the DDF it defines is p-safe.*

Note that algorithm 4.1, which is a representative of traditional disclosure algorithms, is not p-safe for $P(D') : |D'| > 1$. This is exactly the problem I intuitively described in the patient information example in Section 4.1.

Since we are interested in maximizing data utility, we would like to choose a disclosure schema that appears as early as possible in the candidate schema sequence.

Definition 11. *Given two DDFs A and A', I say that A' is* locally better *than A, denoted as $A' \prec A$, if:*

$$\begin{aligned}&\forall x \in D, (A'(x) < A(x) \vee A'(x) = A(x))\\ &\textit{and } \exists x \in D, A'(x) < A(x)\end{aligned}$$

Definition 12. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$, a DDF A is said to be p*-optimal *if:*

$$\begin{aligned}&A \textit{ is } p\textit{-safe, and}\\ &\forall A' \prec A,\ A' \textit{ is not } p\textit{-safe}\end{aligned}$$

*A deterministic algorithm is said to be p-*optimal *if the DDF it defines is p-optimal.*

In the following section, I study the problem of finding a p-optimal DDF in terms of its computability and complexity.

4.3 Computing p-optimal DDFs

For a given problem setting $(D, \mathbb{D}, \mathcal{T})$ and a safety predicate p, I assume that $p(D) = \mathit{true}$. Note that if $p(D)$ were not *true*, the safety predicate p would not be satisfied even if nothing is disclosed. Since the last disclosure schema $T_n \in \mathcal{T}$ is $\{D\}$, the DDF that returns T_n for every input x is p-safe.

Theorem 8. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$, and a safety predicate p such that $p(D) = \mathit{true}$, a p-optimal DDF exists and is computable.*

Proof. To see this, note that the restricted sub-domain D of database states is finite. Therefore, there are only finitely many DDFs, and hence only finitely many p-safe DDFs. Furthermore, as observed earlier, at least one p-safe DDF exists. To compute a p-optimal DDF, enumerate the set of all DDFs, then create the set of all p-safe DDFs by testing the p-safety of each DDF. Then, complete the *locally better* partial ordering on the set of p-safe DDFs into a total ordering. Finally, return the first DDF in the total ordering, which is guaranteed to be p-optimal. □

Clearly, the naive algorithm in Theorem 8 is exponential in the size of D, which, in turn, is exponential in the size of a single database state. Moreover,

Theorem 9. *The following decision problem, referred to as the p-*optimality *problem, is NP-Hard: given a problem setting $(D, \mathbb{D}, \mathcal{T})$, a safety predicate p such that $p(D) = \mathit{true}$, determine whether a given DDF is p-optimal.*

Proof. I use the following decision problem, known to be NP-complete, referred to as the Partition Decision Problem: given a set of numbers $\mathcal{N} = \{d_1, d_2, \ldots, d_m\}$, to decide whether there exists a set $\mathcal{N}_1 \subset \mathcal{N}$ that:

$$\sum_{d \in \mathcal{N}_1} d = \sum_{d \notin \mathcal{N}_1} d$$

is NP-Complete.

Given a Partition Decision Problem instance I reduce it to the p-optimality problem by constructing, in polynomial time, the following problem instance:

- Construct $D = \mathbb{D} = \mathcal{N}$;
- Construct $\mathcal{T} = (T_1, T_2)(m = 2)$, where:
  - $T_1 = \{\{d_1\}, D \setminus \{d_1\}\}$;
  - $T_2 = \{D\}$;
- Compute $\lambda = \sum_{d \in D}$;
- Construct p such that $p(D')$ is true if $\sum_{d \in D'} d \geq \lambda/2$
- Define a DDF A as one that returns T_2 for every $d \in D$
- The problem is to determine whether A is p-optimal.

It is easy to see that the answer to the input Partition Decision Problem is "yes" if the answer to the constructed p-optimality problem is "no." It is also clear that the construction of the p-optimality problem takes polynomial time. $\square$

While the general p-optimality problem is NP-hard, it turns out the complexity can be improved under certain assumptions on the safety predicate and the candidate schema sequence. I have defined the *set-monotonic* safety predicate in Section 2. Here is a similar property of the candidate schema sequence.

Definition 13. *I say that a candidate schema sequence $\mathcal{T}$ is* set-monotonic *if* $\forall x \in \mathbb{D}, \forall 1 \leq i < j \leq n, t(T_i, x) \subseteq t(T_j, x)$.

Note that in the proof of NP-Hardness, I reduce the Partition Decision Problem into a p-optimal Decision problem that has a set-monotonic safety predicate and a set-monotonic candidate schema sequence. These two properties by themselves do not reduce the complexity of the problem much. Instead, they provide a starting point to simpler situations.

4.3.1 Backward Traversal Algorithm

First, I try an approach that traverses the candidate schema sequence backward, i.e., starting with the last T_n with the least data utility (nothing is disclosed) in the order of non-increasing data utility. Suppose both the candidate schema and the safety predicate are set-monotonic. Intuitively, the source of difficulty is in the computation that splits the set D into two sets such that both satisfy the safety predicate p.

Definition 14. *A safety predicate p is said to be* easy-split *if for any set D and D' such that $p(D) = true$, the following two computations can be done in polynomial time with respect to $|D|$:*

- *Whether there exists $D'' \subseteq D' \cap D$ that $p(D'') = p(D \setminus D'') = true$;*
- *Generate a set $D^* \subseteq D \cap D'$, called at-large-split of (D, D'), where:*

  (1) $p(D^*) = p(D \setminus D^*) = true$;

  (2) $\forall D'' \subseteq D \cap D', D^* \setminus D''$, (1) does not hold.

Note that the instance of the p-optimality problem constructed in the NP-hardness proof provides an example of p that is *set-monotonic*, but not *easy-split*. Thus, being *set-monotonic* does not imply being *easy-split*.

Also, being *easy-split* does not imply being *set-monotonic*, as the following example demonstrates. Let $p(D)$ be true for a finite set D of integers if $\sum_{d \in D} d \mod 3 = 0$. Clearly, p is *easy-split* but not *set-monotonic*.

Under the assumption that a safety predicate p is *easy-split* and both p and $\mathcal{T}$ are *set-monotonic*, Algorithm 4.2 in Figure 4.4 computes a p-optimal DDF in polynomial time.

```
01.Input: $\mathcal{T}$,$D$,$p$,$n$;
02.Output: $t$; //output DDF, represented by an array on D
03.Var: $D'$,$D''$,$D^*$; //subset of D
04.Begin
05.   $t[D] \leftarrow T_n$;
06.   $D'' \leftarrow \phi$;
07.   For $i = n-1$ to 1 Step $-1$ do
08.      $D' \leftarrow D \setminus D''$;
09.      For $j = 1$ to $n_i$ do
10.         Compute at-large-split $D^*$ of $(D', s^i_j)$;
11.         $t[D^*] \leftarrow T_i$;
12.         $D' \leftarrow D' \setminus D^*$;
13.      End For
14.      $D'' \leftarrow D'' \cup D'$;
15.   End For
16.   Return $t[D]$;
17.End
```

Figure 4.4: Algorithm 4.2

Table 4.3: The DDF A'_1

$d \in D$	$A'_1(d)$
22030	T_3
22031	T_3
22040	T_3
23000	T_5
24000	T_5

Theorem 10. *For a problem setting $(D, \mathbb{D}, \mathcal{T})$ and a safety predicate p such that both $\mathcal{T}$ and p are set-monotonic and p is easy-split, Algorithm 4.2 returns a p-optimal DDF in polynomial time in $|D| = m$ and $|\mathcal{T}| = n$.*

This theorem follows from the algorithm. Table 4.3 shows the DDF A' computed by *Algorithm* 4.2 when applied to the problem setting of the previous *Example 1*, which I discussed in Section 2. Note that in Example 1 the candidate schema sequence $\mathcal{T}$ is *set-monotonic*, and the safety predicate $p(D') = (|D'| > 1)$ is *set-monotonic* and *easy-split.*

Note that, A'_1 binds the output disclosure schema of 22030 and 22031 with 22040 and therefore protects 22040 from being disclosed. This is not the case for the DDF A_1 in

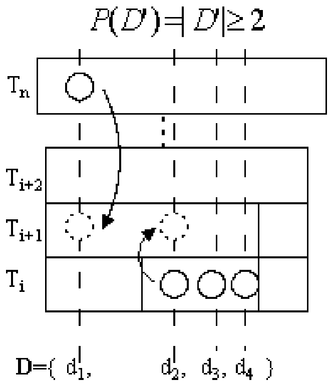


Figure 4.5: A none T_n-safe example

Table 2 (Section 2) computed by a "traditional" disclosure algorithm.

4.3.2 Forward Traversal Algorithms

Note that, in many cases the assumption that $\mathcal{T}$ is set-monotonic is not satisfied. Unfortunately, Algorithm 4.2, which is based on backward traversal of $\mathcal{T}$, would not necessarily compute a p-optimal DDF if we relax this assumption on $\mathcal{T}$. I now develop another algorithm, which is based on forward traversal of $\mathcal{T}$ and for which we are able to relax the assumption on $\mathcal{T}$.

It is easier to check the p-safety property under the assumption that p is set-monotonic, when $\mathcal{T}$ is forward traversed. However, as the following example shows, if a disclosure algorithm does only one forward traversal of $\mathcal{T}$, we cannot always guarantee p-safety.

Figure 4.5 gives an example of this situation. Assume that the safety predicate p is true for D' if $|D'| \geq 2$. The diagram describes a DDF A based on the one-time forward traversal. The leftmost column indicates that for the database state d_1, the DDF returns T_n (indicated with a solid circle). Similarly, the DDF returns T_i for d_2, d_3, and d_4. Clearly, the disclosure set $DS(A, d_1) = \{d_1\}$ violates the safety predicate p. Therefore, DDF A is not p-safe. Interestingly, in the case of database state d_1, no information is explicitly returned to the user. Yet, the adversary can still infer that the true database state is d_1 from just the knowledge of the algorithm.

To deal with this, I can modify the DDF as follows. I select a subset $D' = \{d_2, d_3, d_4\}$ of D, take the subset $D'' = D' \cup \{d_1\}$, and then split them into two sets (D', D'') using the notion of *at-large-split*, where $D'' = \{d_3, d_4\}, D' = \{d_1, d_2\}$. For d_3, d_4, the new DDF will return D'' the same as the original DDF. For d_1, d_2, the new DDF will return T_{i+1}, which is the earliest disclosure schema in which both d_1 and d_2 are in the same partition. Algorithm 4.3 given in Figure 4.6 follows this intuition.

Theorem 11. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$, a safety predicate p that is p set-monotonic and easy-split, Algorithm* 4.3 *returns a p-optimal DDF in polynomial time in $|D| = m$ and $|\mathcal{T}| = n$.*

sketch. From the algorithm, observe that $\forall d \in D$:

$$\exists D' \subseteq D, p(D') \wedge d \in D' \wedge (\exists T_i, s^i_j \in T_i, t[d] = T_i \wedge D' \subseteq s^i_j)$$

Also, because p is set-monotonic, we have

$$\forall d \in D, p(DS(t, d)) = true$$

Thus, $t[D]$ is p-safe. For p-optimality, observe from *Algorithm* 4.3 the following property for any $t'[D]$ such that $t'[D] \prec t[D]$: if $t'[D]$ is p-safe, then the *Algorithm* 4.3 returns $t'[D]$ instead of $t[D]$. Therefore, $t[D]$ is p-optimal.

Finally, it is easy to see from Algorithm 4.3 that its complexity is $O(mn)$, which completes the proof. □

To illustrate Algorithm 4.3, I use the previous *Example 1* again. Note that this example satisfies the conditions of *Algorithm* 4.3. The output DDF A''_1 is shown in Table 4.4.

Note that a p-optimal DDF is not unique. Both A'_1 generated by Algorithm 4.2 and A''_1 generated by Algorithm 4.3 are p-optimal DDFs.

The situation illustrated in Figure 4.5 would not arise if the following property is satisfied, which can be used to further simplify *Algorithm* 4.3.

Table 4.4: The DDF A''_1

$d \in D$	$A''_1(d)$
22030	T_2
22031	T_2
22040	T_5
23000	T_5
24000	T_5

Definition 15. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$, and a safety predicate p that is set-monotonic, I say that p is T_n-safe, if the following condition holds: there exists $D' \subseteq D$ such that*

$$\forall d' \in D', \forall T_i \in \mathcal{T} \setminus \{T_n\}, \forall s^i_j \in T_i,$$

$$d' \in s^i_j \Rightarrow p(s^i_j) = false;$$

$$p(D') = true.$$

I say that p is T_n-safe for $\mathcal{T}$.

In other words, there exists a set D' that satisfies p and such that any p-safe DDF returns T_n for every element in D'.

The property of p being T_n-safe is used in *Algorithm* 4.4 (Figure 4.7), which simplifies *Algorithm* 4.3.

Theorem 12. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$ and a safety predicate p that is set-monotonic and T_n-safe, Algorithm* 4.4 *returns a p-optimal DDF in polynomial time in $|D| = m$ and $|\mathcal{T}| = n$.*

This theorem follows directly from the previous theorem and the T_n-safety property. Note that here I no longer require the easy-split property for p.

So far I have discussed several specified simple cases to avoid solving an NP-hard problem. Each of Algorithms 4.2, 4.3, and 4.4 computes a p-optimal DDF under certain assumptions. Note that the property of p being *set-monotonic* is common in all three. This property plays a central role in reducing complexity. The typical "counting"-based safety predicates would always satisfy the *set-monotonic* property.

4.3.3 Designing p-optimal Algorithms

By definition, a p-optimal algorithm defines a p-optimal DDF. When we talk about a p-optimal algorithm, we are more concerned with efficiency. Note that the algorithms presented so far do not define a p-optimal DDF, but rather compute a p-optimal DDF. Whereas, in practice we typically need an algorithm that defines a p-optimal DDF; i.e., an algorithm that given a problem setting, safety predicate p, and a database state, returns a disclosure schema T_i.

A naive p-optimal disclosure algorithm can first compute a p-optimal DDF, e.g., using algorithms given in Section 3, and then use the output DDF data structure to return a disclosure schema for an input database state x. Such a disclosure algorithm, however, will have the complexity of the algorithm to compute p-optimal DDF. A more efficient p-optimal algorithm can be based on the following steps, under the assumption that p is set-monotonic and T_n-safe.

- Traverse $\mathcal{T}$ from T_1 to T_{n-1}:
  - If $p(t(T_1, x))$ return T_1;
  - Otherwise, for every T_i, construct a set $D' \subseteq t(T_i, x)$ such that no database state in D' has output schema prior to T_i;
  - if $p(D')$ then return T_i;
- return T_n (i.e., no prior T_i was returned for x)

Note that an algorithm following the above steps will define a p-optimal DDF under the specified conditions. Also, it has better average complexity than that of the naive algorithm, which is based on computing the entire p-optimal DDF. However, the worst case complexity is still the same as that of the algorithm to compute a p-optimal DDF, which is polynomial in the size of D and $\mathcal{T}$.

In some cases the size of D may be relatively small, in which case even the naive algorithm may be acceptable. However, in the worst case, for the general problem setting

I have considered, the size of D may be exponential in the size of a single database state. I therefore conclude that a more domain-specific problem setting is necessary for better complexity.

4.4 The Case of Micro-Data Disclosure

4.4.1 The Problem Setting $(D, \mathbb{D}, \mathcal{T})$

In a microdata disclosure problem, we are given an original table that contains a set of attributes ID (such as SSN or name), a set of attributes Q that are non-sensitive but serve as quasi-identifiers, and a sensitive attribute S. The attributes ID or Q can be used to identify an individual. (For simplicity of presentation, I assume that S is a single attribute.) Under this setting, a database state x is the association between the ID and the sensitive attribute S (i.e., the projection of the original table on ID and S).

Let N be the number of tuples in the original table. I denote as $x = \{x_1, x_2, \ldots, x_N\}$ where each i, $1 \le i \le N$ corresponds to an ID in the table. Let $B = \{b_1, b_2, \ldots, b_{n_B}\}$ be the set of all possible values of the sensitive attribute. The domain $\mathbb{D}$ of all possible values of the database state x could be denoted as B^N and has the size of n_B^N. The restricted sub-domain D here is equal to $\mathbb{D}$, $D = \{d_1, d_2, ...d_{n_B^N}\}$. For every d in the restricted sub-domain D, I use $d(x_i)$ to denote the sensitive value of x_i given by d.

The quasi-identifiers contained in the original table are used to generalize the disclosure table by grouping people and their sensitive attributes with the same generalized quasi-identifiers. Given a particular generalization, regardless of how it is obtained, I can have a partition on the set of all tuples in the original table, denoted by $G : \{x_1, x_2, \ldots, x_N\} \to \mathbb{N}$. G takes one tuple as input and outputs a group number. For example, $G(x_1) = G(x_2)$ means that for the generalization G, x_1 and x_2 are in the same group. Every generalization G determines a partition on the domain $\mathbb{D}$, defined as follows:

$$T = \{s | \forall d_1, d_2 \in s, \forall x_i, d_1(G^{-1}(G(x_i))) = d_2(G^{-1}(G(x_i)))\}$$

where $d_1(G^{-1}(G(x_i)))$ represents the bag of sensitive attributes of the set $G^{-1}(G(x_i))$.

Traditionally, we traverse a generalization lattice, in which each node is a generalization, based on some utility function [45]. The sequence of traversed nodes defines a sequences of disclosure schemas. The candidate schema sequence $\mathcal{T}$ is defined as follows:

- $(T_1, T_2, \ldots, T_{n-1})$ is the sequence determined by the sequence of the generalizations;
- T_n, the last schema, which discloses nothing.

4.4.2 The Safety Predicate p

There are two important safety requirements used in the micro-data disclosure problem: k-anonymity [46] and l-diversity [40]. Satisfaction of k-anonymity requires that the size of any group determined by the chosen generalization is at least k. There are multiple variants of the l-diversity property [40]. For example, entropy l-diversity means, that the entropy of the bag of all the possible values of any x_i is greater than $log_2 l$. This is under the assumption of uniform probability distribution of values in the bag (i.e., every value has the same probability to be the true value).

Suppose I have a disclosure set DS for a given x, Note that k-anonymity is *not* a property of a disclosure set. The property of entropy l-diversity is equivalent to the following predicate p on the disclosure set DS:

$$Min_i(En_i(DS)) \geq log_2 l$$

where $En_i(DS)$ is the entropy of the bag of all possible sensitive values for x_i under the assumption of uniform probability distribution.

4.4.3 Disclosure Algorithm for Entropy l-Diversity

In Section 4.3, I have shown that the *set-monotonic* property of the safety predicate p serves as a key condition to achieve a P-Time p-optimal algorithm in the size of D.

Unfortunately, the safety predicate p for entropy l-diversity is not *set-monotonic*. To see this, consider the following example of entropy 2-diversity shown in Figure 4.8, in which only sensitive attributes of the generalized groups are listed.

For T_1, observe that the generalization group that corresponds to the bag $\{A, A\}$ of sensitive attributes violates entropy 2-diversity. For T_2 consider the set D' of all possible database states that violate entropy 2-diversity at T_1. Note that, for every element in D', either the values of the sensitive attribute in Tuple 1 and Tuple 2 are both A, or are both B. However, the probability of being A is 3 times the probability of being B. Thus,

$$En_1(D') = En_2(D') \approx 0.81 < \log_2 2$$

Therefore, D' violates entropy 2-diversity. Now, consider the following set $D'' \subset D'$:

$$D'' = \{(A, A, B, B, A, E, C, D), (B, B, A, A, C, D, A, E)\}$$

D'' contains only two elements. It is easy to observe that

$$\forall 1 \leq i \leq 8, En_i(D'') = log_2 2$$

Thus D'' satisfies entropy 2-diversity and therefore T_2 can be returned as output for every $d \in D''$, which shows that the safety predicate p for entropy l-diversity is not *set-monotonic.*

As a result, we may not be able achieve a P-time p-optimal disclosure algorithm for a microdata disclosure problem and entropy l-diversity without having more specialized conditions. Furthermore, even if the safety predicate p for entropy l-diversity is set-monotonic, we still have the problem that the achieved p-safe disclosure algorithm is P-time in the size of D, which is exponential in the worst case in the size of the original table.

In the following, I develop an algorithm that is (1) polynomial in the size of the original table (not the size of $|D|!$), (2) p-safe, but only (3) *weakly p-optimal.* The notion of *conservative p-safety* and *weak p-optimality* is formalized in the following definitions.

Definition 16. *Given a problem setting $(D, \mathbb{D}, \mathcal{T})$, I say that a predicate q is a* conservative approximation *of a safety predicate p if*

$$\forall D' \subseteq \mathbb{D}, q(D) \Rightarrow p(D')$$

I say that a DDF A is conservatively p*-safe w.r.t. a conservative approximation q if*

$$\forall x \in D, q(DS(A, x)) = true$$

I say that a disclosure algorithm is conservatively p-safe *w.r.t. q if it defines a conservatively p-safe DDF.*

Definition 17. *Given a problem setting* $(D, \mathbb{D}, \mathcal{T})$*, I say that a DDF A is* weakly p-optimal *for a conservative approximation q of p if:*

$$\begin{aligned}&A \textit{ is } p\text{-safe}, \textit{ and}\\ &\forall A' \prec A,\ A' \textit{ is not } \text{conservative } p\text{-safe } \textit{for } q\end{aligned}$$

I say that a disclosure algorithm is weakly p-optimal *for a conservative approximation q of p if the DDF it defines is* weakly p-optimal *for q.*

For the problem setting of microdata disclosure, I define a conservative approximation q of entropy l-diversity as follows:

$$Min_i(En_i(DS)) \geq log_2 l$$

and

$$En_i(DS) \geq En_G(i)$$

where $En_G(i)$ denotes the entropy of the bag of sensitive attribute values in the generalized group which contains ID i. I will refer to this predicate as *conservative l-diversity.*

Algorithm 4.5 for microdata and entropy l-diversity is given in Figure 4.9. In it, the term $rs(g)$ denotes the *rotation set* of the generalized group g of the original table.

Given a sequence of sensitive values associated with IDs in a generalized group $\beta = (b_1, b_2, \ldots, b_k)$ (i.e., ID(1) to ID(k) are generalized to one group), the *rotation set* means the following:

$$rs(\beta) = \{\beta' | \beta' = (b_i, \ldots, b_k, b_1, \ldots, b_{i-1}), (1 \leq i \leq k)\}$$

Note that the *rotation set* $rs(g)$ is the smallest set such that for any ID i that is generalized in group g,

$$En_i(rs(g)) \geq En_G(i)$$

In *Algorithm* 4.5, I modified the candidate schema sequence of the original problem setting to a new sequence $\mathcal{T}'$ constructed as follows. First, we traverse the original sequence up to the first node that satisfies l-anonymity. From that node, select an arbitrary path up the generalization lattice to the root node.

Theorem 13. *Given the modified problem setting* $(D, \mathbb{D}, \mathcal{T}')$*, Algorithm 4.5 is* weakly p-optimal*, where p is entropy l-diversity, w.r.t. to* conservative l-diversity*. Furthermore, Algorithm 4.5 is polynomial time in the size of the original table and the size of the generalization lattice.*

sketch. When a generalized group is checked to satisfy entropy l-diversity, the *rotation set* of any possible database state in this generalized group must also satisfy entropy l-diversity. Since the entropy function is monotonic [40], any conjunction of such *rotation sets* must also satisfy entropy l-diversity. Observe from *Algorithm* 4.5, that the predicate p is guaranteed, and so *Algorithm* 4.5 is p-safe.

Now, *Algorithm* 4.5 returns a disclosure schema as early as possible for the *rotation sets* of all possible database states. Thus, for any DDF A' that is locally better than the DDF defined by *Algorithm* 4.5, the property of weak entropy l-diversity can not be satisfied for A'.

Finally, it is easy to see that Algorithm 4.5 is polynomial in the size of the original table and the size of the generalization lattice. □

Table 4.5: Patient Information 2

Name	**Marital**	**Age**	**Sex**	**Condition**
Alan	M	Old	M	Heart Disease
Bob	M	Old	M	Viral Infection
Clark	M	Middle	M	Cancer
Diana	M	Middle	F	Cancer
Ellen	M	Middle	F	Flu
Fen	M	Middle	F	Ulcer
Grace	S	Middle	F	Gastritis
Helen	S	Middle	F	Pneumonia
Ian	S	Young	F	Gastritis
Jolie	S	Young	F	Pneumonia

4.4.4 Application Example of Algorithm 4.5

I now apply *Algorithm* 4.5 to a patient information disclosure example similar to the one discussed in Section 4.1. The original table is shown in Table 4.5.

In this example, we are about to generate a disclosure table that satisfies Entropy 2-diversity. First, we traverse the generalization lattice to find the first node that satisfies 2-anonymity.

The generalization lattice of quasi-identifiers and its traversal is shown in Figure 4.10. We find the first node ($Marital$, Age, $*$), for which the disclosure table satisfies 2-anonymity as shown in Table 4.6.

I then select a new sequence $\mathcal{T}' = (T_1, T_2, T_3, T_4)$ where T_1 represents $(Marital, Age, *)$, T_2 represents $(Marital, *, *)$, T_3 represents $(*, *, *)$ and T_4 represents that nothing is disclosed.

Note that at T_1, which corresponds to the node ($Marital$, $*, *$), the rotation set of the original table contains one possible database state, i.e., the table given in Table 4.7. This table does not satisfy the *entropy l-diversity.*

As a result, we will generate the final disclosure table based on T_2, which corresponds to the generalization node $(Marital, *, *)$ as shown in Table 4.8. This delayed output schema indeed protects the database state shown in Table 4.7.

Table 4.6: Table for node (Marital, Age, *)

Marital	**Age**	**Sex**	**Condition**
M	O	*	Viral Infection
M	O	*	Heart Disease
M	M	*	Cancer
M	M	*	Cancer
M	M	*	Flu
M	M	*	Ulcer
S	M	*	Gastritis
S	M	*	Pneumonia
S	Y	*	Gastritis
S	Y	*	Pneumonia

Table 4.7: Patient Information 2

Name	**Marital**	**Age**	**Sex**	**Condition**
Alan	M	Old	M	Cancer
Bob	M	Old	M	Cancer
Clark	M	Middle	M	Flu
Diana	M	Middle	F	Ulcer
Ellen	M	Middle	F	Viral Infection
Fen	M	Middle	F	Heart Disease
Grace	S	Middle	F	Gastritis
Helen	S	Middle	F	Pneumonia
Grace	S	Young	F	Gastritis
Helen	S	Young	F	Pneumonia

4.5 Summary

To the best of my knowledge, this is the first time that the problem of maximal data utility while guaranteeing safety under the assumption that the adversary may know the disclosure algorithm and sequence has been studied.

Many interesting research questions remain open. One research question is how to extend the "local optimality" property used in this chapter to "global optimality." This includes finding good measures of data utility that would define total (rather than partial) ordering on possible ways to disclose data. Another research direction is using probabilistic, rather than deterministic disclosure algorithms, and the related extended notions of safety. I conjecture that some of the complexity hurdles may be eliminated. For the case of

Table 4.8: Final Disclosure Table

Marital	**Age**	**Sex**	**Condition**
M	*	*	Cancer
M	*	*	Viral Infection
M	*	*	Heart Disease
M	*	*	Ulcer
M	*	*	Cancer
M	*	*	Flu
S	*	*	Pneumonia
S	*	*	Gastritis
S	*	*	Pneumonia
S	*	*	Gastritis

microdata disclosure and l-diversity in particular, a question remains whether I can define conditions less restrictive than the notions of *conservative* p-safety and *weak optimality.* Finally, extending my techniques to more general settings beyond the setting of generalization sequences is an interesting question that may be applicable to statistical databases.

```
01.Input: $\mathcal{T}$,$D$,$p$,$n$;
02.Output: $t$; //output DDF, represented by an array on D
03.Var: $D'$,$D^*$,$C$; //subset of D
04.     $T'$; //element of $\mathcal{T}$
05.Begin
06.   $t[D] \leftarrow T_n$;
07.   $D' \leftarrow D$;
08.   $T' \leftarrow T_n$;
09.   $C \leftarrow D$;
10.   For $i = 1$ to $n$ do //iterate schemas
11.      For $j = 1$ to $n_i$ do //iterate elements in a schema
12.         If $p(D' \cap s^i_j) \neq \phi$ do
13.            $C \leftarrow D' \cap s^i_j$;
14.            $T' \leftarrow T_i$;
15.            $t[C] \leftarrow T_i$;
16.            $D' \leftarrow D' \setminus C$;
17.         End If
18.      End For
19.   End For
20.   If $D' \neq \phi$ do
21.      If $\exists$ at-large-split $D^*$ of $(D' \cup C, C)$
22.         For $i = 1$ to $n$ do
23.            For $j = 1$ to $n_i$ do
24.               If $D' \cup C \setminus D^* \subseteq s^i_j$ do
25.                  $t[D' \cup C \setminus D^*] \leftarrow T_i$;
26.               End If
27.            End For
28.         End For
29.      Else
30.         For $i = 1$ to $n$ do
31.            For $j = 1$ to $n_i$ do
32.               If $D' \cup C \subseteq s^i_j$ do
33.                  $t[D' \cup C] \leftarrow T_i$;
34.               End If
35.            End For
36.         End For
37.      End If
38.   End If
39.   Return $t[D]$;
40.End.
```

Figure 4.6: Algorithm 4.3

```
01.Input: $\mathcal{T}$,$D$,$p$,$n$;
02.Output: $t$; //output DDF, represented by an array on D
03.Var: $D'$,$D^*$,$C$; //subset of D
04.      $T'$; //element of $\mathcal{T}$
05.Begin
06.   $t[D] \leftarrow T_n$;
07.   $D' \leftarrow D$;
08.   $T' \leftarrow T_n$;
09.   $C \leftarrow D$;
10.   For $i = 1$ to $n$ do
11.      For $j = 1$ to $n_i$ do
12.         If $p(D' \cap s^i_j)$ do
13.            $C \leftarrow D' \cap s^i_j$;
14.            $T' \leftarrow T_i$;
15.            $t[C] \leftarrow T_i$;
16.            $D' \leftarrow D' \setminus C$;
17.         End If
18.      End For
19.   End For
20.   Return $t[D]$;
21.End.
```

Figure 4.7: Algorithm 4.4

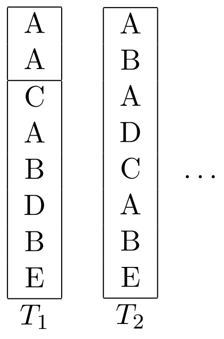


Figure 4.8: An example of Entropy 2-diversity

```
01.Input: $B$; //a set of all sensitive attributes
02.          $\mathcal{G}$; //a generalization lattice
03.          $x$; //a current database state
04.          $l$; //the l of Entropy $l$-diversity
02.Output: $T$; //the output disclosure schema for $x$
03.Var: $D'$,$D^*$,$C$; //subset of D
04.       $T'$; //element of $\mathcal{T}$
05.Begin
06.    Traverse $\mathcal{G}$ using the original sequence $\mathcal{T}$ to
              find $g_1 \in \mathcal{G}$ that satisfies l-anonymity;
07.    Select a new sequence $(g_1, g_2, \ldots, g_{n'})$ from $g$
              to the top point of $\mathcal{G}$;
08.    Let $\mathcal{T}' = (T_1, T_2, \ldots, T_n)(n = n' + 1)$ where
              $(T_1, T_2, \ldots, T_{n-1})$ represents schema sequence based on
              $(g_1, g_2, \ldots, g_{n'})$ and $T_n$ represents that nothing is disclosed
09.    Traverse $\mathcal{T}'$ to find the first $T_i$ such that $t(T_i, x)$ satisfies entropy $l$-diversity;
10.    If $T_i = T_n$ Return $T_n$;
11.    For every $j$ from $n$ down to $i + 1$;
12.        If for every generalized group $g$ in $T_j$, $rs(g)$ does not satisfy Entropy
              $l$-diversity on $T_{j-1}$
13.          Return $T_j$;
14.    End For;
15.    Return $T_i$;
16.End
```

Figure 4.9: Algorithm 4.5

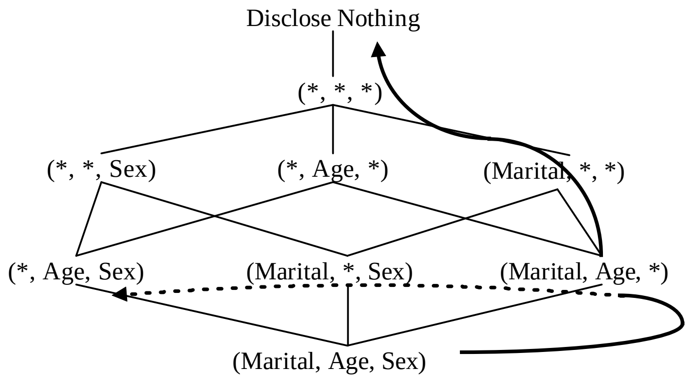


Figure 4.10: Traversing the Generalization Lattice

Chapter 5: Efficient Micro-data Disclosure

5.1 Introduction

In this chapter I study the problem of how to design an efficient and safe heuristic disclosure algorithm for micro-data disclosure with good performances. To start, we recall the linking attack through the following example.

Table 5.1 gives an example of a table containing several patients' medical information. Suppose the relation between the individuals' names, namely, the *identity* attribute and the medical conditions, namely, the *sensitive value* attribute, is considered sensitive. Simply hiding the *identity* (Name) is not sufficient. A tuple's *sensitive value* (Condition) may still be linked to a unique identity in external publicized tables (such as a Voter's List) through common attributes (ZIP, Gender, and DOB in this case), namely, *quasi-identifiers*. Such an inference is usually called the *linking attack* [31]. That is, even if we remove the name attribute from Table 5.1, the identities of patients and hence their medical conditions could still be determined through the unique combination of ZIP, Gender, and DOB. [1]

To prevent the linking attack, such a micro-data table should be further blurred through *generalization*, that is, transforming a *quasi-identifier* into a less precise version (although not considered in this chapter, suppression can also be applied to provide protection against the linking attack [29]). Table 5.2 demonstrates how a micro-data table can be generalized (note that we have simplified the above patient information table into the leftmost table T_0 where we assume each individual can be identified through her unique age). To provide enough protection against the linking attack, table T_0 needs to be generalized to satisfy

[1]Based on the analysis [47], 87% (216 million of 248 million) of the population in the United States had reported characteristics that likely made them unique based only on 5-digit ZIP, gender, and date of birth.

Table 5.1: A Patient's Medical Information Table

Name	**ZIP**	**Gender**	**DOB**	**Condition**
Alice	22030	Female	1949/03/01	flu
Brenda	22030	Female	1959/05/07	tracheitis
Clare	22030	Female	1969/12/01	cancer
Diana	22030	Female	1974/07/09	cancer
Ellen	22030	Female	1975/08/11	pneumonia
Fen	22030	Female	1976/02/23	gastritis

the *k-anonymity* property [31], that is, each individual shares the same generalized quasi-identifier value with at least $k-1$ other individuals in the table. In Table 5.2, $g_1(T_0)$ shows a generalized table that satisfies 2-anonymity.

Table 5.2: A Micro-Data Table with Three Generalizations

Micro-Data Table T_0

Name	**Age**	**Condition**
Alice	60	flu
Brenda	50	tracheitis
Clare	40	cancer
Diana	35	cancer
Ellen	34	pneumonia
Fen	33	gastritis

Generalization $g_1(T_0)$

Age Range	**Condition**
50$\sim$60	flu tracheitis
35$\sim$40	cancer cancer
33$\sim$34	pneumonia gastritis

Generalization $g_2(T_0)$

Age Range	**Condition**
40$\sim$60	flu tracheitis cancer
33$\sim$35	cancer pneumonia gastritis

Generalization $g_3(T_0)$

Age Range	**Condition**
35$\sim$60	flu tracheitis cancer cancer
33$\sim$34	pneumonia gastritis

Nonetheless, k-anonymity by itself is not sufficient, since linking an identity to the second group already reveals the condition of this individual to be cancer. This situation can be avoided by requiring the generalized table to also satisfy the property l-diversity [40]. If generalization $g_2(T_0)$ is released instead of $g_1(T_0)$, then any identity can only be linked to a group of three conditions, among which each is equally likely to be that identity's true condition. In this case, $g_2(T_0)$ satisfies the property of 3-diversity (and at the same time, 3-anonymity).

However, as discussed in the previous chapter, the above statement is true, only if we assume the adversary will never know the fact that $g_1(T_0)$ is first considered (but not disclosed) by the algorithm before $g_2(T_0)$ is disclosed. Otherwise, an adversary can deduce that since $g_1(T_0)$ is not disclosed, the two identities in one of the three groups must both have cancer. Moreover, it must be the second group, because neither the first group nor the third one can be associated with two cancers based on $g_2(T_0)$. That is, both Clare and Diana have cancer. As a result, the adversary gets to know that the real micro-data table must be among the four possible guesses depicted in the upper tabular in Table 5.3, namely, the *disclosure set*.

Table 5.3: Two Disclosure Sets

The Disclosure Set of $g_2(T_0)$ after Considering $g_1(T_0)$

Name	**Age**	**Condition**			
Alice	60	flu	tracheitis	flu	tracheitis
Brenda	50	tracheitis	flu	tracheitis	flu
Clare	40	cancer	cancer	cancer	cancer
Diana	35	cancer	cancer	cancer	cancer
Ellen	34	pneumonia	pneumonia	gastritis	gastritis
Fen	33	gastritis	gastritis	pneumonia	pneumonia

The Disclosure Set of $g_3(T_0)$ after Considering $g_1(T_0)$

Name	**Age**	**Condition**			
Alice	60	flu	tracheitis	cancer	cancer
Brenda	50	tracheitis	flu	cancer	cancer
Clare	40	cancer	cancer	flu	tracheitis
Diana	35	cancer	cancer	tracheitis	flu
Ellen	34	pneumonia	pneumonia	pneumonia	pneumonia
Fen	33	gastritis	gastritis	gastritis	gastritis

Condition			
flu	tracheitis	cancer	cancer
tracheitis	flu	cancer	cancer
cancer	cancer	flu	tracheitis
cancer	cancer	tracheitis	flu
gastritis	gastritis	gastritis	gastritis
pneumonia	pneumonia	pneumonia	pneumonia

To an adversary who obtains the above knowledge, in addition to the disclosed generalized table $g_2(T_0)$, the four guesses in the upper tabular in Table 5.3 are equally likely to be the real micro-data table. Therefore, on the basis of the first row, the adversary's knowledge about Alice's condition can be modeled as a uniform distribution over (flu, tracheitis), namely, the *inherent distribution.* On the other hand, an adversary who does not know about the generalization algorithm will determine Alice's condition based on the generalization $g_2(T_0)$ only, which leads to another uniform distribution $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ over (flu, tracheitis, cancer), namely, the *apparent distribution.*

In most security-related problems, we usually do not consider the applied algorithm as a secret.[2] Therefore, a viable solution to micro-data disclosure using a public algorithm has to enforce privacy properties on the inherent distribution, instead of the apparent distribution. Unfortunately, the computation of inherent distribution is expensive. In fact, it is an NP-hard problem to maximize the data utility of public generalization algorithms while satisfying l-diversity. On the other hand, consider the disclosure set of $g_3(T_0)$ after $g_1(T_0)$ is considered but not disclosed, as shown in the lower tabular in Table 5.3 (note $g_2(T_0)$ is not involved). Interestingly, here every identity's inherent distribution coincides with its apparent distribution.

A question now naturally arises, that is, *can we design algorithms that always produce generalizations whose inherent distribution coincides with apparent distribution?* If the answer is yes, then for such algorithms, which I shall call *coincident* generalization algorithms from now on, I can simply enforce any distribution-based privacy property on the apparent distribution, even though such algorithms are publicly known. In other words, a public algorithm can be as efficient as any secret algorithms, if only it is coincident.

The main contribution of this chapter is two-fold. First, I propose the novel concept of coincident generalization as an efficient solution to micro-data disclosure using public algorithms. I derive necessary and sufficient conditions for any generalization algorithm to be coincident in order to facilitate the design of such algorithms. Second, I instantiate the concept by devising two classes of efficient generalization algorithms, pair-based generalization algorithm and interleaved generalization algorithm, and prove them to be coincident.

[2]This is also called Kerckhoffs' Principle ([48]).

The effectiveness of the proposed algorithms is independent of privacy properties, which makes them applicable to a wide range of applications. My experiments show that those algorithms lead to good data utility and performance as well.

The rest of the chapter is organized as follows. Section 5.2 formally defines coincident generalization. Section 5.3 proposes a family of pair-based coincident generalization algorithms. Section 5.4 proposes a family of interleaved coincident generalization algorithms. I evaluate the proposed algorithms in Section 5.5 through experimental results. We discuss other relevant issues in Section 5.6, and finally conclude the chapter in Section 5.7.

5.2 Coincident Generalization

Section 5.2.1 first discusses knowledge that the adversary can obtain through a public generalization algorithm. Sections 5.2.2 and 5.2.3 then formalize the necessary concepts, and Section 5.2.4 introduces coincident generalization.

5.2.1 Information Disclosure Through Public Generalization Algorithms

In Section 5.1, I have shown that the knowledge obtained from a public generalization algorithm can help the adversary break the privacy protection provided by the typical k-anonymity and l-diversity properties. Now I consider how to formally model such a phenomenon. Typically, given a micro-data table instance as the input, a generalization algorithm will select a particular generalization function among all candidate functions in order to optimize data utility. For example, such a generalization function can be a cut of the taxonomy tree formed by the hierarchy of quasi-identifiers [49]. In selecting the optimal generalization function, a generalization algorithm will typically only look at quasi-identifiers but not sensitive values (for example, the Incognito [38]). Note that generalization algorithms that select generalization functions based on the sensitive values are still under development and are not covered in this chapter.

Therefore, for any input micro-data tables that share the same quasi-identifier attributes, a typical generalization algorithm will go through the same sequence of candidate generalization functions (for example, a series of cuts of the taxonomy tree). Under the worst-case assumption that the generalization algorithm and the quasi-identifiers are both publicly known, such sequences of candidate generalization functions will be also known to the adversary. This knowledge enables the adversary, in combination with the final disclosed result, to determine the sequence of generalization functions checked by the applied generalization algorithms for any micro-data table.

Without loss of generality, I shall formalize the algorithms from the adversary's perception. That is, a generalization algorithm can be represented by a sequence of generalization functions to be checked in order, assuming the input micro-data table is given. For the rest of this chapter, I will abuse the term "generalization algorithms" for both a typical generalization algorithm when discussing the algorithm design, and a generalization algorithm from the adversary's perception when discussing its privacy protection.

5.2.2 Basic Model

I formalize the micro-data disclosure problem as follows (the notations are summarized in Table 5.4):

- Given a micro-data table T_0, an *instance* is any one-to-one relation $T \subseteq \mathbb{I} \times \mathbb{S}$ where $\mathbb{I}$ and $\mathbb{S}$ are the set of *identities* and the multiset of *sensitive values* in T_0, respectively (for the sake of simplicity, I shall omit quasi-identifiers). As typically assumed, both $\mathbb{I}$ and $\mathbb{S}$ are public knowledge as they will appear in any disclosed generalization when suppression is not considered. Let $\mathbb{T}$ be the set of all possible *instances.*

- A *generalization* is a triple $\langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$, where $\mathscr{P}(\mathbb{I})$ is a partition on $\mathbb{I}$, $\mathscr{P}(\mathbb{S})$ is a partition on $\mathbb{S}$, and $\mathcal{R} \subseteq \mathscr{P}(\mathbb{I}) \times \mathscr{P}(\mathbb{S})$ is a one-to-one relation satisfying that $\forall I \in \mathscr{P}(\mathbb{I})\, \forall S \in \mathscr{P}(\mathbb{S}), \langle I, S \rangle \in \mathcal{R} \Rightarrow$
$\mid I \mid = \mid S \mid$. That is, $\mathcal{R}$ relates each *group* of identities to a group of sensitive values of equal cardinality. Let $\mathbb{G}$ be the set of all possible generalizations.

Table 5.4: Notation Table

T_0, T	A given micro-data table, an instance
$\mathbb{I}$, $\mathbb{S}$, $\mathbb{T}$	The set of identities, the multiset of sensitive values, and the set of all possible instances
$\mathscr{P}(\mathbb{I})$, $\mathscr{P}(\mathbb{S})$	A partition on a multiset $\mathbb{I}$, a partition on a multiset $\mathbb{S}$
$g(T) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$	A generalization of T using generalization function $g()$
$G = \langle g_1 g_2 \dots g_n,\ p \rangle$	A generalization algorithm
λ_I	The same order on $\mathbb{I}$ as it appears in T_0
$\mathscr{M}(S)$	The set of all possible permutations on a multiset S
$T(\lambda_I, \lambda)$	An instance formed by two sequences λ_I and λ
ds	Disclosure set

- A *generalization function* $g() : \mathbb{T} \rightarrow \mathbb{G}$, satisfies that if $g(T) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$, then for every $\langle I, S \rangle \in \mathcal{R}$ there must always exist a one-to-one relation $T' \subseteq I \times S$ satisfying $T' \subseteq T$.

- Given a micro-data table T_0, an *adversary's view of an applied generalization algorithm* (or simply a *generalization algorithm*) G is a pair $\langle g_1 g_2 \dots g_n,\ p \rangle$, where $g_1 g_2 \dots g_n$ is a sequence of generalization functions and p is a privacy property. Taking any instance, say, T_0 as the input, G will evaluate each generalization $g_i(T_0)$ for $i = 1, 2, \dots, n$ in the given order. G returns either the first result satisfying p, or ϕ (indicating nothing can be disclosed) if G terminates without p being satisfied.

Table 5.5: An Example of the Notations

$\mathbb{I}$	$g_1(T_0)$		$g_2(T_0)$		$\mathbb{S}$
	$\mathscr{P}_1(\mathbb{I})$	$\mathscr{P}_1(\mathbb{S})$	$\mathscr{P}_2(\mathbb{I})$	$\mathscr{P}_2(\mathbb{S})$	
Alice	I_1^1	S_1^1	I_1^2	S_1^2	flu
Brenda					tracheitis
Clare	I_2^1	S_2^1			cancer
Diana			I_2^2	S_2^2	cancer
Ellen	I_3^1	S_3^1			pneumonia
Fen					gastritis

Example 1. *Table 5.5 rephrases the generalizations $g_1(T_0)$ and $g_2(T_0)$ in Table 5.2 using the above notations.*

5.2.3 Disclosure Set and Distribution of Sensitive Values

As illustrated in Section 5.1, when generalization algorithms are publicly known, I need the concept of *disclosure set* to model what a disclosed generalization may allow an adversary to deduce about the secret micro-data table. Seeing the fact that a public generalization algorithm $G = \langle g_1 g_2 \ldots g_n,\ p\rangle$ returns a generalization $g_i(T_0)$ $(1 \leq i \leq n)$ for some unknown T_0, an adversary's mental image is that any instance $T \in \mathbb{T}$ could have been T_0, unless either $g_i(T) \neq g_i(T_0)$, or, $g_i(T) = g_i(T_0)$ but taking T as the input, G would have returned $g_j(T)$ for some $j < i$.

Example 2. *In the upper tabular of Table 5.3, when $g_2(T_0)$ is returned, any instance T in which both Alice and Brenda have cancer does not appear in the disclosure set, because $g_2(T) \neq g_2(T_0)$. On the other hand, any instance T in which both Brenda and Clare have cancer does not appear either, because the algorithm would have returned $g_1(T)$ instead of $g_2(T)$, even if $g_2(T) = g_2(T_0)$.*

Recall that in Table 5.3, I have adopted the following simplified notation. Instead of enumerating all instances in a disclosure set, I use different permutations on the sensitive values to represent the corresponding instances by assuming an arbitrary but fixed order on identities. More precisely, I need the following notations. Given a micro-data table T_0, I use λ_I for the same fixed order on $\mathbb{I}$ as it appears in T_0. I use $\mathscr{M}(S)$ for the multiset of all possible permutations on any given multiset $S \subseteq \mathbb{S}$. Given a sequence of sensitive values λ of the same length as λ_I, I use $T(\lambda_I, \lambda)$ to denote the one-to-one relation $\{\langle ID, \alpha\rangle : ID$ and α are the j^{th} $(1 \leq j \leq \mid \lambda \mid)$ element of λ_I and λ, respectively $\}$. Using these notations, Definition 18 formalizes the concept of disclosure set.

Definition 18. (Disclosure Set) *Given a micro-data table T_0, a public generalization algorithm G, and a generalization $g(T_0)$ returned by G with T_0 as the input, I call $ds = \{\lambda : \lambda \in \mathscr{M}(\mathbb{S}), G$ returns $g(T_0)$ with $T(\lambda_I, \lambda)$ as the input $\}$ the* disclosure set.

In Section 5.1, I have also shown that knowledge about the generalization algorithm will affect an adversary's belief in terms of the distribution of sensitive values corresponding to each identity. Without the knowledge of the applied generalization algorithm, such distribution is simply based on the disclosed generalization itself. However, if the generalization algorithm is known to the adversary, such distribution must be computed based on the disclosure set. These concepts are formalized in Definition 19.

Definition 19. (Apparent Distribution and Inherent Distribution) *Given a microdata table T_0, a public generalization algorithm that returns $g(T_0) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$ with T_0 as the input, let* ds *be the disclosure set. For any $\langle I, S \rangle \in \mathcal{R}$ and any $ID \in I$, I say*

- *the distribution of distinct elements in the multiset S is the* apparent distribution *of ID, and*
- *the distribution of distinct elements in the multiset $\{\alpha : \langle ID, \alpha \rangle \in T(\lambda_I, \lambda), \lambda \in ds\}$ is the* inherent distribution *of ID.*

Example 3. *Following Example 2, to calcuate Alice's apparent distribution, we need the generalization $g_2(T_0)$ shown in Table 5.2. To calculate Alice's inherent distribution, we need the disclosure set of $g_2(T_0)$ shown in upper tabular in Table 5.3.*

5.2.4 Coincident Generalization

I now formulate the important concept of *coincident generalization.* As illustrated in Section 5.1, the main idea is to design generalization algorithms in such a way that the inherent distribution of every identity under every generalization function always coincides with the corresponding apparent distribution. This concept is formalized in Definition 20.

Definition 20. (Coincident Generalization) *A generalization algorithm G is* coincident*, if when G takes any $T \in \mathbb{T}$ as the input and returns a generalization, the apparent distribution and inherent distribution of every $ID \in \mathbb{I}$ are always identical.*

Example 4. *Following Example 3, we know the generalization algorithm $\langle g_1 g_2, 2\text{-diversity} \rangle$ is not coincident. On the other hand, from the discussion in Section 5.1, we know the algorithm $\langle g_1 g_3, 2\text{-diversity} \rangle$ is coincident.*

Interestingly, there exists an equivalent, but more straightforward, definition for coincident generalization, as stated in Lemma 3. That is, the inherent distributions of all identities within the same group are always identical. The proof of the lemma is based on a simple fact, that is, when the apparent distribution and the inherent distribution are identical, the total number of occurrences of any sensitive value in the disclosure set must remain the same, whether counted horizontally (by identities) or counted vertically (by sequences in the disclosure set).

Lemma 3. *Given any generalization algorithm $G = \langle g_1 g_2 \ldots g_n, p\rangle$, the following two arguments are equivalent:*

(1) G is coincident;

(2) For any instance T_0, any $g_i(T_0) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R}\rangle$ $(1 \leq i \leq n)$ returned by G with the input T_0, any $I \in \mathscr{P}(\mathbb{I})$, and any $ID_1, ID_2 \in I$, the corresponding inherent distribution of ID_1 is always identical to that of ID_2.

Proof: (1)⇒(2) is straightforward. Suppose $\langle I, S\rangle \in \mathcal{R}$, then the inherent distributions of ID_1 and ID_2 are both identical to the same apparent distribution calculated based on S.

For (2)⇒(1), also assume $\langle I, S\rangle \in \mathcal{R}$, and let ds be the disclosure set. For any $\alpha \in S$ and $ID \in I$, use $num(ID, \alpha)$ for the total number of occurrences of α corresponding to ID among all sequences in ds, and use $num'(\alpha)$ for the total number of occurrences of α in S. The following must be true:

$$\frac{num(ID,\alpha)}{\mid ds \mid} = \frac{\mid I \mid \times num(ID,\alpha)}{\mid I \mid \times \mid ds \mid} = \frac{\sum_{j=1}^{|I|} num(ID,\alpha)}{\mid I \mid \times \mid ds \mid} = \frac{num'(\alpha) \times \mid ds \mid}{\mid I \mid \times \mid ds \mid} = \frac{num'(\alpha)}{\mid I \mid}$$

□

Example 5. *In the lower tabular of Table 5.3, Alice, Brenda, Clare, and Diana all have the same inherent distribution. In addition, each subsequence corresponding to these four* *identities in the disclosure set also has the same distribution. The percentage of, say, flu must thus remain the same, whether counted horizontally or vertically:* $\frac{2}{8} = \frac{4\times 2}{4\times 8} = \frac{1\times 8}{4\times 8} = \frac{1}{4}$.

5.3 Pair-Based Coincident Generalization

Section 5.3.1 proposes a family of *pair-based* generalization algorithms and proves them to be coincident. Section 3.2 describes a decomposition of the generalization process. Section 5.3.3 devises two particular algorithms in this family.

5.3.1 A Family of Pair-Based Generalization Algorithms

I now turn to the design of coincident generalization algorithms. We start by observing that in Table 5.2, $g_3(T_0)$ essentially *merges* two equal-size groups in $g_1(T_0)$ into a larger group, and this merging leads to the coincident generalization algorithm described in Example 4. More generally, I propose the family of *pair-based* generalization algorithms in Definition 21.

Definition 21. (Pair-Based Generalization Algorithms) *A pair-based generalization algorithm* $G = \langle g_1 g_2 \ldots g_n, p\rangle$ *satisfies that*

- g_1 *defines groups of the same size;*[3]
- g_{i+1} $(1 \le i < n)$ *is constructed by merging pairs of equal-size groups defined by* g_i.

Next, I prove any pair-based generalization algorithm to be coincident in Theorem 5.3.2. Note that Definition 21 does not specify how pairs of groups should be merged in constructing generalization functions. This fact leaves plenty of possibilities for optimizing this family of generalization algorithms. In Section 5.3.3, I will illustrate such optimization by studying two specific pair-based algorithms.

Theorem 14. *Any pair-based generalization algorithm* $G = \langle g_1 g_2 \ldots g_n, p\rangle$ *is coincident.*

Proof: I prove this theorem in two steps:

[3]To have equal-size groups, it is necessary to allow overlapping between groups in the space of quasi-identifiers, which implies my generalization methods are not full-domain generalizations [31].

1. I first prove the case where g_1 defines groups of size one. Consider any generalization $g_i(T) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$ $(1 \le i \le n)$ and any group $\langle I, S \rangle \in \mathcal{R}$. By Lemma 3, it suffices to show that any two identities $ID_1, ID_2 \in I$ have the same inherent distribution.

   I shall interchangeably refer to an identity in I and its index encoded as a binary string representing a value in $[0, 2^k - 1]$ for some k by assuming $| I |= 2^k$. I say two identities ID, ID' *form a pair* if $ID = ID_1 \oplus ID_2 \oplus ID'$ is true, where $\oplus$ represents the bit-wise X-OR operator. Note that ID_1 and ID_2 themselves also form a pair.

   Let ds be the disclosure set. For any $\lambda \in ds$, we construct λ^T by swapping the sensitive values of every pair of identities in I. The above definition of identities pair then implies the following: ID_1 and ID_2 must correspond to the same two sensitive values in λ and λ^T (in particular, the two sensitive values will be identical if λ and λ^T are the same sequence). Therefore, ID_1 and ID_2 will have the same inherent distribution, only if $\lambda^T \in ds$ is also true. I prove this by mathematical induction on k.

   - **The Inductive Hypothesis:** $\lambda^T \in ds$ holds if $\lambda \in ds$ is true.
   - **The Base Case:** The case of $k = 0$ is trivial since λ, λ^T are identical.
   - **The Inductive Case:** Suppose the inductive hypothesis holds for all $1 \le j < k$. There are two cases.
     - First, the first bit of ID_1 and ID_2 are identical. Consequently, only identities whose first bits are identical can form a pair. In such a case, λ^T will be constructed by only swapping values inside each of the two groups defined in $g_{i-1}(T)$ (these two groups are merged to form S in $g_i(T)$). The inductive hypothesis can be applied to each such group since its size is 2^{k-1}.

       By the definition of *disclosure set*, the condition $\lambda \in ds$ is true iff λ represents an instance T for which the privacy property p holds on $g_i(T)$, and p does not hold on any $g_j(T)$ for any $1 \leq j < i$. Let T' be the instance represented by λ^T. Then p must also hold on $g_i(T')$ since in constructing λ^T, the sensitive values are only swapped within the same group defined in $g_i(T)$. Moreover, p must not hold on any $g_j(T')$ for any $1 \leq j < i$ due to the inductive hypothesis. Therefore, $\lambda^T \in ds$ is true.

  – Second, the first bit of ID_1 and ID_2 are different. In this case, we can regard the construction of λ^T as a two-step process, as follows. First, we swap the sensitive values corresponding to every pair of identities in I that only differ in the first bit. Second, we swap values corresponding to every pair of identities in I whose last $k-1$ bits form a pair (with respect to the last $k-1$ bits of ID_1 and ID_2). Using a similar argument as in the above case, it can be shown that neither steps will affect the evaluation of the privacy property p. Therefore, $\lambda^T \in ds$ also holds in this case.

2. For the case where g_1 defines groups of size larger than one, the proof is similar except that a pair of identities now changes to a pair of sets of identities.

□

5.3.2 Decomposition of Generalization Process

It is important to note that, similar to most existing generalization algorithms[38, 49], a pair-based generalization algorithm must form larger groups (by merging pairs of equal-size groups) based on quasi-identifiers only, and independently of the sensitive values. This requirement is necessary because otherwise additional inference channels will be available from the particular way that the algorithm forms larger groups. The requirement also implies that the generalization process can be decomposed into two separate stages.

First, the algorithm must determine which equal-size groups are *to be* merged (if necessary) by looking at the quasi-identifiers only. Note that, at this stage, the algorithm does not yet know which group will need to be merged with others because that will depend on the sensitive values. Since the merging process is iterative and only pairs of equal-size groups will be merged, the algorithm essentially constructs a (virtual) binary tree on all of the starting groups (assume the total number of tuples is a power of two at this point). Second, the algorithm checks each group's sensitive values and starts to merge pairs of equal-size groups that are siblings of the binary tree that has been pre-determined in the first stage. Such a merging process is repeated iteratively until either all groups satisfy the privacy property (a generalization will be disclosed) or the root of the tree is reached and it does not satisfy the property (nothing will be disclosed).

Clearly, the assumption that the total number of tuples is a power of two is not always true. Therefore, before the first stage of the algorithm, some starting groups that already satisfy the privacy property should be excluded from further generalization. As long as these groups are selected randomly, excluding them will affect neither the apparent distribution nor the inherent distribution of the remaining groups. Therefore, Theorem remains true when we apply a coincident generalization algorithm on the remaining groups of the original table, that is, the algorithm is still coincident for the original input micro-data table.

It is worth noting that there exist plenty of choices in excluding starting groups from further generalization. These choices lead to different algorithms. In the next section, I will study two particular algorithms that show good performance with real-life data as demonstrated in Section 5.5. Intuitively, the first algorithm (Unsafe with Unsafe) excludes as many "safe" groups as possible, such that more groups will not be affected by further generalization. The second algorithm (Unsafe with Any) excludes as few "safe" groups as possible, such that those groups failing the privacy property can be generalized more likely with other groups that have similar quasi-identifiers. Those two algorithms provide different kinds of trade-off for data utility between the groups that already satisfy the privacy property and those groups that do not.

5.3.3 Two Pair-Based Generalization Algorithms

In this section, I study two particular pair-based generalization algorithms. The goal of this study is to demonstrate that by varying the way of merging groups, distinct pair-based generalization algorithms can be designed to satisfy the different requirements of micro-data disclosure applications.

First, given any generalization $g(T) = \langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$ $(1 \leq i \leq n)$ and any $\langle I, S \rangle \in \mathcal{R}$, we say I is an *unsafe* group, if the privacy property does not hold on S; I is a *safe* group, otherwise. Also, given any $\langle I, S \rangle, \langle I', S' \rangle \in \mathcal{R}$, by *merging* I and I', I mean to obtain a new generalization $\langle\ \mathscr{P}(\mathbb{I}) \setminus \{I, I'\} \cup \{I \cup I'\},\ \mathscr{P}(\mathbb{S}) \setminus \{S, S'\} \cup \{S \cup S'\},\ \mathcal{R} \setminus \{\langle I, S \rangle, \langle I', S' \rangle\} \cup \{\langle I \cup I', S \cup S' \rangle\}\ \rangle$.

Unsafe with Unsafe Algorithm UU first obtains starting groups and excludes as many safe groups as possible (line 1-2). It then determines which groups are to be merged by constructing a binary tree (line 3). It merges an unsafe group with another equal-size group (lines 5-6) or a new safe group created by merging existing groups of smaller sizes (lines 7-8), in the given order of preference. It returns an empty set indicating nothing should be disclosed, if none of the above options is possible (lines 9-10).

Several observations can be made about Algorithm UU. First, although the result of each iteration can be considered as a generalization function, this is not explicitly shown because only the final generalization is important to a coincident generalization algorithm. Second, in line 7 we only need to consider safe groups because the unsafe group chosen in line 2 is already the smallest among all unsafe groups. Third, in line 7, we can decide whether a new group of a required size can be obtained through recursively examining two groups of half the size, which takes linear time in the size of the unsafe group. I shall delay the discussion of the initial group size γ and the possibility of returning ϕ to Sections 5.5 and 5.6.

```
Algorithm UU Unsafe with Unsafe
Input: An instance, a distribution-based privacy property $p$, a starting size $\gamma$;
Output: A table generalization, or $\phi$;
Method:
1.  Obtain an initial generalization with group size $\gamma$;
2.  Exclude a random collection of safe groups such that the number of remaining groups
    is the least possible power of 2;
3.  Construct a full binary tree where each leaf node is a remaining group;
4.  while there exists an unsafe group $I$ of the smallest size
5.          if the sibling of $I$ is already a generalized group $I'$ then
6.          Generalize $I$ with $I'$ to form a double-sized group at the father node;
7.          else if the sibling of $I$ is not a generalized group yet then
8.          Generalize all children groups of $I$'s sibling to form a group $I'$ and
            then generalize $I$ with $I'$;
9.          else if there is no sibling of $I$, that is, $I$ is the root then
10.         Return $\phi$;
11.         end if
12. end while
13. Return all generalized groups (including the groups excluded in line 2 ).
```

Figure 5.1: Algorithm UU

Unsafe with Any Algorithm UA shows a different approach to excluding safe groups, that is, it excludes as few safe groups as possible (line 2).

Correctness and Performance Proposition 1 states the correctness of Algorithms UU and Algorithm UA. Clearly, the different approaches to merging groups adopted by Algorithms UU and Algorithm UA will lead to different performance in terms of data utility, for example, the average size of groups in the final generalization. Note that such performance depends on not only the privacy property p but also the overall distribution of sensitive values in the given instance. For example, if the desired property p is set-monotonic (that is, for any $S \subseteq S'$, S' satisfies p if S does), then Algorithm UA is likely to produce a smaller average group size, because merging an unsafe group with a safe group always yields a safe group. On the other hand, if the sensitive values in the given instance are close to a uniform distribution, then Algorithm UU may produce smaller groups, because merging unsafe groups in the given instance is more likely to yield a safe group. I shall delay the detailed study of such performance comparison to Section 5.5.

```
Algorithm UA Unsafe with Any
Input: An instance, a distribution-based privacy property $p$, a starting size $\gamma$;
Output: A table generalization, or $\phi$;
Method:
1.  Obtain an initial generalization with group size $\gamma$;
2.  Exclude a random collection of safe groups such that the number of remaining groups
    is the largest possible power of 2;
3.  Construct a full binary tree where each leaf node is a remaining group;
4.  while there exists an unsafe group $I$ of the smallest size
5.          if the sibling of $I$ is already a generalized group $I'$ then
6.          Generalize $I$ with $I'$ to form a double-sized group at the father node;
7.          else if the sibling of $I$ is not a generalized group yet then
8.          Generalize all children groups of $I$'s sibling to form a group $I'$ and then
            generalize $I$ with $I'$;
9.          else if there is no sibling of $I$, that is, $I$ is the root then
10.         Return $\phi$;
11.         end if
12. end while
13. Return all generalized groups (including the groups excluded in line 2).
```

Figure 5.2: Algorithm UA

Corollary 1. *Any generalization returned by Algorithm UU and Algorithm UA satisfies p.*

Proof: This result directly follows from Theorem 5.3.2 and Lemma 3. Specifically, both algorithms will only return a generalization in which all the groups are safe, that is, the apparent distribution of sensitive values in every group satisfies the privacy property p. By Lemma 3, p is also satisfied on the inherent distribution. □

5.4 Interleaved Coincident Generalization

In this section, I refine the pair-based coincident generalization algorithms by allowing groups to be broken and re-organized. First, Section 5.4.1 introduces a sufficient condition for coincident generalization algorithms. Section 5.4.2 then introduces the family of interleaved generalization algorithms.

5.4.1 A Sufficient Condition for Coincident Generalization Algorithms

Recall that in the proof of Theorem 5.3.2, the key to showing the inherent distribution of any two identities to be identical is to construct a one-to-one mapping on the disclosure set. Intuitively, the existence of such a mapping shows the disclosure set to be symmetric with respect to the two identities. Therefore, both identities correspond to the same set of sensitive values (but may be in different order) and their inherent distributions are thus identical. I formalize such a symmetry property of the disclosure set in Definition 22.

Definition 22. (Self-Symmetric Disclosure Set) *Given an instance T_0, a generalization algorithm G and the generalization $\langle \mathscr{P}(\mathbb{I}), \mathscr{P}(\mathbb{S}), \mathcal{R} \rangle$ $(1 \leq i \leq n)$ output by G with input T_0, the disclosure set ds is* self-symmetric*, if for any $I \in \mathscr{P}(\mathbb{I})$ and any $ID_1, ID_2 \in I$, there always exists a one-to-one mapping $F() : \mathbb{I} \rightarrow \mathbb{I}$ satisfying*

- *$F(ID_1) = ID_2, F(ID_2) = ID_1$, and*
- *for every $\lambda \in ds$, the sequence λ^T obtained by replacing the sensitive value corresponding to ID with that corresponding to $F(ID)$ must appear in ds.*

Example 6. *In Table 5.6, it can be verified that the disclosure set is self-symmetric. For example, with respect to Alice and Diana, $\lambda_1 \in ds \Leftrightarrow \lambda_4 \in ds$ and $\lambda_2 \in ds \Leftrightarrow \lambda_3 \in ds$.*

Lemma 4 shows the self-symmetry of disclosure sets to be a sufficient condition for the generalization algorithm to be coincident. I shall employ this fact in following discussions.

Lemma 4. *A generalization algorithm $G = \langle g_1 g_2 \ldots g_n, p \rangle$ is coincident, if for any T_0 and $g_i(T_0)$ $(1 \leq i \leq n)$, where $g_i(T_0)$ is output of G with input T_0, the disclosure set is self-symmetric.*

Proof: This result is straightforward since the existence of a one-to-one mapping given in the definition of a self-symmetric disclosure set implies that the number of occurrences of any sensitive value in ds must be identical for any pair of identities within the same group. $\square$

Table 5.6: An Example of Self-Symmetric Disclosure Set

$\mathbb{I}$	$g_1(T_0)$		$g_2(T_0)$		$\mathbb{S}$
	$\mathscr{P}_1(\mathbb{I})$	$\mathscr{P}_1(\mathbb{S})$	$\mathscr{P}_2(\mathbb{I})$	$\mathscr{P}_2(\mathbb{S})$	
Alice	I^1_1	S^1_1	I^2_1	S^2_1	cancer
Brenda					cancer
Clare	I^1_2	S^1_2			flu
Diana					tracheitis

ds			
λ_1	λ_2	λ_3	λ_4
cancer	cancer	flu	tracheitis
cancer	cancer	tracheitis	flu
flu	tracheitis	cancer	cancer
tracheitis	flu	cancer	cancer

5.4.2 Interleaved Generalization Algorithms

I now study another family of coincident generalization algorithms, namely, *interleaved* generalization algorithms. Informally, the algorithms studied in the previous section never break any existing group in constructing generalization functions. In contrast, an interleaved generalization algorithm may reorganize members of existing groups into different groups. As I shall show in Section 5.5, this approach may provide better utility compared to pair-based generalization algorithms.

A Special Case To build intuitions, I first describe a special case. Consider the following simple algorithm G_{23} with two aggregation functions g_1 and g_2. First, G_{23} generalizes a given instance using g_1 into groups of size 2γ. Second, for every collection of three groups that needs to be further generalized, G_{23} generalizes the three groups into two new groups both of size 3γ, using g_2, as illustrated in Figure 5.3. The algorithm returns ϕ if the result of g_2 does not satisfy the privacy property.

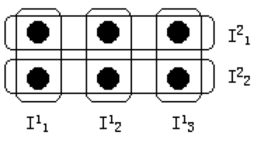


Figure 5.3: A Special Case of Interleaved Generalization

Lemma 5. *Any generalization returned by the above algorithm G_{23} has a self-symmetric disclosure set, and thus G_{23} is coincident.*

Proof: I only show the case of $\gamma = 1$. The generalization given by g_1 clearly has a self-symmetric disclosure set. For the generalization given by g_2, without loss of generality, let ID_1 and ID_2 be the left and middle identities in I_1^2, respectively, and ID'_1 and ID'_2 the left and middle identities in I_2^2, respectively. Let λ be any sequence in the disclosure set ds. I construct λ^T by swapping the sensitive values between ID_1 and ID_2 and also between ID'_1 and ID'_2. Clearly, if each pair of swapped values is identical, then $\lambda = \lambda^T$. Otherwise, I have that $\lambda^T \in ds$ because of the following. By swapping values as described above, I have essentially reordered the two groups I_1^1 and I_2^1. None of the membership relationships of the six groups are affected. Therefore, the algorithm will still return the same generalization (or ϕ) after the swapping. That is, $\lambda^T \in ds$, and ds is self-symmetric. $\square$

Definition 23. (Interleaved Generalization Algorithms) *An interleaved generalization algorithm $G = \langle g_1 g_2 \dots g_n, p\rangle$ satisfies that, for any input instance T_0 and $g_i(T)$ ($1 \le i \le n$) where $g_i(T_0)$ is output of G with input T_0, for any ID_1,ID_2 that appear in any group I_a^i defined in $g_i(T_0)$ ($1 \le i \le n$), one of the following must hold for all $j < i$:*

- *ID_1 and ID_2 both appear in some group I_b^j defined in $g_j(T_0)$, or*
- *$ID_1 \in I_b^j$, $ID_2 \in I_c^j$, and there exists a one-to-one mapping $f() : I_b^j \to I_c^j$ satisfying*
  - *$f(ID_1) = ID_2$, and*
  - *any $ID \in I_b^j$ and $f(ID)$ both appear in some group I_d^i defined in $g_i(T_0)$.*

Theorem 15. *Any interleaved generalization algorithm $G = \langle g_1 g_2 \dots g_n, p\rangle$ is coincident.*

Proof: See the Appendix. $\square$

Example 7. *Figure 5.4(A) illustrates an interleaved generalization algorithm with five generalization functions. It can be verified that for any two identities, we can always swap the*

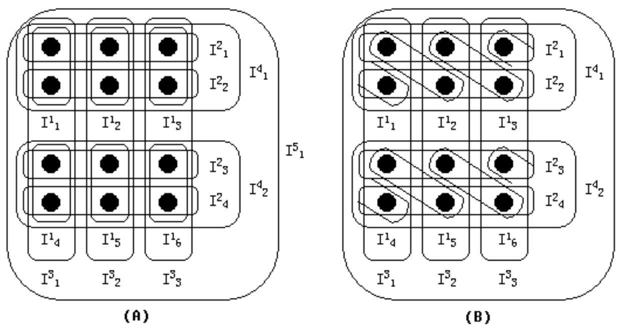


Figure 5.4: Examples of Interleaved Generalization and Non-Interleaved Generalization

sensitive values so that the two identities may exchange their sensitive values without breaking any group. Therefore, the resulting sequence of sensitive values must lead to the same generalization result and thus appear in the disclosure set. This shows the disclosure set to be self-symmetric. It is worth mentioning that an appealing but incorrect definition of interleaved generalization is that we only require the two conditions in Definition 23 for $i-1$ *instead of all* $j < i$*. As illustrated in Figure 5.4(B), swapping values between the* $i-1$ *groups can potentially break lower level groups, which means this is not an interleaved generalization algorithm.*

5.5 Experiments

In this section, I present experimental results on the performance of the proposed coincident generalization algorithms in terms of data utility. I used the popular Adult Data Set taken from the UCI data repository [1]. The quasi-identifier attributes used to form groups of individuals include *Age*, *education*, and *workclass*. The sensitive attribute is *occupation*, which has 14 different values. Our experiments are repeated over three samples of the dataset: all tuples, and tuples having at least one of the most frequent 8 and 5 sensitive values, respectively. I test two popular privacy properties: *p-sensitive* [50] and *Entropy l-diversity* [40] between which *p*-sensitive is set-monotonic. I regard the average group size in the final generalization as the data utility metric, which is similar to the DM metric [51] when suppression is not taken into consideration.

5.5.1 Performance of Pair-Based Coincident Generalization Algorithms

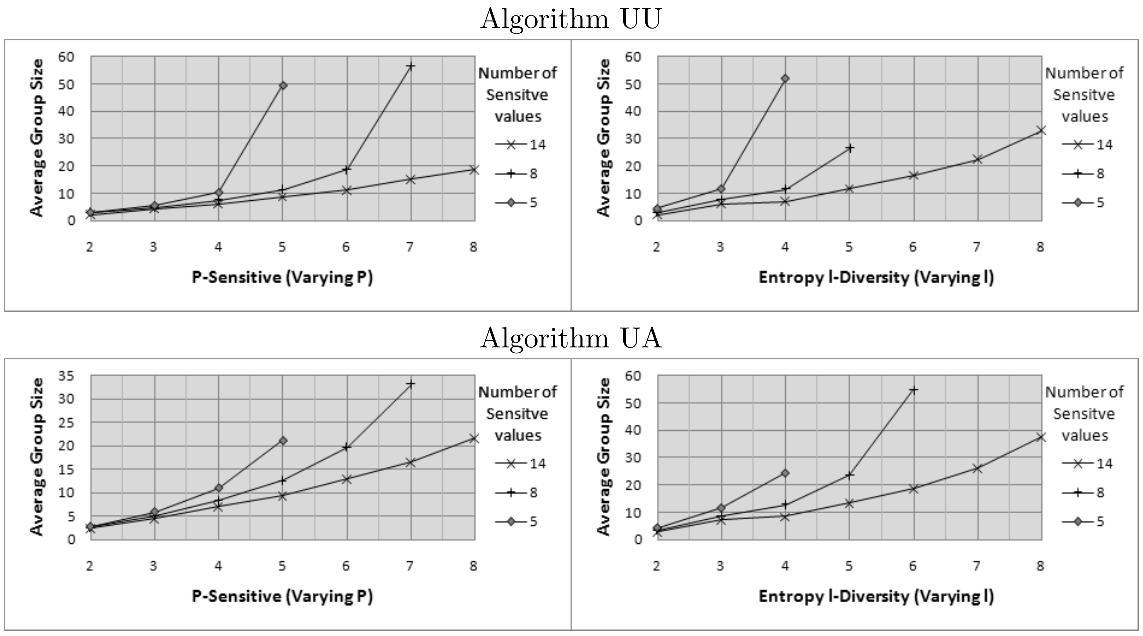
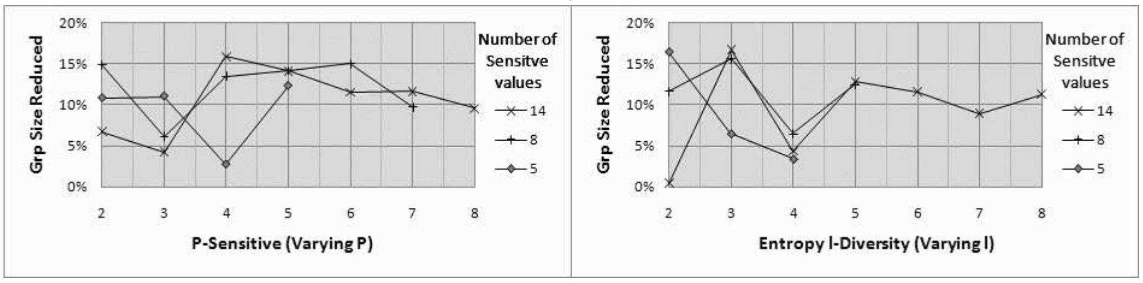


Figure 5.5: Performance of Algorithms UU and UA

Figure 5.5 shows the performance of Algorithm UU (Unsafe with Unsafe) and Algorithm UA (Unsafe with Any), respectively. In Figure 5.5, the x-axis is the parameter of each privacy property, that is, p in p-sensitive and l in Entropy l-diversity. The y-axis represents the average group size in the final generalization. Each line represents one of the three sampled datasets with a different number of distinct sensitive values, 14, 8, and 5, respectively. From the experiments, we can observe that the average group size generally increases with the parameter of a privacy property. This is because a larger parameter represents a more rigid requirement that demands larger groups. The group size decreases as the number of sensitive values increases, because in general a privacy property becomes easier to satisfy with more sensitive values. Also, there is a slight difference between the performances of Algorithms UU and UA, which is discussed in the next section.

G^I over Algorithm UU



G^I over Algorithm UA

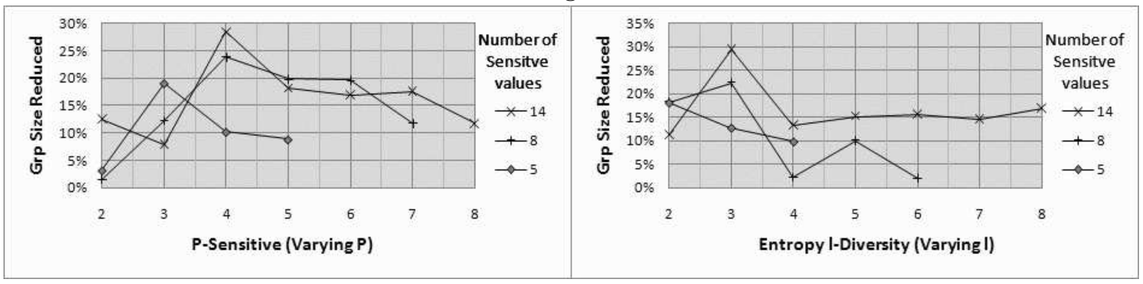


Figure 5.6: Performance of G^I over Algorithms UU and UA

5.5.2 Performance of Interleaved Coincident Generalization Algorithms

Figure 5.6 shows the performance improvement of two interleaved generalization algorithms, both having the size sequence $G^I = (2, 3, 6, 6{*}2, 6{*}2^2, 6{*}2^3, \ldots)$. Compared to Algorithm UU is the one applying the same "Unsafe with Unsafe" strategy as Algorithm UU. Compared to Algorithm UA is the one applying the same "Unsafe with Any" strategy as Algorithm UA. The y-axis is the percentage of reduction in average group size when comparing the above two interleaved algorithms to the two pair-based algorithms. We can see a significant improvement in terms of reduced average group size by applying the interleaved generalization algorithms. There are exceptions when the optimal starting group size is very small, which indicates that the selection of the interleaved generalization size sequence should be based on the anticipated optimal starting group size.

5.5.3 Comparison with Unsafe Generalization Algorithm

I now compare the performance of the proposed coincident generalization algorithms with that of the Mondrain generalization method [52] on the whole adult data set with the same setting described above. It is important to note that the method in [52] can only guarantee a privacy property when the algorithm itself remains secret, whereas my coincident generalization algorithms will remain safe even when publicized. Therefore, this experiment essentially shows the additional cost, in terms of loss in data utility, of guaranteeing privacy using public algorithms. As shown in Figure 5.7, such a cost is only marginal with either of my coincident algorithms.

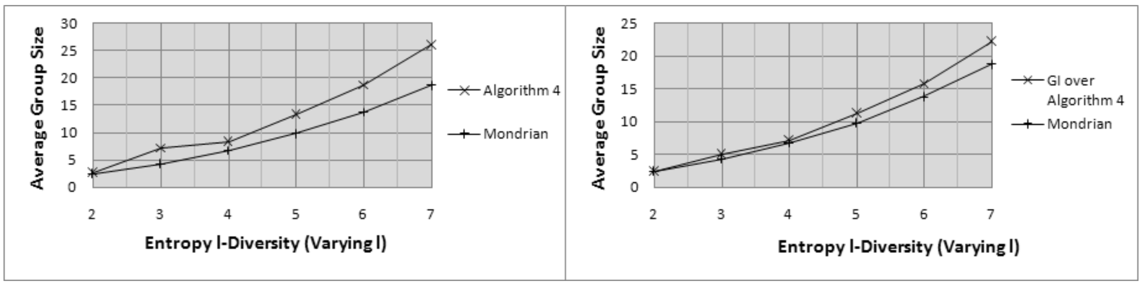


Figure 5.7: Comparing Two Algorithms: Algorithm UA and G^I, with Mondrain

5.6 Discussion

Starting Group Size:

I study the effect of the starting group size γ on the performance of the generalization algorithms with the experiment shown in Figure 5.8. I tested Algorithm UA with Entropy 8-diversity on the whole dataset. We can observe that the starting group size only has a minor effect on performance. Note that for Algorithms UU and UA, a starting group size

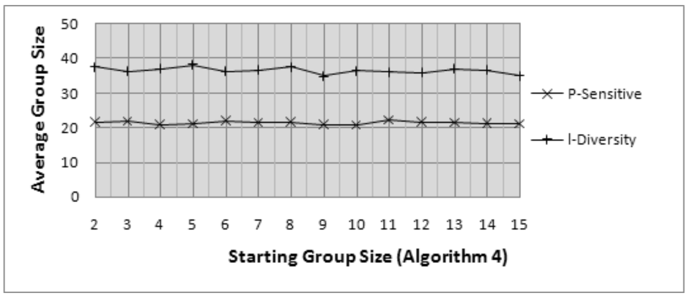


Figure 5.8: Algorithm UA with Different Group Size

γ smaller than the parameter of privacy property will be meaningless since every pair of groups will be immediately merged and thus the performance will be identical to the case of 2γ.

Distribution of Sensitive Values:

As I discussed in Section 5.2, if the distribution of the sensitive values is relatively balanced (e.g., a uniform distribution), Algorithm UU is more likely to have better performance than Algorithm UA; under an unbalanced distribution, such as binomial distribution, Algorithm UA is likely to have better performance than Algorithm UU. This is evidenced by the experimental results based on randomly generated data, with uniformly distributed sensitive values and binomially distributed sensitive values, respectively. The results are shown in Figure 5.9, where the y-axis shows the difference between the average group size of Algorithm UA and that of Algorithm UU.

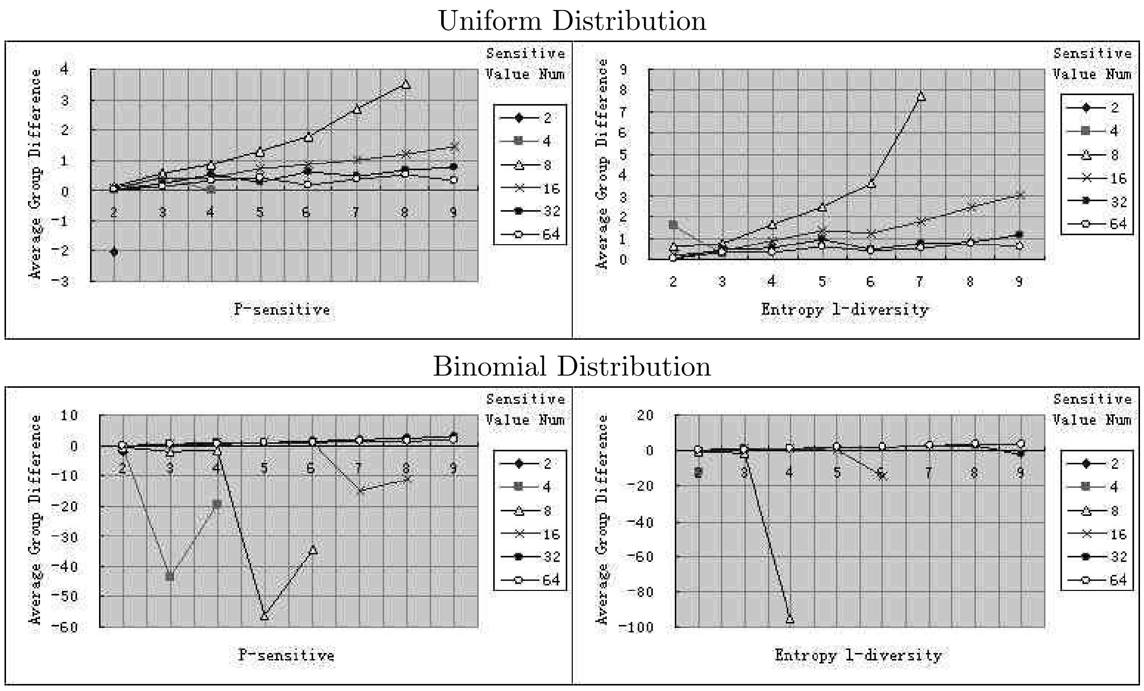


Figure 5.9: Difference between Algorithms UA and UU

Other Data Utility Metrics:

In this chapter, I consider data utility in terms of average group size in the generalization result. In some applications, the group size in the space of generalization attributes is much more important. If so, Algorithm UA will be more practical than Algorithm UU. That is, Algorithm UA can be used to optimize such utility metrics. The reason is that, with the "Unsafe With Any" combining strategy, Algorithm UA can always generalize an "Unsafe" group with its neighbor group, that is, the closest group measured in the space of generalization attributes, to form a "safe" group.

Possibility to Disclose Nothing:

My generalization algorithms all have a possibility of returning ϕ, which means nothing can be disclosed. This is necessary because it is always possible that even after generalizing all identities into one group, the desired privacy property still cannot be satisfied. For example, consider a case where we want to guarantee entropy 2-diversity and the number of different sensitive values is 2. Then if the number of occurrences of the two sensitive values is not identical, any generalization, including the one with a single group, cannot satisfy the desired entropy 2-diversity. In this case, which obviously happens with a very low probability, suppression is a must to solve the problem. To extend my technique with suppressions remains a future work.

5.7 Summary

A common assumption made by existing information disclosure solutions is that the disclosed data is the only source of information available to an adversary. However, with knowledge of the algorithm used for computing the disclosed data, an adversary may deduce more information to violate a desired privacy property. In this chapter, I studied this issue in the context of generalization-based micro-data disclosure algorithms. I then introduced two families of efficient coincident generalization algorithms to guarantee that the knowledge about algorithms will not help an adversary improve his/her guess about the distribution of sensitive values. The algorithms are efficient, and experimental results show that they provide reasonably good data utility. It is my belief that this technique can be extended to a broader range of data applications.

For future work, I shall investigate the case where a generalization algorithm selects a generalization function to apply based on sensitive values of the given table instance (more inference channels will be available to adversaries). I will also extend the proposed coincident generalization algorithms to support suppression.

5.8 Proof of Theorem 15

By Lemma 4, I only need to show that the disclosure set of g_i is self-symmetric. I prove the result by mathematical induction on i.

The Inductive Hypothesis: Under every g_i, the disclosure set ds is self-symmetric.

The Base Case: I need two base cases, that for $i = 1$ and $i = 2$.

- For any group I_j^1 generalized by g_1, let S be the corresponding group of sensitive values and $ds(S)$ be the multiset $\{\lambda : \lambda$ is the subquence corresponding to I_j^1 in some $\lambda' \in ds\}$. In this case, the theorem obviously holds, because $ds(S) = \mathscr{M}(S)$. For any $\lambda \in ds(S)$, I can construct F as $F(ID_1) = ID_2, F(ID_2) = ID_1$ and $F(ID) = ID$ for all $ID \neq ID_1, ID \neq ID_2$.
- For any group I_j^2 generalized by g_2 and any $ID_1, ID_2 \in I_j^2$, let I_b^1 be the group generalized by g_1 such that $ID_1 \in I_b^1$,
  - If $ID_2 \in I_b^1$, I construct F as follows:
    - $F(ID_1) = ID_2$, $F(ID_2) = ID_1$;
    - for any $ID \in \mathbb{I}$ other than ID_1, ID_2, $F(ID) = ID$.

  - If $ID_2 \notin I^1_b$, let $I^1_{b'}$ be the group generalized by g_1 such that $ID_2 \in I^1_{b'}$. By definition, there exists a one-one mapping $f() : I^1_b \rightarrow I^1_{b'}$ such that $f(ID_1) = ID_2$ and for any pair $ID, f(ID)$, there exists $I^2_{a'}$ such that $ID, f(ID) \in I^2_{a'}$. I construct $F(\lambda)$ as follows:
    - For any $ID \in I^1_b$, $F(ID) = f(ID)$;
    - For any $ID \in I^1_{b'}$, $F(ID) = f^{-1}(ID)$;
    - For any else ID, $F(ID) = ID$.
  - In either case, I have that for any $\lambda \in ds$, the sequence λ^T defined on F also satisfies $\lambda^T \in ds$. Therefore, ds is self-symmetric.

The Inductive Case: Suppose the inductive hypothesis holds for any $j \leq i-1$: I show that it also holds for i. For any group I^i_a generalized by g_i and any $ID_1, ID_2 \in I^i_a$, we construct F recursively using the following *procedure ConsF*(i) with the parameter $i(i > 2)$.

Procedure ConsF(i)

- Let I^{i-1}_b be the group generalized by g_{i-1} such that $ID_1 \in I^{i-1}_b$;
- If $ID_2 \in I^{i-1}_b$:
  - Based on the inductive hypothesis, let F' be the constructed one-to-one mapping (either using ConsF or the two base cases) for ID_1 and ID_2 under g_{i-1};
  - Construct F as:
    - For any $ID \in \mathbb{I}$, $F(ID) = F'(ID)$.
  - Return F.
- If $ID_2 \notin I^{i-1}_b$, suppose $ID_2 \in I^{i-1}_{b'}$:
  - Let I^{i-2}_c be the group generalized by g_{i-2} such that $ID_1 \in I^{i-2}_c$;
  - If $ID_2 \in I^{i-2}_c$:

    * Based on conditions of the theorem, let $f : I_b^{i-1} \rightarrow I_{b'}^{i-1}$ be the one-to-one mapping such that every pair $ID \in I_b^{i-1}, f(ID) \in I_{b'}^{i-1}$ is generalized in the same group by g_{i-2};
    * Note that, such f exists because one-one mapping $f' : I_d^{i-2} \rightarrow I_{d'}^{i-2}$ exists for any two identities $ID' \in I_b^{i-1} \cap I_d^{i-2}$ and $ID'' \in I_b^{i-1} \cap I_{d'}^{i-2}$;
    * Based on the inductive hypothesis, let F' be the constructed one-to-one mapping (either using ConsF or the two base cases) for ID_1 and ID_2 under g_{i-2};
    * Construct F as:

      For any $ID \in \mathbb{I}$, $F(ID) = F'(ID)$;

      For any $ID \in I_b^{i-1}$ and $f(ID) \in I_{b'}^{i-1}$ that have not been swapped by F', let F'' be the constructed one-to-one mapping (either using ConsF or the two base cases) for ID and $f(ID)$ under g_{i-2}, replace F as:

      For any $ID \in \mathbb{I}$, $F(ID) = F''(F(ID))$;

      Note that, based on conditions of the theorem, this will not affect any identities whose sensitive values have already been swapped.
    * Return F.

- If $ID_2 \notin I_c^{i-2}$, suppose $ID_2 \in I_{c'}^{i-2}$:
    * Select $ID(w) \in I_c^{i-2}, ID(w') \in I_{c'}^{i-2}$ and the one-to-one mapping $f : I_b^{i-1} \rightarrow I_{b'}^{i-1}$, such that every pair $ID \in I_b^{i-1}, f(ID) \in I_{b'}^{i-1}$ is generalized in the same group by g_{i-2}, and $f(ID(w)) = ID_2$, $f(ID_1) = ID(w')$;
    * Construct F^* in a similar way as the above case but for $ID_1, ID(w')$ instead of for ID_1 and ID_2;
    * Based on the inductive hypothesis, let F'' be the constructed one-to-one mapping (either using ConsF or the two base cases) for ID_1 and $ID(w)$ under g_{i-1};

- $*$ Construct F as:

  For any $ID \in \mathbb{I}$, $F(ID) = F''(F^*(ID))$;

- $*$ Return F.

From the construction above, I have that for any $\lambda \in ds$, the λ^T defined on F also satisfies $\lambda^T \in ds$. Therefore, ds is self-symmetric under g_i. This concludes the proof. $\square$

Chapter 6: Privacy Protection and Restoring in Multi-Source Micro-Data Disclosure

6.1 Introduction

We have been faced with an information explosion in recent years. Search and data mining have been extensively used to help information consumers to filter and absorb the information to which they are exposed. However, the vulnerability of privacy protection has not been adequately addressed and remains a major concern. It is alarming that, for example, the medical information of a person can be easily identified just by linking a public voters' list with some anonymized datum in medical research ([31, 46]). There has been a considerable body of research on how to mitigate such a rising threat to privacy, i.e., to prevent the relation between individuals and sensitive attributes from being identified ([31, 34, 40, 46, 53, 54]). The standard way to preserve privacy is based on monitoring the entire information disclosure process and rejecting any request with a potential privacy violation.

However, there are multiple reasons for failure in preventing privacy violations. First, multiple sources may release information on individuals without centralized disclosure control. In this case, even if information disclosed by each source preserves privacy, the combined information may violate privacy. For example, as shown in ([46]), anonymized disclosure of medical records from a hospital, combined with general information from a voters' list, can be used to infer medical conditions of individuals, which clearly violates their privacy. Second, the requirements for privacy may increase over time due, for example, to new legislation, while sensitive information has already been disclosed. Third, privacy violation may also arise in practice due to exceptions in information disclosure control.

To address the problem, I investigate the following question in this chapter. When a desired privacy property is already compromised due to accidental information disclosure, is there a way to restore it? At first glance, the answer seems to be negative, because the information disclosure process is irreversible. That is, the more information about secret data is disclosed, the more constrained the set of "possible worlds" becomes.

However, for privacy properties that are related to probability distribution or entropy, there is a way to restore the compromised privacy under certain conditions. Although we cannot reverse the disclosure process, we may still be able to restore the compromised privacy by disclosing more true information.

More generally, the question I investigate in this chapter is as follows. Given a set of information objects that jointly violate a desired privacy property, can we extend it to a superset that satisfies the same privacy property. Note that this question is important not only for restoring accidentally compromised privacy as described above, but also for the centralized disclosure control to provide better data availability. For example, if the centralized disclosure control cannot release a collection of data because of privacy violation, we may be able to disclose the data collection in conjunction with additional data; if disclosed together, the privacy can be preserved.

It is worth noting, before giving a concrete example, that apparently the theorem above seems very counter-intuitive. However, as I will also explain in the following example, the idea is not contradicting with the common intuition at all, i.e., in general, more disclosed true information should always lead to worse privacy. The truth is, in an information disclosure problem where privacy of a small group of people does not have enough protection due to some well-defined regulations, there may exist another group of people whose privacy is "over-protected." I observe that in this case, it is possible to sacrifice the "over-protected" privacy of the latter group of people, as long as they meet the regulations and, in return, the privacy of the former group of people can be restored. At the same time, the privacy protection in general is still getting worse. Intuitively, by disclosing more true information, we will always decrease the privacy protection in general but may be able to increase the privacy protection on a small part of the entire group.

A Motivating Example

I limit the scope here to the problem of micro-data disclosure and I consider only generalization-based view disclosure. Some other techniques like data swapping and perturbation ([26–29]) that can also be utilized in the micro-data problem, as discussed in the related work, are not covered in this chapter. To explain the basic idea, consider a medical information disclosure example. A table of patients' medical information is shown in Table 6.1, part of which is publicly accessible through, say, a voters' list, shown in Table 6.2.

Table 6.1: Patient Information Table

Name	**Sex**	**Age**	**Employer**	**Condition**
Alan	M	23	ABC, Inc.	Heart Disease
Bob	M	24	ABC, Inc.	SARS
Clark	M	25	ABC, Inc.	Viral Infection
Donald	M	26	ABC, Inc.	SARS
Ellen	F	27	ABC, Inc.	Viral Infection
Fen	F	28	ABC, Inc.	SARS
Garcia	F	28	ABC, Inc.	Flu

Table 6.2: Information from Voters' List

Name	**Sex**	**Age**	**Employer**
Alan	M	23	ABC, Inc.
Bob	M	24	ABC, Inc.
Clark	M	25	ABC, Inc.
Donald	M	26	ABC, Inc.
Ellen	F	27	ABC, Inc.
Fen	F	28	ABC, Inc.
Garcia	F	28	ABC, Inc.

Now assume one view of Table 6.1 is disclosed by a data authority (e.g., hospital) upon request as shown in Table 6.3(A). Note that this view is generalized enough not to reveal the individuals' medical conditions, which are considered sensitive. More formally, Table 6.3(A) satisfies the property of *Recursive (1,2)-Diversity* ([40]). This means that, in any group of individuals present in the view and indistinguishable by non-sensitive attributes (*i.e., four* *tuples in the view having the same Sex Attribute "Male" form a group*), the maximum frequency of any sensitive value cannot exceed 0.5. To calculate the frequency of a sensitive value, the number of occurrences of the sensitive value in the multiset of sensitive values, which are associated with the group, are counted and then divided by the size of the multiset. In the example, the condition $SARS$ has the maximum appearance frequency of 0.5.

Table 6.3: Two Disclosed Views of Table 1

Sex	**Condition**
M	Heart Disease
M	SARS
M	SARS
M	Viral Infection

(A)Male-Cond in ABC, Inc. (first four tuples)

Age	**Condition**
26˜28	Flu
26˜28	SARS
26˜28	SARS
26˜28	Viral Infection

(B)Age-Cond in ABC, Inc. (last four tuples)

Recall that we want to prevent any individual's medical condition from being identified through the combined disclosed and public information. As I will prove in Section 6.2, the satisfaction of *Recursive (1,2)-Diversity* by Table 6.3(A) guarantees that any adversary cannot win the following game to infer any individual's medical condition with the probability of 0.5. Assume the adversary has a random oracle ([55]) whose output domain is the set of all possible patient information tables that yield the same result as the original Table 6.1 in both the disclosed views (Table 6.3(A)) and public information (Table 6.2), if computed the same way. For example, Table 6.4 could be one of the possible outputs. Intuitively, a random oracle will respond to every query with a random response chosen uniformly from its output domain. The adversary is interested in knowing whether an individual, *id*, that appears in Table 6.2, is associated with a medical condition *s*. He or she will query the random oracle for an answer. The adversary wins if in the outcome (a possible table), *id* is associated with *s*. I say that there is a violation of privacy if, for any individual and medical condition, the probability for the adversary to win this game is higher than 0.5 or another pre-established bound. Clearly, the adversary can estimate the probability of winning the game within any desired statistical confidence interval by playing the game sufficiently many times.

Table 6.4: A Possible Patient Information Table Regarding the Disclosed Views Shown in Table 6.2 and Table 6.3(A), where (*any*) represents any possible medical condition

Name	**Sex**	**Age**	**Employer**	**Condition**
Alan	M	23	ABC, Inc.	SARS
Bob	M	24	ABC, Inc.	Heart Disease
Clark	M	25	ABC, Inc.	Viral Infection
Donald	M	26	ABC, Inc.	SARS
Ellen	F	27	ABC, Inc.	(*any*)
Fen	F	28	ABC, Inc.	(*any*)
Garcia	F	28	ABC, Inc.	(*any*)

In this example, the output domain of the adversary's random oracle is rather simple. It can be shown that the probability of $Alan$ having $SARS$, $Heart\ Disease$, or $Viral\ Infection$ in an outcome is 0.5, 0.25, or 0.25, respectively. The same result applies to Bob, $Clark$, and $Donald$.

Similar to Table 6.3(A), Table 6.3(B) presents another generalized view that also satisfies the desired *Recursive (1,2)-Diversity*. That is, by disclosing Table 6.3(B) (and also Table 6.2) to the adversary, we will not have a violation of privacy. However, if the adversary gets both Table 6.3(A) and Table 6.3(B), privacy will be violated. Note that a property defined on a single view, such as *Recursive (1,2)-Diversity*, cannot be applied to this multiple view disclosure case to protect the privacy. Still, we can check privacy violation through the above game based on the adversary's random oracle. In this case, based on the disclosed information, *Donald* can only be associated with either $SARS$ or $Viral\ Infection$. With the constraint of *Donald* being associated with $SARS$, there are $3! \times 3!$ such tables in the output domain of the adversary's random oracle. With the constraint of *Donald* being associated with $Viral\ Infection$, there are 3×3 such tables in the output domain of the adversary's random oracle. Therefore, we can compute that the probability of *Donald* having $SARS$ in an outcome is 0.8 and the desired privacy property is violated.

Based on the above observation, it is clear that Table 6.3(A) and Table 6.3(B) should not be disclosed together in order to protect *Donald*'s privacy. However, as I have discussed in the beginning, without a centralized data disclosure control authority, it may happen that two distributed data authorities, hospital and insurance company, for example, may disclose the two views above without notifying each other. In this case, all of the privacy protection techniques based on preventing unsafe disclosures in advance would fail.

Unfortunately, once Tables 6.3(A) and 6.3(B) have been disclosed, we cannot revoke them from the adversary's knowledge. But we can discover that in this case, when *Donald*'s privacy is not well protected, the privacy of *Alan*, *Bob*, *Clark*, *Ellen*, *Fen*, *and Garcia* are "over protected." That is, *Alan* can be associated with $Heart\ Disease$, $SARS$, and $Viral\ Infection$, and when checking with the adversary's random oracle, the probabilities of *Alan* having $Heart\ Disease$, $SARS$, and $Viral\ Infection$ in an outcome are $1/3(\approx 0.3), 2/5(= 0.4)$, and $4/15(\approx 0.3)$, respectively. This also applies to *Bob* and *Clark*. And when checking with the adversary's random oracle, the probabilities of *Ellen* having Flu, $SARS$, and $Viral\ Infection$ in an outcome are $1/3(\approx 0.3), 2/5(= 0.4)$, and $4/15(\approx 0.3)$, respectively. This also applies to Fen and *Garcia*. Clearly, the privacy protection of these six people is not close to the desired bound (a probability of 0.5). This gives us a chance, by sacrificing the "over-protected" part, to restore the violated privacy of *Donald*, in a way of disclosing more true information about the original Table 6.1. One way to do it is to disclose the following two views of Table 6.1 (shown in Table 6.5).

Table 6.5: Further Disclosed Views

Age	**Condition**
25˜26	Viral Infection
25˜26	SARS

(A)Age-Cond in ABC, Inc.

Age	**Condition**
26˜27	SARS
26˜27	Viral Infection

(B)Age-Cond in ABC, Inc.

Now with the four views (Tables 6.3(A,B) and Tables 6.5(A,B)) disclosed, we can compute that the probability of *Donald* having $SARS$ in an outcome is reduced to 0.5 and the probability of *Donald* having $Viral\ Infection$ is also 0.5. This result occurs because, in the updated output domain of the adversary's random oracle, there are 2×2 tables with *Donald* being associated with $SARS$ (at the same time, both *Clark* and *Ellen* being associated with $Viral\ Infection$), and there are 2×2 tables with *Donald* being associated with $Viral\ Infection$ (at the same time, both *Clark* and *Ellen* being associated with $SARS$).

Similarly, it can be shown that the probability for any other person to have any medical condition in an outcome is equal to 0.5 in the worse case. Therefore, the desired privacy property is now satisfied. Clearly, at the same time, the privacy protection in general is still getting worse, i.e., there are 45 different tables in the outcome of the adversary's random oracle with Tables 6.3(A,B) disclosed and there are only 4 different tables in the outcome of the adversary's random oracle with all four tables disclosed. This illustrates the ability of the proposed technique to restore the compromised privacy through additional information disclosure.

Clearly, it is not always possible to restore compromised privacy by the same technique. For example, in the extreme case, if we have disclosed to the adversary that *Donald* has $SARS$, nothing can be done to restore the privacy.

I should also clarify that in the above example, I assume that the adversary cannot differentiate the disclosure of Tables 6.3(A,B) and the disclosure of Tables 6.5(A,B), i.e., the adversary does not have the knowledge that the disclosure of Tables 6.5(A,B) is to restore the violated privacy by the disclosure of Tables 6.3(A,B). This can be reasonable in practice when, for example, two disclosures are executed from different information sources for different purposes or Tables 6.5(A,B) can be just disclosed instead of Tables 6.3(A,B) when the disclosure control monitor finds out the potential privacy violation by disclosing Tables 6.3(A,B). I will also discuss how to apply the proposed technique under more flexible assumptions.

Summary

In this chapter, I study the problem of restoring compromised privacy for micro-data disclosure with multiple disclosed views. More specifically, the contributions of this chapter are as follows.

First I propose a new property, called γ-Privacy, for privacy protection in a micro-data disclosure problem when multiple views are disclosed. Given the disclosed views and publicly available information, the set PIS of "all possible worlds"(*i.e., possible tables that* *would yield the same disclosure results*), is defined. γ-Privacy intuitively means that in a randomly (uniformly) selected instance, the probability of any individual being associated with a sensitive value is at most γ. I then prove that, for the case of a single disclosed view, γ-Privacy is equivalent to the property of Recursive $(\frac{\gamma}{1-\gamma}, 2)$-Diversity. This intuitively means that the proposed property is a "natural" extension of l-Diversity, which is defined only for a single disclosed view, to multiple views.

Second, I prove that deciding on whether γ-Privacy is satisfied by a set of disclosed views is #P-complete. Third, to mitigate the high computational complexity, I relax the property of γ-privacy to be satisfied with (ϵ, θ) confidence, i.e., that the probability of disclosing a sensitive value of an individual be at most $\gamma + \epsilon$ with statistical confidence θ, where ϵ is an arbitrary small positive constant. I propose a Monte Carlo-based algorithm to check the relaxed property in $O((\lambda\lambda')^4)$ time for constant ϵ and θ, where λ is the number of tuples in the original table and λ' is the number of different sensitive values in the original table.

Finally, I turn to the problem of restoring compromised privacy. Namely, given a set of disclosed views that violates γ-Privacy, can we extend it to a superset of views that jointly satisfy γ-Privacy. I propose heuristic polynomial time algorithms, which are based on enumerating and checking additional disclosed views. I conduct a preliminary experimental study on heart disease records taken from the UCI data repository ([1]), which demonstrates that the proposed polynomial algorithms restore privacy in up to 60% of compromised disclosures. I also discuss how to apply the proposed technique under different assumptions when the adversary is also aware of this technique.

6.2 Modeling the Problem

In this chapter, I focus on the problem of micro-data disclosure. Consider a micro-data table *baseT* with schema $D = (ID, QI_1, \ldots, QI_a, SA_1, \ldots, SA_b)$, where: (1) ID is an attribute used to identify an individual, such as $Name$ or SSN; (2) $QI_1, \ldots, QI_a$ are attributes that serve as quasi-identifiers of the ID attributes *(i.e., they can be used to identify an individual* *or a small set of individuals.)*, such as *Age*, *Employer*, or *Address*; (3) $SA_1, \ldots, SA_b$ are attributes that are considered private information, such as a medical condition. In this chapter, I limit the scope to the cases such that ID is a key of *baseT* and there is only one SA attribute in *baseT*. Also, I use a multiset version of the relational model and algebra ([56]).

I assume that information disclosures about *baseT* are limited to the following two forms:

- Public Knowledge disclosure, *publicT* is a projection of *baseT* without the private attribute. I.e., $publicT = \pi_{ID,QIs}(baseT)$.
- Generalized disclosure, (V, ψ), is a generalized view of *baseT* that is disclosed upon request. I.e., $V = \pi_{SA}(\sigma_\psi(baseT))$, where ψ is a propositional formula on QI attributes that represents a generalization and is also publicly known. Note that we may have a sensitive value appearing multiple times in V.

Note that, for a particular table *baseT*, there can be only one Public Knowledge disclosure but multiple generalized disclosures. For the sake of simplicity, I will call the combination of the Public Knowledge disclosure and the multiple generalized disclosures a "micro-disclosure" of *baseT*, denoted by $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$. In the previously discussed medical information example, Table 6.2 is a Public Knowledge disclosure of Table 6.1, while Table 6.3(A) and Table 6.3(B) are two generalized disclosures. For Table 6.3(A), $Sex = Male$ is the selection condition and $26 \leq Age \leq 28$ is one for Table 6.3(B).

I assume that an adversary is able to collect all the disclosed information. To obtain the relation between ID and SA of the original table *baseT*, for every generalized disclosure (V, ψ), the adversary can compute a new view $V^I = \pi_{ID}(\sigma_\psi(publicT))$. The adversary then gets to know that the sensitive values associated with the IDs that appear in V^I are the values that appear in V. All of these disclosed information serve as constraints on the adversary trying to correctly guess the original table. In other words, we can represent the adversary's knowledge of the original table *baseT* by a *possible instance set*, defined as follows:

Definition 24. *Given a micro-disclosure* $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$*, the Possible Instance Set, PIS, is the set of all tables* T *such that:*

- $\pi_{ID,QI}(T) = publicT$ *and*
- $\forall i (1 \leq i \leq n), \pi_{SA}(\sigma_{\psi_i}(T)) = V_i$

Without loss of generality, in the remainder of this chapter, I assume that in any micro-data disclosure problem, the generalized disclosures $(V_1, \psi_1), \ldots, (V_n, \psi_n)$ of a given *baseT* always satisfy the following two properties:

(1) The generalized disclosures provide a *cover* for the tuples in *baseT*. That is, for any $id \in \pi_{ID}(baseT)$, there exists at least one generalized disclosure $(V_i, \psi_i)(1 \leq i \leq n)$ such that $id \in \pi_{ID}(\sigma_{\psi_i}(baseT))$. In fact, if an ID value appears in the Public Knowledge disclosure but does not appear in any of the generalized views, there would be no information for the adversary to infer the sensitive value associated to that ID value. Therefore, the adversary can just eliminate this ID value from the Public Knowledge disclosure and potentially the original table *baseT*. This does not affect how the adversary can infer the sensitive values associated with other individuals. Consequently, this property guarantees the corresponding *possible instance set*, PIS, to be a finite set.

(2) The generalized disclosures are *well connected*, i.e., we cannot divide the set of the selections functions of the generalized disclosures into two non-empty sets, $\{\psi_{i_1}, \ldots, \psi_{i_k}\}$ and $\{\psi_{i_{k+1}}, \ldots, \psi_{i_n}\}$, such that:

$$(\vee_{j=1}^{k} \psi_{i_j}) \wedge (\vee_{j=k+1}^{n} \psi_{i_j}) = false$$

Given a micro-data disclosure problem, if the generalized disclosures do not satisfy this property, we will be able to decompose this problem into two independent problems. The given Public Knowledge disclosure *publicT* can be divided into two tables, $publicT' = \sigma_{\psi'}(publicT)$ and $publicT' = \sigma_{\neg\psi'}(publicT)$. Correspondingly, the generalized disclosures can be divided into two groups (with $\psi_i \Rightarrow \psi'$ or $\psi_i \Rightarrow \neg\psi'$), along with the two Public Knowledge tables above to form two micro-data disclosure problems, respectively.

With all of the collected information, the adversary can try to understand the sensitive values that are associated with each individual that appears in *publicT*. To formalize this process, consider the following (PIS, id, s)-guessing game. To determine whether an ID value id is associated with a sensitive value s, the adversary randomly selects a table, using uniform distribution, from the PIS with respect to a given micro-disclosure. The adversary wins the game if, in the selected table, id is associated with s. Therefore, to protect the individual's privacy, my goal is to guarantee that the adversary cannot win this (PIS, id, s)-guessing game for any id, s with a high probability.

Definition 25. *A micro-disclosure $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$ is said to be γ-Private ($0 \leq \gamma < 1$) if the possible instance set PIS with respect to Δ satisfies the following: for any id and s, an adversary cannot win the (PIS, id, s)-guessing game with a probability higher than γ. In this case, I will also say that the PIS is γ-Private.*

Recall from the medical information example discussed in the previous section that: (1) the PIS with respect to Table 6.2 (Public Knowledge) and Table 6.3(A) (generalized) is a 0.5-Private ($N > 0$), and (2) the PIS with respect to Table 6.2 (Public Knowledge) and Table 6.3(A,B) (generalized) is not.

As I show in Theorem 16, the 0.5-Private property can be regarded as an extension to the *Recursive* $(1, 2)$-*Diversity* for multiple generalized disclosures, i.e., they are equivalent when applied to the problem with a single generalized disclosure. Clearly, the 0.5-Privacy can be applied to multiple generalized disclosures, while the *Recursive* $(1, 2)$-*Diversity* cannot. It is worth noting that if we have multiple generalized disclosures such that any two of them do not intersect with each other, a property like *Recursive (1, 2)-Diversity* can be applied. However, as I have previously assumed in the well-connectivity of PIS, such a multiple generalized disclosure case can be decomposed into multiple independent cases of single generalized disclosure. Theorem 16 states the relation between two properties.

Theorem 16. *The possible instance set PIS, with respect to a Public Knowledge disclosure publicT and a single generalized disclosure (V, ψ), is a γ-Private, if and only if, V satisfies Recursive ($\frac{\gamma}{1-\gamma}$, 2)-Diversity.*

Proof. For any ID value *id* and sensitive value s that appear in *publicT* or V, the fact that V satisfies *Recursive ($\frac{\gamma}{1-\gamma}$, 2)-Diversity* guarantees that the probability of *id* to be associated with s in an output of the adversary's random oracle is less than or equal to γ. This is true because in a single view disclosure case, the possible instance set PIS can be regarded as a set of all possible permutations of sensitive values appearing in V (with a fixed order of ID values appearing in *publicT*). Therefore, among all $|PIS|$ possible instance tables, there are at most $\gamma|PIS|$ tables having a particular *id* associated with a particular sensitive value s. Therefore, PIS is γ-Private.

On the other hand, if V does not satisfy *Recursive ($\frac{\gamma}{1-\gamma}$, 2)-Diversity*, there must exist an ID value *id* and a sensitive value *id*, such that the number table among PIS having *id* associated with s is greater than $\gamma|PIS|$. Consequently, PIS cannot be a γ-Private. □

Clearly, the property of γ-Private can be regarded as an extension of the property of *Recursive* $(\frac{\gamma}{1-\gamma}, 2)$*-Diversity.* It can be understood that we can design similar private properties as the γ-Private to extend the rest of properties in the l-Diversity family ([40]) and even other privacy properties like t-Closeness ([53]). However, in this chapter, I limit my discussions to the γ-Private only.

Theorem 17. *The problem of whether a micro-disclosure $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$ is γ-Private is decidable.*

Proof. Note that because of the two properties of generalized disclosures I previously discussed, the PIS with respect to Δ is finite. Theorem 17 is then straightforward because I can verify the property of γ-Privacy through enumeration, i.e., $\forall id, s$,

$$\frac{|\{T | T \in PIS, (id, s) \in \pi_{ID,SA}(T)\}|}{|PIS|} \leq \gamma \tag{6.1}$$

$\square$

Note that the set $\{T | T \in PIS, (id, s) \in \pi_{ID,SA}(T)\}$ in (1) can also be regarded as a possible instance set PIS' with respect to a new micro-disclosure including Δ and the fact that *id* is associated with s. Recall the medical information example discussed in Section 6.1. Besides the Public Knowledge disclosure, Table 6.2, if we have two generalized disclosures (Table 6.3(A) and Table 6.3(B)), the corresponding PIS_1 will have 45 tables, among which there are 36 tables containing the relation $(Donald, SARS)$; if we have four generalized disclosures (Table 6.3(A,B) and Table 6.5(A,B)), the corresponding PIS_2 has only 8 tables, among which there are 4 tables containing the relation $(Donald, SARS)$. Therefore, PIS_2 is a 0.5-Private, while PIS_1 is not. Theorem 17 guarantees the computability of the decision problem to check the proposed γ-Private property. However, in practice, it is not always feasible to achieve this goal through enumeration.

Theorem 18. *The problem of whether a micro-disclosure* $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$ *is* γ*-Private is #P-complete.*

Sketch. I prove this by showing that it is #P-complete to compute $|PIS|$ with respect to Δ. This is true because any polynomial algorithm to verify γ-Privacy will lead to a polynomial algorithm to compute $|PIS|$ and γ-Privacy can certainly be verified by computing $|PIS|$. Clearly, computing $|PIS|$ is #P because whether a given table T is in PIS can be verified in polynomial time. To prove the hardness of computing $|PIS|$, I show that to compute $|PIS|$ even in a special case is #P-hard. Let *baseT* contain four sets of IDs, $V_1^I, V_2^I, V_3^I, V_4^I$, where $|V_1^I| = |V_3^I|$ and $|V_2^I| = |V_4^I|$. Let $(V_1, \psi_1), \ldots, (V_4, \psi_4)$ be four generalized disclosures such that $V_1 = V_2 = V_3 = V_4$ and $\pi_{ID}\sigma_{\psi_1}(publcT) = V_1^I \cup V_2^I$, $\pi_{ID}\sigma_{\psi_2}(publcT) = V_2^I \cup V_3^I$, $\pi_{ID}\sigma_{\psi_3}(publcT) = V_3^I \cup V_4^I$, $\pi_{ID}\sigma_{\psi_4}(publcT) = V_4^I \cup V_1^I$. Let $s_1, \ldots, s_{\lambda'}$ be the set of different sensitive values. To compute $|PIS|$, I have to compute how many different assignments of $n_1, \ldots, n_{\lambda'}$, where $n_i (1 \leq i \leq \lambda')$ represents the number of times that the sensitive value s_i is associated with V_1. (If n_i is fixed, the number of times that the sensitive value s_i is associated with V_2^I, V_3^I, V_4^I is also fixed.)

I reduce the problem of computing the number of contingency tables with prescribed row and column sums in the $2 \times \lambda'$ case, which is proved to be #P-complete in the work [57] on this problem.

Consider an arbitrary $2 \times \lambda'$ contingency table counting problem with variables $n_1, \ldots, n_{\lambda'}$, $n'_1, \ldots, n'_{\lambda'}$ and the corresponding constraints: $\Sigma_{i=1}^{\lambda'} n_i = a_1$, $\Sigma_{i=1}^{\lambda'} n'_i = a_2$, and $n_i + n'_i = b_i (1 \leq i \leq \lambda')$.

Let $|V_1^I| = a_1$, $|V_2^I| = a_2$, $appear(s_i) = b_i (1 \leq i \leq \lambda')$, where $appear(s_i)$ is the number of appearances of s_i in V_1. It is clear that to compute the number of different contingency tables, it is equivalent to compute the number of different assignment of $n_1, \ldots, n_{\lambda'}$ in this problem of Mirco-Disclosure. $\square$

Therefore, to verify the proposed privacy property efficiently, in a given micro-data disclosure problem with multiple generalized disclosures and a large original table, I need to seek an alternative approach.

6.3 Relaxed γ-Privacy and Its Verification by Monte Carlo Simulation

In this section, I discuss a stochastic approach to verify the proposed privacy property. By Theorem 17, I show that, in order to verify whether a given PIS is γ-Private, I have to compute the left side of inequality (1). We denote it by $p(id \sim s)$:

$$p(id \sim s) = \frac{|\{T | T \in PIS, (id, s) \in \pi_{ID,SA}(T)\}|}{|PIS|}$$

Instead of computing $p(id \sim s)$ through an enumeration process, we can estimate it with a pre-defined statistical confidence, using a Monte Carlo simulation. More formally, let $T_1, \ldots, T_N$ be N tables that are randomly sampled from the given PIS, I investigate the statistical properties of the sample mean $\bar{p}(id \sim s)$, to approach the population mean $p(id \sim s)$:

$$\bar{p}(id \sim s) = \frac{\Sigma_{i=1}^{N} f(T_i)}{N} \tag{6.2}$$

where $f(T)$ is defined as:

$$f(T) = \begin{cases} 1, & (id, s) \in \pi_{ID,SA}(T); \\ 0, & otherwise. \end{cases}$$

Correspondingly, I define a new stochastic version of the γ-Private property as follows, based on the fact that the distribution of $\bar{p}(id \sim s)$ will approximate a normal distribution with a large number N. (Note that we have $0 \leq \bar{p}(id \sim s) \leq 1$, which requires γ should not be close to either 0 or 1.)

Definition 26. *Given a micro-disclosure $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$, two constants ϵ and θ, where $0 < \epsilon, \theta < 1$, and an unbiased estimator $\bar{p}(id \sim s)$ of $p(id \sim s)$ for every id and s in Δ, I say that Δ is γ-Private with (ϵ, θ)-Confidence if for any id and s, we have*

- *$p(id \sim s) \leq \gamma$ with statistical confidence greater than or equal to θ, OR*
- *$\gamma - \epsilon \leq p(id \sim s) \leq \gamma + \epsilon$ with statistical confidence greater than or equal to θ.*

I say that Δ is non-γ-Private with (ϵ, θ)-Confidence if:

- *for any id and s, we have $p(id \sim s) > \gamma$ with statistical confidence greater than or equal to θ, AND*
- *there exist id and s, we have $\gamma - \epsilon \leq p(id \sim s) \leq \gamma + \epsilon$ with statistical confidence less than θ.*

Note that by the second condition of Definition 26, if $p(id \sim s)$ is close enough to the security bound γ (within ϵ), a satisfaction of the privacy property will be granted. The reason for having this major difference from Definition 25 is that if $p(id \sim s) = \gamma$, we will meet the problem of endless sampling when checking only the first condition. By Definition 26, we can immediately obtain the following:

Theorem 19. *Given a micro-disclosure Δ and the value of unbiased estimator $\bar{p}(id \sim s)$ of $p(id \sim s)$, if for any id, s in Δ, there exist $\gamma_1, \gamma_2 (\gamma_2 - \gamma_1 < \epsilon)$ such that $p(id \sim s) \in [\gamma_1, \gamma_2]$ with confidence θ, we have either (1) Δ is γ-Private with (ϵ, θ)-Confidence or (2) Δ is non-γ-Private with (ϵ, θ)-Confidence.*

Next, I discuss how to construct the random input generator for the Monte Carlo simulation. In the problem setting, the ideal random input generator would be the adversary's random oracle. However, in practice, it is not easy to obtain such a random oracle, because, as I have shown in the previous section, to decide a bound for the number of sensitive values that can be associated with an ID value in a given PIS is #P-hard. To mitigate this problem, I adopt a weighted sample mean using a simpler version of Markov chain sampling, instead of the regular sample mean.

For the sake of simplicity, I first discuss how to compute a weighted sample mean through Markov chain sampling based on an abstractively constructed tree structure over PIS. Then I discuss how to construct such a tree structure in practice to fulfill the entire solution.

Given a possible instance set PIS, which is finite, I construct a Tree $\mathcal{T}$ such that:

- the height of $\mathcal{T}$ is λ, where λ is the size (i.e., number of tuples) of the original table $baseT$;
- each non-leaf node has at most λ children;
- $\mathcal{T}$ has $|PIS|$ leaf nodes and each leaf node represents a different element of $|PIS|$.

I call any tree that satisfies all of the three properties above a *Constructing Tree* of the given PIS. I denote by a a node of $\mathcal{T}$, $d(a)$ the number of children of a and a_0 the root of $\mathcal{T}$. I use the following Algorithm 6.1 to select a leaf node, i.e., an element of PIS through $\mathcal{T}$.

```
Algorithm 6.1: Sampling a Table from PIS (abstraction)
1. Let $i = 0$, $w = 1$;
2. While ($a_i$ is not a leaf node) Do
3.       Randomly select(with uniform distribution) $a_{i+1}$
   from the children of $a_i$;
4.       Let $w = w * d(a_i)$;
5.       Let $i = i + 1$;
6. End While
7. Output $(w, T)$, where $T = a_i$.
```

Figure 6.1: Algorithm 6.1

I then can construct a weighted sample mean given a number N of sampled result $(w_1, T_1), \ldots, (w_N, T_N)$. Equation (6.2) can be rewritten as follows:

$$\bar{p}(id \sim s) = \frac{\Sigma_{i=1}^{N} f(T_i) w_i}{\Sigma_{i=1}^{N} w_i} \tag{6.3}$$

Theorem 20. *$\bar{p}(id \sim s)$ defined in (3) is an unbiased estimator of $p(id \sim s)$.*

Proof. This result can be obtained easily by the fact that the probability of a given $T_i (1 \leq i \leq N)$ to be selected by Algorithm 6.1 is $1/w_i$. □

By Liapounov's Central Limit Theorem ([58]), the probability distribution of $\bar{p}(id \sim s)$ approaches a normal distribution. We then can decide whether a given micro-disclosure is

γ-Private with (ϵ, θ)-Confidence or non-γ-Private with (ϵ, θ)-Confidence in polynomial time by means of the following result.

Theorem 21. *Given a micro-disclosure Δ, and the corresponding PIS and its constructing tree $\mathcal{T}$, it is sufficient to collect $\lceil (\frac{\alpha}{\epsilon})^2 \rceil$ samples (by Algorithm 6.1) to decide whether Δ is γ-Private with (ϵ, θ)-Confidence or non-γ-Private with (ϵ, θ)-Confidence, where $\alpha = 2\sqrt{2}erf^{-1}(\theta)$ and $erf()$ is the Gauss Error Function.*

Proof. For every id and s in Δ, the variance estimator of $\bar{p}(id \sim s)$ can be computed and bounded as:

$$\nu^2 = \frac{1}{N(\Sigma_{i=1}^{N}|w_i| - 1)} \Sigma_{i=1}^{N} |w_i| (f(T_i) - \bar{p}(id \sim s))^2 < \frac{1}{N} \leq (\frac{\epsilon}{\alpha})^2$$

This is true because, in this case, when $N \geq 4$ (which should be true), we have:

$$\frac{1}{(\Sigma_{i=1}^{N}|w_i| - 1)} \Sigma_{i=1}^{N} |w_i| (f(T_i) - \bar{p}(id \sim s))^2 < 1$$

where for all $0 \leq 1 \leq N$, $w_i > 1$ and $f(T_i)$ values either 0 or 1. Therefore, we have:

$$\nu < \frac{\epsilon}{\alpha} = \frac{\epsilon}{2\sqrt{2}erf^{-1}(\theta)} \tag{6.4}$$

Note that the bound $\frac{\epsilon}{2\sqrt{2}erf^{-1}(\theta)}$ is independent from the (id, s) pair. Based on the properties of normal distribution, we have:

$$p(id \sim s) \in [\bar{p}(id \sim s) - \sqrt{2}erf^{-1}(\theta)\nu, \bar{p}(id \sim s) + \sqrt{2}erf^{-1}(\theta)\nu]$$

with a statistical confidence θ. Based on Equation (6.4), we have:

$$(\bar{p}(id \sim s) + \sqrt{2}erf^{-1}(\theta)\nu) - (\bar{p}(id \sim s) - \sqrt{2}erf^{-1}(\theta)\nu) < \epsilon.$$

Theorem 19 completes this proof. □

Note that Theorem 21 shows that a constant number of samplings is enough to verify the satisfaction of the proposed privacy property by a given micro-disclosure. However, we have to be aware of the computational complexity of the sampling process itself. Clearly, we cannot construct entirely a *constructing tree* of the PIS with respect to the micro-disclosure, because the number of nodes of a *constructing tree* can be exponential to λ, the size of *baseT*. Next, I discuss how to execute *Algorithm* 6.1 efficiently, more specifically, in $O((\lambda\lambda')^4)$ time, through a *constructing tree* $\mathcal{T}$ without actually constructing it.

The Sampling Process

How to efficiently sample a possible solution to a combinatorial problem where exact counting is #P-hard has been well studied (e.g., [57,59]). The work [57] shows how to uniformly sample a contingency table. In this section, I show that the sampling can be done efficiently in the problem.

First, I introduce some preliminary notations. Given a micro-disclosure $\Delta = ($*publicT*, $\{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$, I call the following linear equation system with constraints the *solver* of Δ, denoted by $\mathcal{SOL}(\Delta)$.

Let $s_1, \ldots, s_{\lambda'}$ be the complete list of different sensitive values appearing in Δ, and naively, $\lambda' \leq \lambda$. Let $id_1, \ldots, id_\lambda$ be the complete list of ID values appearing in Δ. Let $X^{\lambda\lambda'} = \{x_{i,j}\}, (1 \leq i \leq \lambda, 1 \leq j \leq \lambda')$ be the variable representing whether an ID value is associated with a sensitive value, i.e., $x_{i,j} = 1$ means that id_i is associated with s_j and $x_{i,j} = 0$ otherwise. Therefore, any possible instance table in the PIS with respect to Δ can be represented by a different assignment of $X^{\lambda\lambda'}$ (an assignment can be regarded as a vector of size $\lambda\lambda'$). The $\mathcal{SOL}(\Delta)$ includes the following two kinds of equations, based on the requirements of the given micro-disclosure:

- $\forall i (1 \leq i \leq \lambda), \Sigma_{j=1}^{\lambda'} x_{i,j} = 1$. That is, each ID value can be associated to only one sensitive value in one possible instance table;

- $\forall k, j (1 \leq k \leq n, 1 \leq j \leq \lambda'), \Sigma_{id_i \in \pi_{ID}\psi_k(publicT)} x_{i,j} = app(V_k, s_j)$, where $app(V_k, s_j)$ represents the number of times s_j appears in V_k. That is, a possible instance table must yield the same result as *baseT* for any generalized disclosure.

- Note that in $\mathcal{SOL}(\Delta)$, I assume each variable $x_{i,j}, (1 \leq i \leq \lambda, 1 \leq j \leq \lambda')$ can be any real number within the interval $[0, 1]$.

Theorem 22 follows easily from the properties of micro-disclosure and the proof of Theorem 16 in the work [57].

Theorem 22. *Given a micro-disclosure Δ, its solver $\mathcal{SOL}(\Delta)$ defines a polyhedron, of which every vertex $A = (a_{1,1}, \ldots, a_{\lambda,\lambda'})$ is an integral point, i.e., $a_{i,j} = 0 \vee a_{i,j} = 1, (1 \leq i \leq \lambda, 1 \leq j \leq \lambda'$.*

Sketch. Assume there is a vertex represented by an assignment vector A (of $X^{\lambda\lambda'}$), in which there exists $0 < a_{i_1,j_j} < 1$. Consider all $a_{i,j}$ in A such that $a_{i,j} = 1$ or $a_{i,j} = 0$, assume the corresponding variable $x_{i,j} = a_{i,j}$ to be a constant in $\mathcal{SOL}(\Delta)$. I now reduce to a polyhedron with smaller dimension, where the projection A' of A should still be a vertex. However, every constant coefficient of this new linear equation system is an integer and every variable coefficient is 1, therefore, either (1) there exists a small vector $\bar{\epsilon}$ such that $A' - \bar{\epsilon}$ and $A' + \bar{\epsilon}$ are both in the polyhedron, or (2) A' is the only solution. However, (1) is contradicted to the fact that A' is a vertex, while (2) is contradicted to the micro-disclosure Δ itself because of the existence of the *baseT*, i.e, *baseT* (with all variables to be 0 or 1) is always a solution to $\mathcal{SOL}(\Delta)$. This step completes the proof. $\square$

I describe the sampling process by Algorithm 6.2. In it, I denote by T a table over 2 attributes for *id* and s. I denote by $\Delta \boxplus T$ the new micro-disclosure in which both Δ and T are disclosed. Note that T can be also regarded as a generalized disclosure where nothing is indeed generalized.

Theorem 23. *Given a micro-disclosure Δ and its corresponding PIS, (1) Algorithm 6.2 has the same functionality as Algorithm 6.1 through a constructing tree of the PIS; (2)*

Algorithm 6.2: Sampling a table from PIS (instantiation)

```
1.  Let $T = \phi, w = 1$;
2.  For $i = 1$ to $\lambda$ do
3.      Let $S = \phi$;
4.      For $j = 1$ to $\lambda'$ do
5.          If $\mathcal{SOL}(\Delta \boxplus (T \cup \{(id_i, s_j)\}))$ is not solvable
6.              Continue;
7.          End If;
8.          Find a vertex $A$ of the resulting polyhedron;
9.          If $A$ is not an integral point
10.             Continue;
11.         End If;
12.         $S = S \cup \{s_j\}$;
13.     End For;
14.     Let $w = w * |S|$;
15.     Random select $s$ from $S$;
16.     Let $T = T \cup \{(id_i, s)\}$;
17. End For;
18. Output $(w, T)$;
```

Figure 6.2: Algorithm 6.2

Algorithm 6.2 generates an output in $O((\lambda\lambda')^4)$ time, where λ is the size of baseT and λ' the number of different sensitive values appeared in baseT.

Proof. Part (1) is clear because for any $T \in PIS$, it must be contained in a possible output (w, T) of Algorithm 6.2. Also, based on Theorem 22, in every output (w, T) of Algorithm 6.2, we know that $T \in PIS$ and $1/w$ is the probability of T to be output by Algorithm 6.2.

For (2), note that it cost $O((\lambda\lambda')^3)$ to solve a linear equation system, and finding a vertex of a polyhedron can be done in $O(\lambda\lambda')$, where $\lambda\lambda'$ is the number of variables (dimension). Therefore, Algorithm 6.2 will output in $\lambda\lambda' O((\lambda\lambda')^3 + \lambda\lambda') = O((\lambda\lambda')^4)$ time. $\square$

Note that in Algorithm 6.2 (line 12), if the vertex point that we found is an integral point, there exists a possible instance table consistent with the disclosures. That is, s_j is a valid choice for id_i. Theorem 21 says that for a constant ϵ and θ, we need a constant number of samplings to be able to verify a γ-Privacy with (ϵ, θ)-Confidence. With Theorem 21 and

Theorem 23, we have the following result:

Theorem 24. *Whether a micro-disclosure of a table baseT is γ-Private with (ϵ, θ)-Confidence or non-γ-Private with (ϵ, θ)-Confidence is decidable in $O(\lceil \frac{2\sqrt{2}erf^{-1}(\theta)}{\epsilon} \rceil (\lambda\lambda')^4)$ time, where λ is the size of baseT, λ' the number of different sensitive values appeared in baseT and* erf *the Gauss Error function.*

6.4 Restoring Compromised Privacy

In this section, I discuss how to restore compromised privacy, given a micro-disclosure $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$, which violates the desired γ-Privacy or γ-Privacy with (ϵ, θ)-Confidence by releasing additional generalized disclosures. That is, we would like to find a new micro-disclosure $\Delta' = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n), (V_{n+1}, \psi_{n+1}), \ldots, (V_{n+K}, \psi_{n+K})\})$ which extends Δ, such that Δ' satisfies the desired privacy property. To do that, we need to answer the following two questions:

- How many additional generalized disclosures do we want to disclose?
- What generalized disclosures should be disclosed?

6.4.1 Selecting an Additional Generalized Disclosure

Besides the generalized disclosures that have been disclosed, we can construct any generalized disclosure which by itself satisfies the desired privacy property. Note that the original table *baseT* has a limited size λ. Therefore, we have at most $2^\lambda - (\lambda + 1)$ different choices of additional disclosures. The problem of whether there exists one of them such that we can restore the privacy by disclosing it is thus decidable. However, whether there exists an efficient algorithm to decide (for γ-Privacy with (ϵ, θ)-Confidence) is still an open research question. Note that enumeration is not computationally practical, because, in the worst case, we can have $2^\lambda - (\lambda + 1) - n$ different candidates.

In this section, I propose heuristic algorithms in which additional generalized disclosures are restricted to the refinements of the previously disclosed ones. To illustrate, I extend the medical information example from Section 6.1. The original table *baseT* is shown in Table 6.6.

Table 6.6: Extended Patient Information Table

Name	Sex	Age	Employer	Condition
Alan	M	23	ABC, Inc.	Heart Disease
Bob	M	24	ABC, Inc.	SARS
Clark	M	25	ABC, Inc.	Viral Infection
Donald	M	26	ABC, Inc.	SARS
Ellen	F	27	ABC, Inc.	Viral Infection
Fen	F	28	ABC, Inc.	SARS
Garcia	F	28	ABC, Inc.	Flu
Helen	F	29	ABC, Inc.	SARS
Jessica	F	30	ABC, Inc.	Viral Infection
Kathy	F	30	ABC, Inc.	Heart Disease
Lucy	F	33	ABC, Inc.	Viral Infection

Besides the Public Knowledge disclosure, we have three released generalized disclosures. Two of them are exactly the same two views shown in Table 6.3(A) and Table 6.3(B). The third generalized disclosure is shown in Table 6.7.

Table 6.7: The Third Generalized Disclosure

Age	Condition
29~33	SARS
29~33	Viral Infection
29~33	Viral Infection
29~33	Heart Disease

Note that in this example we will have the corresponding possible instance set PIS that is not well-connected, as discussed in previous sections. Assume that, as in Section 6.1, we consider 0.5-Privacy. It can be computed that in this example, the adversary will win the $(PIS, Donald, SARS)$-guessing game with a probability higher than 0.5, which violates the 0.5-Privacy.

To restore the compromised privacy, one possibility is to disclose Table 6.5(A) and Table 6.5(B) as discussed in Section 1. Another possibility, in this example, is to disclose the following generalized disclosure shown in Table 6.8, together with either Table 6.5(A) or Table 6.5(B) to restore the compromised privacy.

Table 6.8: Another Additional Generalized Disclosure

Age	Condition
26 or 33	SARS
26 or 33	Viral Infection

However, from the information consumer's point of view, Table 6.8 is a worse choice than any of the Table 6.5(A) or Table 6.5(B), because the latter two provide the desired information in a more precise way. That is, Table 6.5(A) is a refinement of Table 6.3(A) and Table 6.5(B) is a refinement of Table 6.3(B).

Therefore, in practice, it may make sense to limit the choices of additional generalized views to such a category. Following this restriction, in this chapter, I consider an additional generalized disclosure selected from the set $\mathcal{R}(\Delta)$, where $\mathcal{R}(\Delta)$ is called a single refinement set: for a given micro-disclosure $\Delta = (publicT, \{(V_1, \psi_1), \ldots, (V_n, \psi_n)\})$, it is computed as $\mathcal{R}(\Delta) = \{(V, \psi) | \exists i (1 \leq i \leq n), \sigma_\psi(baseTbl) = \sigma_\psi \sigma_{\psi_i}(baseT)\}$, where ψ is a propositional formula without *negation, disjunction*, and "$\neq$". Therefore, we have $|\mathcal{R}(\Delta)| \leq n\lambda^2 \leq \lambda^3$, where $\lambda = |publicT|$.

6.4.2 Selecting the Number of Additional Disclosures

Next, we need to decide how many additional generalized disclosures we need to release in order to restore a compromised privacy. Note that, because the original table *baseT* has the size λ, we at most disclose λ additional generalized disclosures. Therefore, we have up to $\binom{|\mathcal{R}(\Delta)|}{1} + \cdots + \binom{|\mathcal{R}(\Delta)|}{\lambda}$ different choices, which are computationally infeasible to enumerate.

Unfortunately, for any constant K, we can construct an example, in which, in order to restore the compromised privacy, we need to disclose exact K additional generalized disclosures from $\mathcal{R}(\Delta)$, as follows.

Let the *baseT* contain $3K + 1$ ID values $id_1, \ldots, id_{3K+1}$ and $K + 2$ different sensitive values $s_1, \ldots, s_{K+3}$, in which for any $i (i \not\equiv 0 \ mod \ 3, i \neq 3K + 1)$, id_i is associated with $s_{i\ mod\ 3}$; for any $i (i \equiv 0\ mod\ 3)$, id_i is associated with $s_{i/3+2}$; id_{3K+1} is associated with s_1. We have K generalized disclosures $(V_1, \psi_1), \ldots, (V_K, \psi_K)$, where $\pi_{ID}\sigma_{\psi_i}(baseT) = \{id_{3(i-1)+1}, id_{3(i-1)+2}, id_{3(i-1)+3}, id_{3K+1}\}, (1 \leq i \leq K)$. It can be computed that the corresponding micro-disclosure Δ is not 0.5-Private, because the probability for an adversary to win the (PIS, id_{3K+1}, s_1)-guessing game is $\frac{2^K}{2^K+1}$.

In this case, the only way to restore the 0.5-Privacy is to disclose K additional generalized views, each containing two tuples from a single previously disclosed view and satisfying 0.5-Privacy by itself.

6.4.3 A Naive Efficient Algorithm To Restore Privacy

Based on the discussion above, I design a naive algorithm to restore privacy, given a micro-disclosure Δ that violated a desired γ-Privacy with (ϵ, θ)-Confidence, as follows: given a constant parameter K, the algorithm simply enumerates all possible 1-combinations, 2-combinations, ..., K-combinations of $\mathcal{R}(\Delta)$, the single refinement set of Δ, and outputs the first possible set of generalized disclosures that leads to a new micro-disclosure that is γ-Private with (ϵ, θ)-Confidence or outputs "fail" otherwise. Clearly, this naive algorithm will stop in $O(\lambda^{3K+4}\lambda'^4)$ time, where λ is the number of ID values in the original table and λ' is the number of sensitive values. I will show that, through experiments in Section 6.5, in practice we can have a good percentage of successful privacy restoring by this naive algorithm, even with $K = 1$.

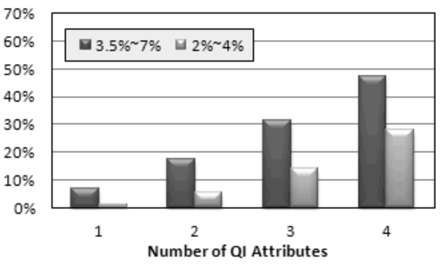


Figure 6.3: Privacy Restoring by a Single View

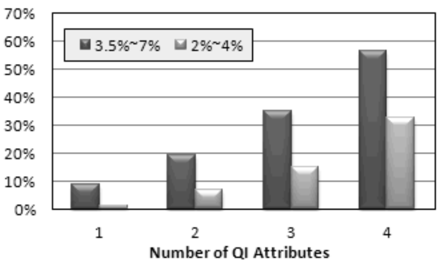


Figure 6.4: Privacy Restoring by up to 2 Views

6.5 Experiments

I conducted preliminary experiments based on the heart disease records datum borrowed from the UCI data repository ([1]). Each of the four repository tables has such attributes as person's *name, age*, and *level*, which classifies a heart disease into 5 categories. I consider the attribute *level* as sensitive and select 4 other attributes including age as the QI attributes. Each time, I consider one of the four repository tables, each having around 200 tuples. Queries in the study are selections on a single QI attribute that require the value to be in an interval $[a, b]$. The interval is selected randomly, with a selected uniformly from the interval range of the attribute, and $b - a$ chosen uniformly (1) from 3.5% to 7% or (2) from 2% to 4% of the size of the attribute range. If a query by itself violates 0.5-Privacy, it is discarded. If not, I check if all queries so far collectively satisfy 0.5-Privacy with (ϵ, θ)-confidence, where $\epsilon \leq 0.1$ and $\theta \geq 0.95$, and stop the iterative process if they do not.

I then check whether it is possible to restore the violated 0.5-Private with (ϵ, θ) confidence. Figure 6.3 refers to the case when I use a single additional view to try and restore privacy. The set of all views considered are all possible refinements (in terms of the interval in the selection condition) of previously disclosed views. Note that because all QI attributes in the study are integers, there is a finite number of considered views. Figure 6.3 gives the percentage of cases *(in* 1000 *runs)*, in which privacy was restored for the cases of 1,2,3, and 4 QI attributes, and for both ranges (1) from 3.5% to 7% (2) from 2% to 4% used in the original views. Figure 6.3 shows that the percentage of restored cases grows to about 50% when there are 4 QI attributes.

Figure 6.4 gives the same information as Figure 6.3 for the case when I use up to two (rather than one) additional views to restore privacy. For this case, Figure 6.4 shows that the percentage of restored cases grows to about 60% when there are 4 QI attributes.

Note that when the privacy was not restored, we did not know if the restoration was indeed not feasible. It may be feasible if we disclose more flexible additional views (not just the refinements of the previously disclosed ones) and/or disclose a larger number of additional views (not just 1 or 2). Clearly, the percentage of restored cases out of all restorable cases is higher or equal to that in the diagram. Note that while the study is preliminary, it indicates that the percentage of restored cases is significant, even with a single additional disclosed view.

6.6 Discussion

In this section I discuss and provide more justification for the assumptions that are made for the adversary model. Recall that the definition of the γ-Private is based on the assumption that the adversary will try to compromise an individual's privacy contained in a private table through a random guessing game. What is hidden is that we assume the adversary will collect all of the disclosed information, including Public Knowledge disclosure and generalized disclosures of the private table. The following two assumptions have not been addressed: (1) the adversary may acquire only part of the published information; and (2) the adversary may acquire even more information about the privacy restoring process. In this section, I discuss, in two different cases, how the process of privacy restoring can be affected by (2).

Intention of Privacy Restoring: The adversary may get to know the intention of trying to restore a compromised privacy. In this case, to safely restore a compromised privacy, in a given micro-data disclosure problem, whether to perform a privacy restoring should not depend on the original *baseT*. Formally, the decision should be "simulatable" to the adversary. To explain the idea, consider a modified example of medical information disclosure that has been discussed in Section 6.1. Assume that we have a different *baseT* shown in Table 6.9, but the Public Knowledge disclosure (Table 6.2) and the two generalized disclosures (Tables 6.3(A, B)) remain the same. Note that in this example, the adversary may be misled by the disclosed information and find out that there is a high probability of *Donald* having *SARS*. One may think that it could be a good choice to stay with such a status. However, this is not a safe decision, because if the adversary acquires the knowledge that we have been trying to restore a compromised privacy, the fact that we have done nothing in this example will immediately reveal the fact that *Donald* has *Viral Infection*. This problem can be solved simply by making the privacy restoring decision independent from the original *baseT*.

Table 6.9: Modified Patient Information Table

Name	**Sex**	**Age**	**Employer**	**Condition**
Alan	M	23	ABC, Inc.	SARS
Bob	M	24	ABC, Inc.	SARS
Clark	M	25	ABC, Inc.	Heart Disease
Donald	M	26	ABC, Inc.	Viral Infection
Ellen	F	27	ABC, Inc.	SARS
Fen	F	28	ABC, Inc.	SARS
Garcia	F	28	ABC, Inc.	Flu

Preferences in Disclosure Selection: The adversary may obtain the following two facts: (1)when we are trying to restore a compromised privacy, we may have some preferences in the selection of additional generalized disclosures; and (2) among all the released generalized disclosures, some are used to restore a compromised privacy. In this case, the adversary may also have an opportunity to compromise an individual's privacy immediately. To illustrate, consider the same example shown in Table 6.9. To restore the compromised privacy for *Donald* as discussed above, we may disclose two additional generalized disclosures: one is the same as shown in Table 6.5(B) and the other one is shown in Table 6.10(A). This works fine under the assumptions discussed in previous sections. However, if the adversary discovers that we are trying to minimize generalization for the additional disclosure, the fact that Table 6.10(B) is not disclosed as the additional generalized disclosure will immediately reveal that *Clark* is highly probable to have *Heart Disease*. This is true because if *Donald* has *Viral Infection* and *Clark* has *SARS*, Table 6.10(B) should be disclosed instead of Table 6.10(A). Therefore, under such strong assumptions of the adversary model, we should be very careful to select an additional generalized disclosure. A complete solution for this case remains future work.

Table 6.10: Additional Disclosure Selection

Age	**Condition**
24 or 26	Viral Infection
24 or 26	SARS

(A)

Age	**Condition**
25 $\sim$ 26	Viral Infection
25 $\sim$ 26	SARS

(B)

6.7 Summary

In this chapter, I studied the problem of restoring compromised privacy for micro-data disclosure with multiple disclosed views by disclosing more views, in terms of both theoretical foundation and practical solutions. Many research questions remain open. One is a more comprehensive study on heuristic algorithms to restore privacy in terms of their complexity and efficacy. Another direction is the notion of optimality, i.e., given multiple ways to restore privacy, how do we decide on the best. Also important is extending the results to additional, possibly more relaxed, adversary models.

Chapter 7: Conclusion and Future Research

7.1 Conclusion

This dissertation presents a number of techniques that address a basic question in the problem of information sharing: how secure is an information sharing process, including the applied information disclosure algorithms, and if it is not, how to make it secure? Or more intuitively, what information does an applied information disclosure algorithm really disclose with respect to a particular input?

This problem has been overlooked by the research community for a long time. However, it is extremely important to all data applications that deal with the information disclosure problem. In any case that the information providers do not carefully address this problem, serious security/privacy violation may take place.

This dissertation discusses the problem in the environment of two major information sources, numerical databases and relational category databases (micro-data).

For numerical data disclosure applications, this dissertation first studies the existing solution to a similar problem proposed in the work of simulatable auditing. This dissertation shows that the existing solution does have a security guarantee but far from the best with respect to data utility.

This dissertation then proposes a new model, called simulatable binding. Based on which, this dissertation develops several algorithms for popular queries that are proved to have the locally-best performances.

For category data disclosure applications, this dissertation first studies the existing solutions to define the proper notion of security/privacy properties. This dissertation then shows that the existing notions only guarantee the declared properties when the applied disclosure algorithms remain secret to anyone other than the information providers. This assumption is, however, too strong to be practical.

This dissertation then proposes a general notion, called p-safety, where p is any security/privacy constraint the information providers want to assure. Any algorithm that guarantees the p-safety will at the same time guarantee "p" when the algorithm itself is known to the public.

However, this dissertation proves that, in general (and most cases in practice), algorithms that guarantee the proposed p-safety property cannot optimize their results with respect to the data utility. That is, it is an NP hard problem.

This dissertation gives a discussion in detail of how to design an efficient heuristic algorithm such that it guarantees the p-safety property and can produce outputs that are close to the optimal ones with respect to the inputs.

In the end, this dissertation discusses the problem of protecting security/privacy when multiple information providers are disclosing the same set of information without collaboration or centralized control. It is shown that under such conditions, the security/privacy constraint of any information provider is not guaranteed. This dissertation opens another way to solve this problem. That is, under certain conditions, the violated security/privacy constraints of the information providers can be restored by disclosing more information. This is then able to be done by any third party information providers in a truly distributed fashion.

7.2 Future Research

There are still a number of problems that have not been solved and are open to future research. In the numerical databases, when the information providers have to answer comprehensive categories of aggregation queries from the information consumers, how to design correspondingly efficient disclosure algorithms based on the model of simulatable binding is still open. This dissertation only solves the problem for a limited family of queries.

In the problem of micro-data disclosure, the way to approach the optimal algorithms with respect to data utility is never ending. Although the heuristic algorithms designed in the dissertation produce outputs with relative good performances in general, there are still data applications with high data utility requirements that need more specially well-designed disclosure algorithms.

In a more general range, how to secure the information disclosure process deserves to be discussed in all kinds of data applications related to information disclosure or information sharing. For example, in the model network environments, information is more likely to be disclosed in streaming or a more distributed fashion. How to extend the real “safe” information disclosure framework discussed in this dissertation into these problem settings is very challenging and important.

Bibliography

Bibliography

[1] A. Asuncion and D. Newman, "UCI machine learning repository (Data Provider: Andras.Janosi, hungarian institute of cardiology; William.Steinbrunn, university hospital, zurich, switzerland; Matthias.Pfisterer, university hospital, basel, switzerland; Robert.Detrano, v.a. medical center, long beach and cleveland clinic foundation.)," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[2] N. Adam and J. Wortmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515–556, 1989.

[3] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of ACM PODS*, 2003, pp. 202–210.

[4] C. Dwork and K. Nissim, "Privacy-preserving data mining on vertically partitioned databases," in *Proceedings of CRYPTO*, 2004, pp. 528–544.

[5] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *Proceedings of ACM PODS*, 2005, pp. 128–138.

[6] S. Warner, "Randomized response: A survey technique for eliminating error answer bias," *Journal of American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[7] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM SIGMOD*, 2000, pp. 439–450.

[8] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *Proceedings of ACM PODS*, 2003, pp. 211–222.

[9] R. Agrawal, R. Srikant, and D. Thomas, "Privacy-preserving olap," in *SIGMOD*, 2005, pp. 251–262.

[10] N. Mishra and M. Sandler, "Privacy via pseudorandom sketches," in *Proceedings of ACM PODS*, 2006, pp. 143–152.

[11] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "Auditing boolean attributes," *Journal of Computer and System Sciences*, vol. 66, no. 1, pp. 244–253, 2003.

[12] F. Chin and G. Ozsoyoglu, "Auditing for secure statistical databases," in *Proceedings of ACM'81 conference*, 1981, pp. 53–59.

[13] J. B. Kam and J. D. Ullman, "A model of statistical database and their security," *ACM TODS*, vol. 2, no. 1, pp. 1–10, 1977.

[14] F. Chin, "Security problems on inference control for sum, max, and min queries," *Journal of ACM*, vol. 33, no. 3, pp. 451–464, 1986.

[15] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant, "Auditing compliance with a hippocratic database," in *Proceedings of ACM VLDB*, 2004, pp. 516–527.

[16] D. Dobkin, A. K. Jones, and R. J. Lipton, "Secure databases: protection against user influence," *ACM TODS*, vol. 4, no. 1, pp. 97–106, 1979.

[17] S. P. Reiss, "Security in databases: A combinatorial study," *Journal of ACM*, vol. 26, no. 1, pp. 45–57, 1979.

[18] J. Biskup and P. A. Bonatti, "Controlled query evaluation for known policies by combining lying and refusal," *Annals of Mathematics and Artificial Intelligence*, vol. 40, no. 1-2, pp. 37–62, 2004.

[19] K. Kenthapadi, N. Mishra, and K. Nissim, "Simulatable auditing," in *Proceedings of ACM PODS*, 2005, pp. 118–127.

[20] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards robustness in query auditing," in *ACM VLDB*, 2006, pp. 151–162.

[21] A. Dobra and S. E. Feinberg, "Bounding entries in multi-way contingency tables given a set of marginal totals," in *Foundations of Statistical Inference: Proceedings of the Shoresh Conference 2000.* Springer Verlag, 2003.

[22] A. Slavkovic and S. E. Feinberg, "Bounds for cell entries in two-way tables given conditional relative frequencies," *Privacy in Statistical Databases*, 2004.

[23] I. P. Fellegi, "On the question of statistical confidentiality," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 7–18, 1993.

[24] L. H. Cox, "Solving confidentiality protection problems in tabulations using network optimization: A network model for cell suppression in the u.s. economic censuses," in *Proceedings of the Internatinal Seminar on Statistical Confidentiality.* International Statistical Institute, Dublin, 1982, pp. 229–245.

[25] L. H.Cox, "New results in disclosure avoidance for tabulations," in *International Statistical Institute Proceedings of the 46th Session.* Tokyo, 1987, pp. 83–84.

[26] G. T. Duncan and S. E. Feinberg, "Obtaining information while preserving privacy: A markov perturbation method for tabular data," in *Joint Statistical Meetings.* Anaheim,CA, 1997.

[27] P. Diaconis and B. Sturmfels, "Algebraic algorithms for sampling from conditional distributions," *Annals of Statistics*, vol. 1, pp. 363–397, 1998.

[28] T. Dalenius and S. Reiss, "Data swapping: A technique for disclosure control," *Journal of Statistical Planning and Inference*, vol. 6, pp. 73–85, 1982.

[29] L. H. Cox, “Suppression, methodology and statistical disclosure control,” *Journal of the American Statistical Association*, vol. 90, pp. 1453–1462, 1995.

[30] G. Miklau and D. Suciu, “A formal analysis of information disclosure in data exchange,” in *ACM SIGMOD*, 2004.

[31] P. Samarati, “Protecting respondents’ identities in microdata release,” in *IEEE TKDE*, 2001, pp. 1010–1027.

[32] A. Meyerson and R. Williams, “On the complexity of optimal k-anonymity,” in *ACM PODS*, 2004.

[33] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee, “Toward privacy in public databases,” in *Theory of Cryptography Conference*, 2005.

[34] X. Xiao and Y. Tao, “Personalized privacy preservation,” in *ACM SIGMOD*, 2006.

[35] J. Byun and E. Bertino, “Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges,” *SIGMOD Rec.*, vol. 35, no. 1, pp. 9–13, 2006.

[36] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, “k-anonymity: Algorithms and hardness,” *Technical report, Stanford University*, 2004.

[37] R. J. Bayardo and R. Agrawal, “Data privacy through optimal k-anonymization,” in *IEEE ICDE*, 2005.

[38] K. LeFevre, D. DeWitt, and R. Ramakrishnan, “Incognito: Efficient fulldomain k-anonymity,” in *ACM SIGMOD*, 2005.

[39] Y. Du, T. Xia, Y. Tao, D. Zhang, and F. Zhu, “On multidimensional k-anonymity with local recoding generalization,” 2007.

[40] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in *IEEE ICDE*, 2006.

[41] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *IEEE ICDE*, 2007.

[42] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei, “Minimality attack in privacy preserving data publishing,” in *ACM VLDB*. VLDB Endowment, 2007, pp. 543–554.

[43] C. Yao, X. S. Wang, and S. Jajodia, “Checking for k-anonymity violation by views,” in *ACM VLDB*. VLDB Endowment, 2005, pp. 910–921.

[44] Y. Li, L. Wang, X. Wang, and S. Jajodia, “Auditing interval-based inference,” in *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, 2002, pp. 553–567.

[45] D. Kifer and J. Gehrke, “Injecting utility into anonymized datasets,” in *ACM SIGMOD*, 2006.

[46] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," *Technical report, CMU, SRI*, 1998.

[47] L. Sweeney, "Uniqueness of simple demographics in the u.s. population," in *LIDAP-WP4, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA*, 2000.

[48] A. Kerckhoffs, "La cryptographie militaire," *Journal des Sciences Militaires*, vol. 9, pp. 5–38, 1883.

[49] K. Wang and B. C. Fung, "Anonymizing sequential releases," in *ACM SIGKDD*. New York, NY, USA: ACM, 2006, pp. 414–423.

[50] T. Truta and B. Vinay, "Privacy protection: p-sensitive k-anonymity property," in *Data Engineering Workshop 2006*.

[51] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *IEEE ICDE*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 217–228.

[52] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *IEEE ICDE*. Washington, DC, USA: IEEE Computer Society, 2006, p. 25.

[53] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *IEEE ICDE*, 2007.

[54] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM SIGMOD*, May 2000, pp. 439–450.

[55] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS*, 1995.

[56] P. W. P. J. Grefen and R. A. d. By, "A multi-set extended relational algebra - a formal approach to a practical issue," in *IEEE ICDE*, 1994, pp. 80–88.

[57] M. Dyer, R. Kannan, and J. Mount, "Sampling contingency tables," in *CCC*, 1997, pp. 487–506.

[58] M. Mether, "The history of the central limit theorem," *Sovelletun Matematiikan erikoistyöt*, vol. Mat-2, no. 108, 2003.

[59] A. Bertoni, M. Goldwurm, and M. Santini, "Random generation and approximate counting of ambiguously described combinatorial structures," in *STACS*, 2000, pp. 567–580.

Curriculum Vitae

Lei Zhang has been a PhD student in the Center for Secure Information Systems at George Mason University (GMU) since 2004. His major is Information Technology. Before joining GMU, he received a B.E. degree in Computer Science and Engineering from Tsinghua University, China, in 2001, and a M.E. degree in Computer Science and Engineering from Tsinghua University, China, in 2004. His research focuses on information security, privacy, and databases.