SIMULATABLE AUDITING IN MICRO-DATABASES

LI DENG

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS

SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

OCTOBER 2010

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Li Deng

Entitled: Simulatable Auditing in Micro-Databases

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

Master of Applied Science in Information Systems Security

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

Signed by the final examining committee:

Anjali Awasthi Chair

Benjamin Fung Examiner

Yuhong Yan Examiner

Lingyu Wang Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____
Dean of Faculty

Date Oct 21, 2010

Abstract

Simulatable Auditing in Micro-Databases

Li Deng

How to protect individuals' privacy while releasing microdata tables for analysis purposes has attracted significant attention. We study the case where different microdata tables generalized over the same underlying secret table may be released upon users' queries. To satisfy privacy constraints, an auditing system must determine whether the next query can be safely answered based on the history of answered queries. However, when answering a new query is not safe, denying it may not be, either, since a denial itself may still convey some sensitive information to the user. We first model this issue in the context of releasing microdata tables. Inspired by the Simulatable Auditing technique in statistical databases, we propose a safe strategy for auditing queries that ask for microdata tables generalized over secret tables. The strategy can provide provably safe answers and good data utility. We also study how to efficiently maintain the history of answered queries for the auditing purpose. To the best of our knowledge, this is the first study on the simulatable auditing issue of microdata queries.

Acknowledgments

I would like to express my heartfelt appreciation to many people who provide help for this thesis in many ways:

First of all, I would like to express my sincerely thanks to my advisor, Dr. Lingyu Wang for providing tremendous guidance and support. He has always been a source of inspiration and motivation for me. He has excellent insightful view at suggesting research problems that are well motivated and also theoretically challenging. My interactions with him have greatly shaped my perspectives about research. I am especially thankful for his encouragement during my initial level and his support during the final level.

I would also like to thank my family and friends for the support they provided me through my entire life. They are my ultimate source of joy and inspiration.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the study.

Li Deng

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The issue of privacy-preservation in data publication has drawn significant attention lately. There are two fundamentally conflicting goals in such a data publication application: Data utility and privacy. On one hand, many organizations need to give datasets to third parties for analysis or research purposes. On the other hand, such organizations need to protect individuals' private information contained in such datasets. For example, a healthcare organization may share with researchers their databases of medical records but the organization is obliged to keep the privacy of its patients. An ideal solution in such scenarios should maximize the data utility while protecting individuals' privacy. In this thesis, we address this issue by providing models and strategies that allow analysts to query a dataset while protecting individuals' private information contained in the dataset.

Privacy issues exist in many forms of data publications. First, private data may be published as the so-called *micro-data* tables in which explicit identifiers have been removed and other identifying values have been generalized with coarser-grained values. Nonetheless, individuals may be re-identified from such a micro-data table by linking it to other data sources. For example, as shown in [45], anonymized micro-data tables of medical records from a hospital may be combined with identifying information from a voters' list to infer medical conditions of individuals, which clearly violates their privacy. Second,

numerical private data may also be published as their aggregates. In such a case, private values may be inferred by combining multiple aggregates over such values. In this thesis, we shall focus on data publication in terms of micro-data tables.

From another aspect, data publication applications may be classified into two categories, namely, *online* and *offline* applications. In an offline application, data owners will compute one or more micro-data tables over the secret dataset before releasing the tables. Performance is generally not critical here since the computation may take as much time as needed. Also, what micro-data to release is mainly decided by the data owner and users of such micro-data usually only play a passive role. On the other hand, in an online application, users interact with a database to ask for specific forms of data. In such a case, performance is typically more important since users' queries cannot be adversely delayed. Also, since it is the user who decides which query to be asked next, data owners must examine each new query together with all previously answered queries to determine whether privacy requirement can be satisfied. In this thesis, we shall address the online application in which users may ask for micro-data tables. We call such a system *micro-database*.

Micro-data tables are obtained using generalization techniques that transform a secret relational table into a micro-data table containing coarser values for satisfying privacy properties, such as k-anonymity [43] and l-diversity [34]. For example, a secret table containing patients' personal information, such as names, genders, social identity numbers, addresses, DOBs, and disease information, may be generalized into a micro-data table that has names and social identity numbers removed, and other attributes, except disease information, generalized. Although significant efforts have been seen on privacy preservation for releasing micro-data tables, most of these work are for the offline application of pre-computing one or more micro-data tables to be released, which is different from the online application that we shall study.

Closest to our work, the issue of micro-data auditing is studied in [9] where multiple

views on a secret table are checked for violations of privacy requirements. However, the case where a query cannot be answered and the denial may disclose sensitive information is not the focus in [9]. This issue is addressed in the context of statistical databases in [25], that is, whether giving answers or denials to queries asking for statistical aggregates (SUM, MAX) will violate privacy requirements. We address a similar problem in the different context of micro-databases, and we shall propose an improved strategy than the one in [25].

This thesis addresses three issues: (1) Modeling what information is disclosed by previously disclosed answers to queries; (2) Determining whether a new query is safe with respect to privacy requirements and the answer can be given; (3) Designing an online auditing strategy to satisfy the privacy requirements while not adversely impacting the data utility. Based on the privacy property ℓ-uncertainty introduced in [9], we shall define the auditing problem in the context of a micro-database. We propose a novel auditing strategy for micro-databases for determining whether answering a query satisfies ℓ-uncertainty. We study the maintenance of a so-called *knowledge dictionary* that keeps track of all previously answered queries and their answers. We also study how concurrent queries can be more efficiently handled and provide optimized strategies for this purpose.

The reminder of the thesis is organized as follows. The rest of this chapter reviews relevant background knowledge in inference control, simulatable auditing, and micro-data disclosure. In Chapter 2, we discuss related work on auditing and micro-data disclosure. In Chapter 3, we give motivating examples and an overview of the auditing system, its assumptions, and basic notations. The proposed strategy for the auditing system and corresponding algorithms are presented in Chapter 4. In Chapter 5, we discuss how to efficiently update knowledge dictionary for the system. As extensions of the work, we present the handling of concurrent queries and improved strategies for improving data utility in Chapter 6. In Chapter 7, we give conclusions and future work.

1.1 Inference control

Our study of the auditing problem is partly inspired by a similar problem studied in the context of inference control in statistical databases. In this section, we briefly review inference control.

Inference control has been extensively studied for more than forty years. The goal is to prevent adversaries from inferring secret numeric values from their statistical aggregates. There are broadly two types of inference control techniques, restriction-based and perturbation-based, for protecting privacy in statistical databases. In the restriction-based techniques, the user queries the database through a privacy protection mechanism, which may decide to either deny the query or answer it, in order to ensure privacy. Perturbation-based techniques add random noises to sensitive data or answers to queries such as to prevent the violation of privacy requirements.

To determine the safety of answering queries, restriction-based techniques base their decisions on various properties, including the minimal number of values aggregated by a query, the maximum number of common values aggregated by different queries, the approximate intervals that can be guessed from query results, and the maximal rank of a matrix representing all answered queries. Instead of enumerating all the methods in the literature, we show a few examples of restriction-based techniques in the following.

First, one of the first efforts on inference control is the *query set size control*. The subset of a relational table that matches a query's WHERE clauses is called a query set. Query set size control denies a query whose query set includes less than k tuples. If the database has totally n tuples, then the query whose query set is greater than $n - k$ is prohibited, since its complement includes less than k tuples, which violates the control. For example, consider the simple relation $comm$ shown in Table 1, and suppose the query set size control is in place with $k = 2$. The following query has a singleton query set (whose only member is the first record), and hence is denied.

```
SELECT SUM(commission)
FROM   comm
WHERE  employee='Alice' and quarter='Q1' and location='Domestic'
```

employee	quarter	location	commission
Alice	Q1	Domestic	800
Alice	Q1	Interational	200
Bob	Q1	Domestic	500
Mary	Q1	Domestic	1200
Mary	Q1	International	800
Bob	Q2	Interational	1500
Mary	Q2	Domestic	500
Jim	Q2	Domestic	1000

Table 1: An Example Relation $comm$ for Inference Control

By considering that the `WHERE` clause is the conjunction of three conditions, we can ask the following two queries. The query set of both queries is between k (two) and $n - k$ (seven). Both queries will thus be allowed. Subtracting the result of the second query from that of the first query leads to the inference of the first record. We can similarly infer any query asking for the aggregation of less than k records.

```
SELECT SUM(commission)
FROM   comm
WHERE  quarter='Q1' and location='Domestic'
```

```
SELECT SUM(commission)
FROM   comm
WHERE  employee<>'Alice' and quarter='Q1' and location='Domestic'
```

The above tracker will depend on records to be inferred, and each inference requires a different tracker. A better way is to find a universal tracker and pad it to any record (or query) to be inferred. For example, in Table 1 we can form the tracker with the selection condition `employee='Mary' or employee='Jim'`. Because the tracker's query set has four records, the following two queries will both be allowed.

```
SELECT SUM(commission)
FROM   comm
WHERE  employee='Mary' OR employee='Jim'
```

```
SELECT SUM(commission)
FROM   comm
       employee<>'Mary' AND employee<>'Jim'
```

Adding the results to the two queries together gives us the total commission in the table. Next, without loss of generality, suppose we want to infer the first record. We ask the following two queries.

```
SELECT SUM(commission)
FROM   comm
WHERE  (employee='Alice' AND quarter='Q1' AND location='Domestic')
       OR (employee='Mary' OR employee='Jim')
```

```
SELECT SUM(commission)
```

```
FROM    comm
WHERE   (employee='Alice' AND quarter='Q1' AND location='Domestic')
        OR (employee<>'Mary' AND employee<>'Jim')
```

We can see that the first query pads the first record with the tracker, and the second query pads it with the complement of the tracker. Adding the result to the two queries and then subtracting the value from the total commission that we have computed earlier leads to the inference of the first record. Similarly we can infer any query asking for the aggregation of less than k records. Because the inference pads the query to be inferred with both the tracker and the complement of the tracker. In order for such two queries to be allowed, the size of the tracker must meet a more stringent condition, that is it must be between $2k$ and $n - 2k$. However, more complicated trackers may relax this condition. Trackers are indeed examples of the more general linear system attack we shall discuss shortly. However, the study of trackers nonetheless demonstrates the ineffectiveness of the query set size control.

The above tracker attacks can be generalized with the *linear system attack* based on the following model. Given sensitive values $x_1, x_2, \ldots, x_n$, any SUM query on those values can be modeled as an equation $\sum_1^n a_i x_i$, where $a_i = 1$ if x_i is in the query set and $a_i = 0$, otherwise. A collection of m queries thus form a linear system $AX = D$, where A is an $m \times n$ binary matrix, $X = (x_1, x_2, \ldots, x_n)$, and D is the vector of query results. An inference using the tracker can then be modeled as a sequence of elementary row operations on the matrix A (that is, multiplying a row with a non-zero number, swapping two rows, adding a row to another multiplied by a number).

In order to determine whether any given queries may lead to an inference, we must answer the question: Does there exist a sequence of elementary row operations that transform A into a unit row vector (that is, a vector with one 1-element and all others being 0's)? Chin and Ozsoyoglu show that this can be determined by first transforming A into its reduced

row echelon form (RREF) through a special sequence of elementary row operations, that is the Gauss-Jordan elimination.

The key result lies in that any matrix A can be transformed into a unit row vector through a sequence of elementary row operations, if and only if its RREF includes such a vector. The if part is trivial, because the Gauss-Jordan elimination itself is a sequence of elementary row operations. The only if part can be proved by contradiction. A unit row vector whose 1-element is in the first, second, or fourth position clearly cannot be a linear combination of those four rows in the matrix. A unit row vector whose 1-element is in other positions cannot, either. Because any linear combination of the four rows will have at least one non-zero element in the first, second, or fourth position. The above result is essentially a precise model for inferences of unbounded real values using SUM-only queries.

The above result also yields a method for checking whether a new query, taken together with queries answered before, will cause inferences. A straightforward but inefficient approach is to keep all answered queries and re-computing the RREF when each new query is received. For m queries on n values, the Gauss-Jordan elimination takes time $O(m^2n)$. Considering that the elementary row operations on a matrix is associative, a better approach is to incrementally updates the RREF for each newly answered query. *Audit Expert* maintains the RREF of a matrix that corresponds to the set of queries answered so far. A query is denied when adding it to this set causes an inferences. Because the RREF of a matrix only includes linearly independent rows, the total number of rows must be no greater than the rank of matrix. For queries over n values, this rank cannot be greater than n. After a new row is added to the matrix, the RREF of the new matrix can thus be computed in $O(n^2)$ time.

1.2 Simulatable Auditing

Recall that in an online query auditing application, the data owner will receive a sequence of queries from a user for which answers must be providied The answer is either the true answer of the query, or a denial. The key issue in such an application is that upon receiving a new query that cannot be safely answered, even denying the answer may cause privacy properties to be breached. The issue is first addressed in [25] and a corresponding *Simulatable Auditing* strategy is proposed as a solution. Our work extends the idea of simulatable auditing from statistical databases to micro-data publication applications.

To make the above issue more concrete, we consider some examples of statistical database auditing. Suppose that an adversary sends the first query asking for $sum(x_1, x_2, x_3)$ and gets the answer 15. The adversary then asks for the second query $max(x_1, x_2, x_3)$, which is denied. The denial tells the attacker that if the answer to the second query were given, then some data value would have been compromised (that is, known by the adversary). The adversary can then guess as follows. If $max(x_1, x_2, x_3) > 5$ were true, then the query would not have been denied, since no value can be compromised. Also, $max(x_1, x_2, x_3) < 5$ is not possible either, since $sum(x_1, x_2, x_3)$ equals 15. Consequently, $max(x_1, x_2, x_3) = 5$ must have been the true answer to the second query, and the adversary then immediately learns that $x_1 = x_2 = x_3 = 5$.

This problem is possible even with the same type of queries. For example, suppose a query asks for $max(t_1, t_2, t_3, t_4)$ and is answered as 10. Then for the second query asking for $max(t_1, t_2, t_3)$, it is not safe to deny since the denial will tell the adversary that t_4 must have the value 10, because this is the only case a value can be compromised and thus the query would be denied. Similarly, if the second query is answered, the adversary can ask for a third query for any two of these three values, and we would be in the same situation where a denial is not safe.

The above examples show that the auditing system may fail to protect data privacy

even when denying a query. The denial itself leaks information from which the adversary learns secret values. The reason is that the auditing system decides to deny a query based on information not available to the adversary (that is, the secret values). That is, if the decision for allowing or denying a query depends on the actual data values, then the decision will convey information to avdersaries, even if it is simply a denial.

A simulatable auditing strategy is proposed in [25]. The model is based on the idea that the auditing process is simulatable in the sense that all decisions to either deny or answer a query is independent of the actual data values. In this way, the denial of a query does not convey any information to the adversary beyond what is already known by him/her from previously answered queries' answers. In another word, the adversary can tell by himself/herself what decision would be made. Simulatable auditing keeps all posted queries $q_1, ..., q_t$ and the answers $a_1, ..., a_t$ to decide whether to answer or deny a newly posed query q_{t+1}. Note that queries $q_1, ..., q_t$ are only the answered queries. The denied queries will not be counted as knowledge of adversaries since the denied queries are asked by the adversaries and the denials do not carry any additonal information.

More formally, simulatable auditing is defined in [25] as: *An auditor is simulatable if the decision to deny or give an answer to the query q_t is made based exclusively on $q_1, ..., q_t$, and $a_1, ..., a_{t-1}$ (and not a_t and not the dataset X=$x_1, ..., x_n$) and possible also the underlying probability distribution D from which the data was drawn.*

1.3 Micro-Data Disclosure

Unlike in statistical databases, the key issue of micro-data disclosure is for the data owner to release generalized views, called microdata tables, over the secret table while protecting individual's privacy. Various privacy properties have been proposed, such as k-anonymity, l-diversity, t-closeness, uncertainty, indistinguishability, and so on. Various generalization techniques have also been proposed to transform a relational table into micro-data tables

that satisfy given privacy properties. We shall review these through a few examples in the following.

An example of micro-data table T is shown in Table 2. Suppose the sensitive attribute is patients' medical conditions. Simply deleting the identifier Name before releasing the table is not sufficient due to potential linking attacks using the quasi-identifiers, the attribute DoB in this case. The sensitive attribute for a tuple may potentially be linked to a unique person though such quasi-identifiers.

A Micro-Data Table T

Name	**DoB**	**Condition**
Alice	1990	flu
Bob	1985	cold
Charlie	1974	cancer
David	1962	cancer
Eve	1953	headache
Fen	1941	toothache

Generalization $g_1(T)$

DoB	**Condition**
1980∼1999	flu
	cold
1960∼1979	cancer
	cancer
1940∼1959	headache
	toothache

Generalization $g_2(T)$

DoB	**Condition**
1970∼1999	flu
	cold
	cancer
1940∼1969	cancer
	headache
	toothache

Generalization $g_3(T)$

DoB	**Condition**
1960∼1999	flu
	cold
	cancer
	cancer
1940∼1959	headache
	toothache

Table 2: A Micro-Data Table and Three Generalizations

To prevent the linking attack, the micro-data table needs to be generalized to protect each individual's anonymity in the table. The k-anonymity property is proposed for this purpose. k-anonymity is to obfuscate quasi-identifier values such that an adversary can only link a real world individual to at lease k tuples in the micro-data table. The tabular $g_1(T)$ in Table 1 shows a generalization that satisfies 2-anonymity. Any identity in this release will belong to a group of two tuples that are anonymous. Therefore, a linking attack

cannot bind a person to a unique tuple through the quasi-identifier. However, k-anonymity by itself is not sufficient since linking a person to the second group in $g_1(T)$ already reveals the condition to be cancer; or if the adversary knows that Eve does not has toothache, by linking Eve to third group, Eve's condition has been compromised.

To avoid having only the same sensitive values in a group and the above external knowledge-based attacks, the generalization must also ensure enough diversity inside each group of sensitive values, namely, to satisfy the l-diversity property. For example, assume 2-diversity is desired. If the generalization $g_2(T)$ is disclosed, a person can at best be linked to a group with three different conditions among which each is equally likely to be that person's real condition. The desired privacy property is thus satisfied. Various algotithms have been proposed to apply l-diversity in micro-data generalization.

On the other hand, when the generation algorithms is publicly known, the above privacy metric needs to be applied more carefully. If we do not consider such knowledge, an adversary looking at $g_2(T)$ in Table 2 can guess that the three persons in each group may have the three conditions in any given order. Therefore, the adversary's mental image of T is a set of totally $3! \times 3! = 36$ table instances, each of which is equally likely to be T (a common assumption is that the quasi-identifier attribute is public knowledge).

Now suppose we consider an adversary knows the generalization algorithm, which has considered $g_1(T)$ before it discloses $g_2(T)$. This knowledge will enable the adversary in excluding some invalid guesses from the permutation set. The only reason that the system release $g_2(T)$ instead of $g_1(T)$ is that $g_1(T)$ violates the data privacy requirement. So the adversary can refine his/her guesses of T to a smaller set of tables. As the result, it can be shown that 2-diversity will be violated in this particular case (we shall omit details here).

Solutions to the above problem exists, such as that in [56]. In order to determine whether a generalization is safe to disclose, a *disclosure set* is used to model adversaries' mental image of the secret table in order to evaluate the desired privacy property, such as

l-diversity. The evaluation is based on all considered, but unused generalizations, and is a recursive process. For example, consider how to compute the disclosure set of next generalization, $g_3(T)$, in Table 2. In the permutation set of $g_3(T)$, we need to exclude any table T, if either $g_1(T)$ or $g_2(T)$ satisfies 2-diversity. However, to determine whether $g_2(T)$ satisfies 2-diversity, we would have to compute the disclosure set of $g_2(T)$ to determine if it violates 2-diversity of $g_1(T)$.

The above discussions based on privacy properties such as k-anonymity and l-diversity only apply to the disclosure of anonymized micro-data table in offline applications. For online applications where users may ask for specific views over the micro-data table, only limited efforts have been made, as demonstrated in Table 3 where data is published as multiple views over the same secret table.

Private Micro-Table

Name	**Job**	**Salary**	**Condition**
George	Manager	70K	cold
John	Manager	90K	Obesity
Bill	Lawyer	110K	HIV

$\prod_{Name} \sigma_{Salary>80K}$

Name
John
Bill

$\prod_{Name} \sigma_{Salary<105K}$

Name
George
John

$\prod_{Condition} \sigma_{80K<Salary<100K}$

Condition
Obesity

Table 3: Privacy compromised from views

Uncertainty and indistinguishability are two privacy properties proposed for the online application of data publication. Uncertainty means that given a set of k distinct private values, the adversary cannot tell which private value an individual actually has. The example of applying uncertainty problem to multiple micro-data views is shown in table 3, where none of the three released answers to queries will violate the privacy requirement. However, by combining all three views, John's medical condition will be known.

Indistinguishability refers to the property that the adversary cannot see the difference

among k distinct individuals based on the released micro-data, such that none of the k individuals would draw special attentions from the adversaries. Indistinguishability is another aspect of privacy that is different from uncertainty. For example, an adversary may reveal an employee John's salary to be in a large interval (say, 100K to 300K annually). There may be enough uncertainty. However, if the adversary also reveal that the salaries of all other employees are in ranges that are very dierent from John's range (say, in ranges of 50K to 100K), then John's privacy may still be considered as violated, since he will draw special attentions from adversaries. As another example, suppose from the released data we can infer that all patients in a hospital may only have Cold or SARS except that John may have Cold or AIDS. Even though the uncertainty of John's condition has the same amount of uncertainty as that of the other patients, John may still feel his privacy is violated, since he is the only one who possibly has AIDS. For the protection from being brought to the attention of others, the indistinguishability should be applied.

Chapter 2

Related work

There are generally two categories of applications for protecting data privacy, online applications and offline applications. In the former, users query the database and the database may deny the query or alter the answer to a query in order to ensure privacy. This is also known as query auditing [13,22,25,41]. In the offline application, the original database is first sanitized or generalized so as to preserve privacy before releasing. The generalization technique has received significant attention lately [1,4,20,28,29]. The results obtained in the online applications may yield better data utility in the sense that only queries of interest to a user are answered, whereas in the offline applications, the user has more accesses to data since the entire sanitized database is available to him/her.

The main challenge for online auditing is to answer queries without allowing inferences of secret individual values. The earliest auditing work include [13] and [41] for statistical databases, which show that with queries involving at least k data elements, each pair overlapping in at most r data elements, any data set can be compromised in $(2k-(l+1))/r$ queries by an adversary knowing l data elements. For fixed k, l and r, if the auditor denies the $(2k-(l+1))/r$ and further queries, the individual data elements are protected. The auditor logs all queries and denies the query q_i satisfying that $|q_i| < k$, $|q_i \bigcap q_t| > r(t < i)$, or $i < (2k-(l+1))/r$.

The auditing methods in [15] checks whether each new query can be safely answered based on a history of previously answered queries. These auditors are simulatable since the denies do not leak any informaiton about the actual data. The authors of [15, 22, 25] considered the same problem in more specific settings of both offline and online auditing. The auditing methods in [25, 33] consider if the decision itself leaks any privacy while solving the problem that a denial answer leaks information to adversaries . In [25], the authors build a simulatable auditor in statistical database for SUM and MAX queries. The auditor checks whether each new query can be safely answered based on the set of previous query answers, which is known as the knowledge base. In this method, the denied query does not leak any information since the decision to answer a new query or not does not depend on actual databases. To improve data utility for simulatable auditing, the authors in [33] propose a method that bind denied queries together to increase data utility while keeping data privacy in an acceptable range.

In the offline auditing problem, the auditor is given a set of queries $q_1, ..., q_n$ and true answers $a_1, ...a_t$ and it must determine if a breach of privacy is possible with those queries answered. If only sum or only max queries are considered, then polynomial-time auditing algorithms are known to exist [7]. However, when sum and max queries are intermingled, then determining whether a value can be uniquely determinded is known to be NP-hard [15]. In [24], the auditors consider auditing subcube queries which take the form of a sequence of 0s, 1s and *s where the *s represent "don't care". For example, the query 10**1* matches all entries with a 1 in the first position, 0 in the second, 1 in the fifth and anything else in the remaining positions. Assuming sum queries over the subcubes, they demonstrate that when a compromise can occur will depend on the number of *s in the queries and also on the range of input data values. In [22], the authors investigate the offline sum auditing problem of boolean data. They begin by proving that the offline sum auditing problem is coNP-hard. Then they give an efficient offline sum auditing algorithm

in the case that the queries are one-dimensional, based on some ordering of the elements, say $x_1, ..., x_n$, each query involves a consecutive sequence of values such as $x_i, x_{i+1}, ..., x_j$.

The offline application of releasing tabular data have received significant attention [1, 4, 20, 28, 29]. A number of information disclosure limitation techniques have been designed for the publication of census tabular data. Two main approaches have been proposed for protecting sensitive cells in such data: *Data Swapping* [19, 38, 48] and *Cell Suppression* [30]. However, these techniques may compromise data integrity of the table. The data swapping approach involves moving data entries from one cell to another in the contingency table in a manner that is consistent with the set of published marginals. In the data suppression approach, cells with low counts will simply be deleted, which then might lead to the deletion of more cells. A measurement of information disclosed through query results over secret view of a table based on the perfect secrecy notion by Shannon is given in [18]. The authors in [11] tackle the problem ascribed to the independence assumption made in [18].

The important notion of k-anonymity has been proposed by Samarati and Sweeney [43, 44, 47] as a model of privacy, where the basic idea is to make a set of records indistinguishable from each other with respect to identifying attributes, namely, quasi-identifers. This concept has received tremendous interest in recent years. In the k-anonymity model, we suppress or generalize some entries in a table so as to ensure that for each tuple in the modified table, there are at least k-1 other tuples in the same table that are identical to the tuple in terms of quasi-identifiers. Consequently, even with the knowledge of an individual's quasi-identifying attributes, an adversary cannot locate an individual's record to more than a set of at least k records. In other words, releasing a table under k-anonymization keeps each individual hidden in a crowd of k-1 other people.

To achieve optimal k-anonymity with the most data utility is known to be computationally intractable [36]. Optimal k-anonymity has been proved to be NP-hard for $k \geqslant 3$. Much

efforts have been seen on efficient k-anonymity algorithms [2,3,14,16,26,42,43], whereas the safety of the algorithms is generally assumed. These algorithms render the data coarser by generalizing the data to make it less specific, or suppress the data and perturb the data to replace with a random value the actual value. A personalized requirement for anonymity is studied in [55]. In [21], the authors approach the issue from a different perspective, that is, the privacy property is based on generalization of the protected data and could be customized by users.

Other models are proposed to address limitations of k-anonymity. A model based on the intuition of *blending individuals in a crowd* is proposed in [46]. Many work focus on the deficiency of allowing insecure groups with a small number of sensitive values, such as l-diversity [34], t-closeness [31], $alpha$-k-anonymity [53], and so on. Machanavajjhala et al. [34] proposed l-diversity as an important step beyond k-anonymity which overcomes the shortcoming that an adversary can discover the sensitive value, when there is little diversity, or through external knowledge. The model requires each equivalence class to have at least l well-represented values for each sensitive attribute. The authors in [55] observe that l-diversity cannot prevent attribute disclosure when multiple records in the table correspond to one individual. They propose to have each individual specify privacy policies about its own attributes. t-closeness is a privacy notion that requires that the distribution of a sensitive attribute in any equivalence class to be close to the distribution of the attribute in the overall table.

In [54], a data anonymitization approach divides one table into two parts for release; one includes original quasi-identifiers and a group of id, and the other include the association between the group id and the sensitive attribute values. In addition, a generic model called GBP was proposed to unify the perspective of privacy guarantees in both generalization-based publishing and view-based publishing [10]. In [9] and [8], the authors give the privacy properties for micro-data disclosure when publication is based on multiple views

since multiple views may reveal data privacy in a non-obvious way.

While most existing work assume the disclosed generalization to be the only source of information available to an adversary, recent work [56] and [52] show the limitation of such an assumption. In addition to such information, the adversary may also know about the disclosure algorithm. With such extra knowledge, the adversary may deduce more information and eventually violate the privacy property. In the work of [56] and [52], the authors discover this issue and correspondingly introduce models and algorithms to address it. However, the method in [52] depends on a specific privacy property, whereas the one in [56] is more general, but it also incurs a high complexity.

Chapter 3

Preliminaries

In this chapter, we first give motivating examples. We then briefly overview the system architecture and discuss privacy metric that we choose for our system. Finally, we introduce basic notations that will be used later in this thesis.

3.1 Motivating examples

Suppose the relationship between individuals and their disease names is considered sensitive. Simply deleting the *identity* (Name) is not sufficient since a tuple's sensitive value may still be linked to a unique identity through the *quasi-identifier*(Age) when combined with external knowledge, namely *linking attack*. We assume individuals can be identified by their unique ages for this example (in general, the quasi-identifier is usually a combination of multiple attributes). Now suppose a query asks for information about those whose age is between 32 to 60. The real answer of this query based on the database is shown as A_1 in Table 4, which is simply a relational table. Hiding the names in the answer will not help to protect privacy, since the ages can still identify each individual.

For protecting privacy, the relationship between names (or ages) and diseases shown in the real answer must be hidden. A simple solution we may immediately come up with is

to give the answer that contain all the values but with the association between names and diseases removed, shown as F_1 in Table 4. F_1 contains the same values for each attribute as the real answer A_1 does. However, it is shown as two independent relations (separated by the double vertical lines). In each relation, the values are simply sorted in ascending order. We call such a relation a *fact*. In this case, based on the released facts, adversaries can only reason that any individual identity can be linked to one of the five diseases (since there is one duplicate) among which each is equally likely to be that identity's true disease.

Real Answer of Query1(A_1)

Name	Age	Disease
Alice	60	flu
Bob	50	cancer
Clark	40	cancer
Diana	35	headache
Ellen	34	gastritis
Fen	33	pneumonia

Fact of Query1(F_1)

Name	Age	Disease
Alice	60	cancer
Bob	50	cancer
Clark	40	flu
Diana	35	gastritis
Ellen	34	headache
Fen	33	pneumonia

Fact of Query 2(F_2)

Name	Age	Disease
Bob	50	cancer
Clark	40	cancer
Diana	35	gastritis
Ellen	34	headache
Fen	33	pneumonia
George	30	HIV

Fact of Query 3(F_3)

Name	Age	Disease
Bob	50	Denial
Clark	40	Denial

Table 4: Micro-data query auditing issues

However, simply hiding the relation is not enough to protect data privacy. Suppose the second query is asked about those of an age between 30 to 50. The answer should be the fact F_2 shown in Table 4. This answer seems to be safe since any identity can be linked equally likely to a group of five diseases. However, this is not true if adversaries consider the two query answers altogether. Combining F_1 and F_2, an adversary can easily deduce that Alice has flu and George has HIV (consider "subtracting" F_1 from F_2 and vice versa).

To protect privacy, the auditor must thus respond with a denial in order to prevent such a situation.

We say this is the *simple strategy* for auditing, which denies a query if its answer will lead to privacy breach. A flowchart illustrating the simple strategy is shown in Figure 1. As shown in the figure, a query is sent to the database; the answer of this query is sent to the auditor as one of the inputs to the strategy for making a decision as whether to give the answer. In the figure, G stands for a knowledge dictionary which contains the information about all previously released facts (that is, answers to queries). It is also one of the inputs to the auditor's strategy for making a decision. If the answer to current query combined with the knowledge dictionary will violate privacy, a denial must be given instead of the answer; otherwise, an answer is given.

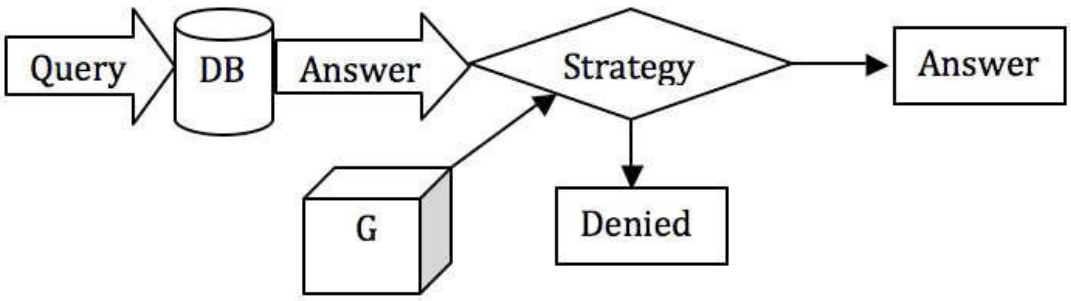

Figure 1: The Simple Strategy

If we were to apply the simple strategy to the online auditing problem, a privacy breach may become unavoidable. For example, suppose a new query asking for age between 40 to 50 is received after the system has released F_1. After the auditor checks whether answering this new query breaches data privacy, the auditor gives a denial, instead of F_3, as the response. As the adversary gets a denial, he/she would be able to deduce the disease

information of Bob and Clark as follows. The adversary first infers that the possible diseases for Bob and Clark from F_1 lead to 11 possible answers as shown in Table5. These 11 possible answers is combined by the known knowledge F_1. Among the 11 combinations, only one shows that Bob and Clark both have cancer. This is the only case where answering this new query is not safe, and thus a denial is given, if we assume 2-uncertainty as the privacy property. Therefore, the denial itself tells the adversary that both Bob and Clark must have cancer. The auditor has no choice at this time, since neither answering the query nor denying it is safe.

Name	Disease
Bob	cancer
Clark	cancer

Name	Disease
Bob	cancer
Clark	flu

Name	Disease
Bob	cancer
Clark	gastritis

Name	Disease
Bob	cancer
Clark	headache

Name	Disease
Bob	cancer
Clark	pneumonia

Name	Disease
Bob	flu
Clark	gastritis

Name	Disease
Bob	flu
Clark	headache

Name	Disease
Bob	flu
Clark	pneumonia

Name	Disease
Bob	gastritis
Clark	headache

Name	Disease
Bob	gastritis
Clark	pneumonia

Name	Disease
Bob	headache
Clark	pneumonia

Table 5: The possible answers

To prevent the situation in which a denial will still breach privacy in online auditing, another strategy, *simulatable auditing* [25] can be applied here. A simulatable auditor will make decisions based on the query and knowledge from previously released answers only, without referring to the actual database. If there exists a possibility to breach data privacy by combining previously answered queries and the answer to the new query, regardless of what the database might be, the new query should be denied by the auditor. To apply simulatable auditing in our example, the auditor should give a denial to the query asking for age between 40 to 50, since there exists the above possibility (a database in which Bob

and Clark both have cancer) that breaches data privacy, even though the actual database may be different.

The flowchart of the simulatable auditing strategy is shown in Figure 2. A query is sent to the auditor. The auditor decides whether to answer this query based on the new query and the knowledge dictionary. If it is impossible to breach data privacy by combining this new query and previously released answers, regardless what the actual database is, then the query will be sent to the database and an answer is given out. Otherwise, the query is denied.

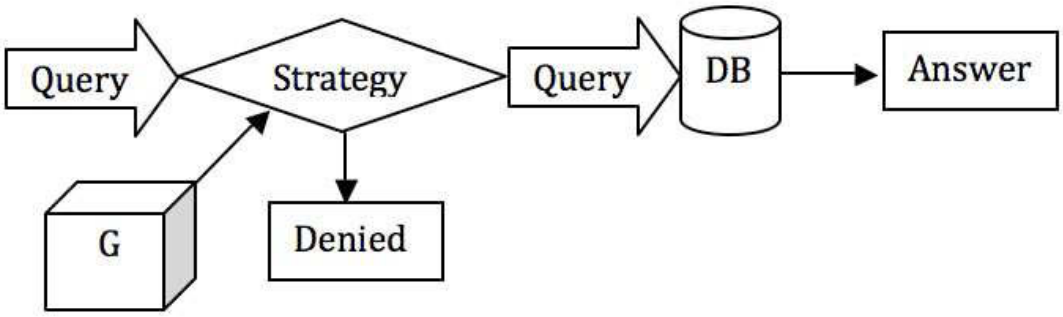

Figure 2: The Scenario of Simulatable Auduting Strategy

The simulatable strategy can protect data privacy. It, however, may be overly strict in the sense that many safe queries may be denied. For example, suppose F_1 (shown in Table 6) has been released. The second query asks for Alice and Bob. By looking at F_1, we can see that the privacy of Alice and Bob may be breached, either when Alice and Bob both may have cancer, or when they both have headache. Therefore, if we apply simulatable auditing strategy here, we would give a denial as the answer to this query right away.

However, the privacy is actually not breached in this case, although simulatable auditing says so, due to the following. There are more than one possibility for breaching privacy. Now consider the simple strategy again. If Alice and Bob both have the same disease,

any one of the above two, then the simple strategy will deny the query. The adversary receiving this denial could learn that Alice and Bob both have cancer, or they both have headache. Assuming the privacy property is that one identity must be associate with more than one sensitive value, then this case does not violate the privacy property. Hence, it is safe to check the actual database for this query, even though there do exist possibilities for breaching data privacy.

In summary, the problem of simulatable auditing strategy is that it requires a denial to never leak any information. However, a response of the auditor may actually leak information, but that information does not breach the data privacy. In order to protect data privacy and release more useful information, we should allow a denial to leak information as long as it does not breach data privacy. The above example leads to the key observation of our approach. That is, the set of all possible table instances, which match previously answered queries and the new one, may be deemed as unsafe under the simulatable auditing strategy, while it can actually satisfy the privacy property. Therefore, it is safe to release the answer to the new query in such a case.

A question now naturally arises, that is, what exactly is the adversary's knowledge obtained from all answered queries and released facts. Suppose the two facts shown in Table 6 (F_1 and F_2) have been released. F_1 and F_2 share some common identities. For those common identities, the associated diseases must appear in both facts. Based on this, we can easily obtain all possible disease values for each identity, as listed in Table 6.

To make decisions based on the simulatable auditing strategy, we should analyze the knowledge dictionary together with the new query. For example, suppose another query (Alice, Bob, Diana) is being asked. Consider the list of possible values. Suppose those three identities all have the same disease: Alice, Bob and Diana all have flu, cancer, or headache. Obviously, in any of these cases, the facts will breach data privacy, since all three people's disease will be the same. However, looking at F_1, we will see that these

F_1

Name	Age	Disease
Alice	33	cancer
Bob	34	cancer
Clark	35	flu
Diana	40	headache
Ellen	50	headache
Fen	60	pneumonia

F_2

Name	Age	Disease
Diana	25	cancer
Ellen	28	flu
Fen	30	flu
George	33	HIV
Helen	34	headache
Ian	35	headache

Possible value list

Name	Disease
Alice	flu, cancer, headache, pneumonia
Bob	flu, cancer, headache, pneumonia
Clark	flu, cancer, headache, pneumonia
Diana	flu, cancer, headache
Ellen	flu, cancer, headache
Fen	flu, cancer, headache
George	flu, cancer, headache, HIV
Helen	flu, cancer, headache, HIV
Ian	flu, cancer, headache, HIV

Table 6: An example showing the need for a meaingful knowledge dictionary

case are not possible, since in F_1 no disease name has appeared three times. That is, the possible values of those identities are not independent, but are correlated. Therefore, the above list is not convenient for the analysis, and we need a better model of adversaries' knowledge.

3.2 Overview

The online auditing problem between an auditor and an adversary can be informally described as the following. The adversary sends a series of queries in an attempt to breach data privacy using responses from the auditor, whereas the auditor will audit all queries to model the adversary's knowledge, and deny queries if not doing so will allow the adversary

to breach data privacy using the knowledge and answers. At the same time, the auditor would like to provide as many answers as possible, since the query may as well be from a normal user instead of an adversary.

The high level architecture of the micro-database auditing system is illustrated in Figure 3. A *query* (Query) is asked over a secret *database* (DB) by an outside user. To satisfy a given privacy property, the auditor follows a *strategy* (Strategy) in processing the query. A *response* (Response) to the query is sent to the outside user based on the strategy, a *knowledge dictionary* (Knowledge Dictionary) that model all previous answered queries and their answers, and/or the database. The auditor transforms all responses to a *fact set* (Fact Set), from which the knowledge dictionary is updated.

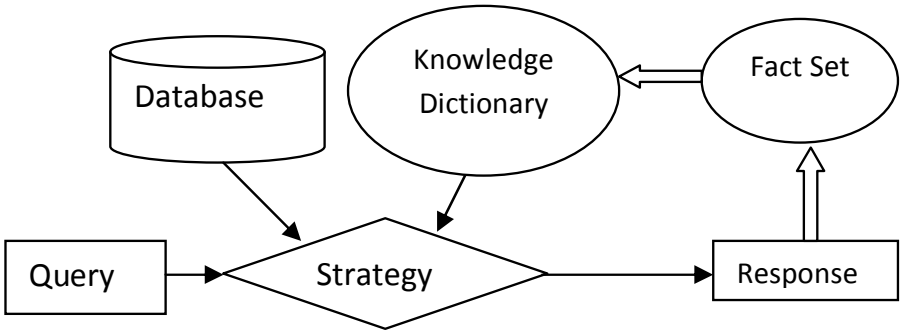

Figure 3: System Architecture

More formally, let T be a relational table consisting of n tuples. For simplicity we consider one attribute of interest contain private information about individuals. A micro-data query q specifies a subset of tuples in T. We consider the following online query auditing problem: Queries $q_1, ..., q_{n-1}$ have already been received and the corresponding answers $a_1, ..., a_{n-1}$ have already been given, where each a_i is either the correct answer to the query q_i, or "denied", for $i = 1, ..., n-1$; given a new query q_n, decide whether to return "denied" if answering the query will violate given privacy property, or the correct

answer to q_n, otherwise.

For privacy property, we follow the uncertainly concept introduced in [9]. The reason we do not use ℓ-diversity or k-anonymity is that in a micro-database, an adversary's knowledge about each identity in an anonymized group may be different, which renders these privacy property proposed for single-view release inapplicable [9]. In our system, we name the privacy property as ℓ-uncertainty. ℓ-uncertainty basically requires that any identity can be deterministically associated with less than ℓ sensitive values in the secret table, where an association is obtained based on all released facts and the answer to the new query. Notice that this concept does not require identities to be grouped, as with k-anonymity or l-diversity.

3.3 Notions and Notations

We use DB for a micro-database that contains relations where each relation T has n tuples $t_1, t_2..t_n$ (n is a positive integer) under the schema D=($ID, QI_1, ..., QI_i, S$), where ID is an attribute that is can be linked to an individual, such as Name or SSN; $QI_1, ..., QI_i$ are quasi-identifier attributes, such as Age, Address, and Occupation; S is an attribute containing private information, such as medical conditions (in our discussions we shall limit our scope such that there is only one sensitive attribute). As commonly assumed in the literature, the relation T is considered as secret but its projections on $ID, QI_1, ..., QI_i$ and that on S are both publicly known.

A fact is a collection of two independent relations with the schema $(ID, QI_1, ..., QI_i)$ and (S), respectively. We use F to denote a set of facts $F_1, F_2, .., F_{l-1}$ that includes all responses the system has given to previously asked queries $q_1, q_2..q_{l-1}$ (we will not investigate the details of a query which can be in any suitable format). Use $q_j(T)$ for the query answer computed based on the secret table T for query q_j. We use $q_j(T)[id]$ for the set

of values of the ID attribute and $q_j(T)[s]$ for that of the S attribute. We use similar notations for the a fact F_j, that is, $F_j[id]$ and $F_j[s]$. We also use $F_j[id]_1$,$F_j[id]_2$.. $F_j[id]_k$ and $F_j[s]_1$,$F_j[s]_2$.. $F_j[s]_k$ to denote elements of the sets.

An *instance* I is a possible guess of the secret table T in the sense that regarding I as T will not cause a conflict to any previously released fact in F. We use $\mathbf{G}$ for the *knowledge dictionary*, which is the collection of all possible *instances* based on all released facts in F. Initially, when no fact has been released from the database, the knowledge dictionary $\mathbf{G}$ would simply include all one-to-one mapping from the projection of T on $ID, QI_1, ..., QI_i$ to the projection on S.

DB	Database
T	Micro-data table
ID	Identity attribute
S	Sensitive attribute
F_i, F	A fact and the set of released facts
$\mathbf{G}$	Knowledge dictionary
I	An instance
q, q_i	A query
$q(T), q_i(T)$	The real answer of a query
$q_i(T)[id], q_i(T)[s]$	The projection of an answer on ID and S, respectively
$F_i[id], F_i[s]$	The projection of a fact on ID and S, respectively

Table 7: Notation Table

Chapter 4

ℓ-Uncertainty Simulatable Strategy

In this chapter, we shall present our simulatable strategy and discuss how the auditor can make decisions while protecting data privacy by satisfying ℓ-Uncertainty.

4.1 Overview

From our motivating examples, we have seen that denials may leak information to adversaries. Such a leakage happens when sensitive information flows from the database to adversaries through the denials to queries. In other words, if a decision whether to answer a query is made based on actual data in the database, then the decision itself may carry information about that data.

A simple solution to the above information leakage problem is to not only deny a query when answering it breaches data privacy, but also deny some safe queries that are randomly chosen by the auditor. This solution prevents the leakage of information caused by denials because the adversary will not know whether a denial is the result of protecting data privacy, or simply a random choice. However, the solution may not be practical. The problem is that the auditor must keep track of all randomly denied queries, since the adversary could repeat a query for which the auditor is expected to give the same answer. Moreover, how

to determine whether two queries are asking for the same result is difficult.

The simulatable auditing model proposed in [25] takes a simpler approach. That is, only previously answered queries and their answers are used to make decisions about whether to answer a new query. Therefore, a denial of queries will not convey any information to an adversary beyond what is already known from previous answered queries and answers. Moreover, the denied queries will not count towards the adversary's knowledge because the denied queries will not leak any information and they would not change the decision for following queries.

From the motivating examples, we know that when a new query's answer leads to unsafe possible instances of the secret database, those unsafe instances altogether may actually be safe. That is, although each such instance does not provide enough uncertainty, they together do, since the adversary cannot determine which such instance indeed corresponds to the secret database. Consequently, it is not necessary for a denial to leak absolutely no information, as required by the simulatable auditing strategy, but instead a denial can leak information but within a safe range.

The key difference between our proposed strategy and the simulatable auditing [25] is that the decision to deny a query for leaking too much information, or to check database for an answer, is independent of the actual database. Our auditor is simulatable since the decision to check database or not is independent of the actual database, so the denial of a query will not breach data privacy beyond what may have already been known based on previous queries and answers.

Definition 1 *An auditor is simulatable if the decision to deny a query or check the database for an answer is only based on previously disclosed information, but not the actual database. The previously disclosed information here include all queries and their corresponding answers, as encoded in the knowledge dictionary.*

Our general strategy works as follows: We compute all possible answers to the new

query q_l using knowledge dictionary alone, without referring to the actual database. For each of these possible answers, we check if privacy is breached based on the ℓ-uncertainty privacy property. We then combine all unsafe answers and check if they together breach privacy. If this is indeed the case, the query will be denied. Otherwise, the query will be sent to the database to compute the answer. If the answer breaches data privacy, the query is denied; the query is answered, otherwise.

4.2 The Strategy

We now design our simulatable auditor. In a database with n secret records, a new query asks for information about a subset of the records. Given a set of previously answered queries $q_1, ..., q_{l-1}$ and their corresponding responses $f_1, ..., f_{l-1}$, and the new query q_l, the simulatable auditor must deny q_l if answering it breaches the data privacy. That is, considering all potential answers to this query, if the collection of all possible instances of q_l does not satisfy ℓ-uncertainty, then this new query should be denied. Notice that in this step, the decision to deny a query or not is independent of the actual database and the real answer to the new query, since we have not referred to either of them. Then, if the query is not denied after this step, then we would check the database and its real answer. If the answer would breach data privacy, the query will still be denied. Otherwise, it will be answered.

Notice that the response from a database auditor is either the real answer computed based on the database, or a denial for protecting data privacy. We now consider how a real answer satisfying ℓ-uncertainty is related to a fact (that is, a collection of separate relations, as defined in the previous chapter). Because the association between those separate relations in a fact has been removed and those relations have the same size, a fact itself satisfies ℓ-uncertainty if the relation containing sensitive values include ℓ distinct values. We have seen that when a fact is disclosed, the knowledge that an adversary can deduce

is usually more than what the fact itself contains. Such knowledge can be modeled as the updated knowledge dictionary (that is, the model of a set of all possible instances of the secret database based on the information disclosed so far) with this new fact taken into consideration. Therefore, we emphasize that when we say a fact satisfies ℓ-uncertainty, it implies that not only the fact itself satisfies ℓ-uncertainty but also the updated knowledge dictionary does. More precisely, we say the knowledge dictionary satisfies ℓ-uncertainty if every identity can be associated with at least ℓ distinct sensitive values among all possible instances of the secret database. We will delay the discussion of updating knowledge dictionary in next section.

Given the two inputs, a new query q_l and the current knowledge dictionary G, the auditor must make a decision as whether to check the database or to deny the query. First of all, the secret table must be one of the possible instances modeled by the knowledge dictionary. Assuming it is the instance *I*, the new query q_l that asks for information about *I* is safe to forward to the database, if its corresponding fact does not breach data privacy and the updated knowledge dictionary, with this query answered, would not breach data privacy, either. We can check each possible database instance in the knowledge dictionary against the privacy property. More formally, we have the following.

Definition 2 (ℓ-Uncertainty Safe Instance) *Given the knowledge dictionary* $\mathbf{G}$ *computed from the set of previously disclosed facts* F*, and a new query* q_l*, we say a table instance* I *(*I $\in \mathbf{G}$*) is* ℓ*-uncertainty safe, denotes as* $\ell SI(I|q_l, \mathbf{G})$*, if* $F' = q_l(I)$ *and the new knowledge dictionary computed from* $\mathbf{G}$ *and* F' *are both safe*.

In Definition 2, the condition for an instance to be ℓ-uncertainty safe is for the fact resulted from the new query, and the new knowledge dictionary resulted from releasing this fact to both satisfy ℓ-uncertainty. As we have discussed, we can determine whether a fact satisfies ℓ-uncertainty simply based on whether the fact includes at least ℓ distinct sensitive values. In order to determine whether a knowledge dictionary satisfies ℓ-uncertainty, we

need to count the number of distinct sensitive values that are assoicated with each identity among all possible instances. For example, suppose there are ten identities and the knowledge dictionary represents a group of ten instances. If all identities are associated with exactly five distinct sensitive values, except one identity to only two distinct sensitive values, then we say this knowledge dictionary only satisfies ℓ-Uncertainty with ℓ=2.

Next, the auditor will collect all instances that are unsafe for answering the new query, and will try to determine whether it is safe to go to the secret database to compute the real answer. Here an unsafe instance is one modeled in the knowledge dictionary to which a denial may breach data privacy. In other words, if this instance is indeed the secret table, then denying the new query after applying the strategy will breach the data privacy. For example, suppose there exists only one instance that is unsafe with respect to ℓ-uncertainty among the set of all possible instances, and suppose after we check the database we find that the real answer must not be given, then at this time a denial will immediately tell adversaries that this unsafe instance is indeed the secret database.

More precisely, with given knowledge dictionary $\mathbf{G}$ and a new query q_l, the query separates current knowledge dictionary to two parts, based on whether an instance is ℓ-uncertainty safe. Let $\triangle(q_l, \mathbf{G}, \ell)$ be the set of unsafe instances, that is, $\triangle(q_l, \mathbf{G}, \ell) = \{I_i | I_i \in \mathbf{G}, \neg \ell SI(I_i | q_l, \mathbf{G})|\}$. As we know, the knowledge dictionary models all possible database instances, so the complement of the set of unsafe instances with respect to the knowledge dictionary will be the set of instances that satisfy ℓ-uncertainty property, which we shall denote as $\nabla(q_l, \mathbf{G}, \ell)$.

Now we look more closely at how a denial may breach data privacy. From the previously examples, we learned that a denial breaches privacy when the adversary infers that the secret table must belong to the set of unsafe instances, $\triangle(q_l, \mathbf{G}, \ell)$; if this set itself breaches data privacy, then the denial would also do so. By saying this set breaches data privacy, we mean that there will be less than ℓ distinct sensitive values associated with at

least one identity across all instances in this set. For example, among all instances in set $\triangle(q_l, \mathbf{G}, \ell)$, suppose we always have either $a = S1$ or $a = S2$, then the privacy of identity a will be breached if we assume ℓ=3. Note that this is true, even though the adversary is not really given the query answer, but only knows the real database is in the set $\triangle(q_l, \mathbf{G}, \ell)$.

One may ask a similar question about the set of safe instances, that is, whether the set of safe instances may also breach data privacy so it is not safe for an adversary to infer that the actual instance is in this set using the responses. However, the answer is always *no* according to Lemma 1.

Lemma 1 *Adversarial knowledge of $T \in \nabla(q_l, \mathbf{G}, \ell)$ does not breach data privacy.*

Proof: By Definition 2, for any instance $I \in \nabla(q_l, \mathbf{G}, \ell)$, the new dictionary G_n computed from the query answer $q_l(I)$ and previous knowledge dictionary $\mathbf{G}$ is ℓ-uncertainty safe. By the definition of knowledge dictionary, we know that for any $I' \in G_n$ (including I), I' is a possible instance in the sense that assuming I' to be the real database will not cause any conflict to the query answer $q_l(I)$, that is, $q_l(I) = q_l(I')$. Since the new knowledge dictionary G_n is computed based on the query answer $q_l(I)$ and previous knowledge dictionary $\mathbf{G}$, which are both identical for I and I', we have that I and I' must both lead to the same new knowledge dictionary G_n, which is ℓ-uncertainty safe. Therefore, all instances in G_n are ℓ-uncertainty safe, that is, we have $G_n \subseteq \nabla(q_l, \mathbf{G}, \ell)$. Also, the new knowledge dictionary of an instance actually forms an equivalence relation over $\nabla(q_l, \mathbf{G}, \ell)$. Since any such equivalence class must satisfy ℓ-uncertainty, $\nabla(q_l, \mathbf{G}, \ell)$ must do so, too, because the number of distinct sensitive values that can be associated with each identity will increase monotonically in the number of possible instances. This concludes the proof. □

Next, to prevent a denial from breaching privacy, the auditing system must make the decision as whether to check the database in a simulatable manner. That is, the decision should be made in a way that adversaries can potentially repeat the process by themselves.

Note that from the above discussions, we have not referred to the real database in determining whether the set of safe and unsafe instances. This means our method is indeed simulatable. Therefore, it would be sufficient for the auditor to only access the real database when the set $\triangle(q_l, \mathbf{G}, \ell)$ satisfies ℓ-uncertainty property.

First of all, we use $\lfloor \triangle(q_l, \mathbf{G}, \ell) \rfloor$ to denote the least number of distinct sensitive values associated with each identities among all possible instances in (the union of) all knowledge dictionaries. We can thus compare this number with the privacy property ℓ to decide whether a denial would breach privacy and whether we should go to check the real database. For example, assuming there are three instances I_1, I_2, I_3 in the set $\triangle(q_l, \mathbf{G}, \ell)$ for query q_l. In the data dictionary of I_1, we have $a = S1$; in that of I_2, we have $a = S2$; for I_3 we have $a = S1$. Then we have $\lfloor \triangle(q_l, \mathbf{G}, \ell) \rfloor$ equal to 2. In summary, we have the following definition and result.

Definition 3 (ℓ-Uncertainty Safe Query) *Given the knowledge dictionary* $\mathbf{G}$ *computed from the set of facts* $\mathbf{F}$ *and given a new query* q_l*, we say query* q_l *is* ℓ*-uncertainty safe, denotes as* $\ell SQ(q_l, \mathbf{G})$*, if* $\lfloor \triangle(q_l, \mathbf{G}, \ell) \rfloor \geq \ell$.

Lemma 2 *Denying a query when a query is* ℓ*-uncertainty safe will not breach data privacy.*

The notion of ℓ-uncertainty safe query is to determine if the auditor can safely access the actual database. From lemma 2, we know that if the set $\triangle(q_l, \mathbf{G}, \ell)$ satisfies ℓ-uncertainty property, to access database and give out a denial does not breach data privacy. This means to determine whether a query is ℓ-uncertainty safe query discloses the same information as a denial of this query. That is, the denial will not allow an adversary to associate any identity to less than ℓ distinct sensitive values. For this purpose, we will check whether $\lfloor \triangle(q_l, \mathbf{G}, \ell) \rfloor$ is less than ℓ by looking at all instances in the set and count vertically to make sure every identity has ℓ distinct sensitive values.

In computing the number of associations between identities and sensitive values, an identity must appear in all the instances of $\triangle(q_l, \mathbf{G}, \ell)$. If an identity does not appear in

one instance, then this identity can basically be ignored in determining privacy breaches since the identity could potentially be associated with any sensitive value in the domain. That is, if there are n unsafe instances in $\triangle(q_l, \mathbf{G}, \ell)$ and one identity only appears in $n-1$ instances, then this identity may take any sensitive value (in the n^{th} instance).

The algorithm for checking whether a query is ℓ-uncertainty safe is shown below.

Algorithm 1 ℓ-Uncertainty Safe Query Checking

Input: Knowledge dictionary $\mathbf{G}$ and a new query q_l

Output: True or False

1: for each $I \in$ G

2: if I is not $\ell SI(I|q_l, \mathbf{G})$

3: insert I into $\triangle(q_l, \mathbf{G}, \ell)$;

5: end if

6: end for

7: for each id $\in \triangle(q_l, \mathbf{G}, \ell)$;

8: count how many sensitive values to which the id could be assigned;

9: end for

10:if there exists a count less than ℓ

11: return False;

12:else

13: return True;

14:end if

After a query passes ℓ-uncertainty checking, the auditor can access the actual database. The actual database could be either an instance in set $\triangle(q_l, \mathbf{G}, \ell)$ or that in set $\nabla(q_l, \mathbf{G}, \ell)$ (recall that we have not accessed the actual database so far). If the database is in set $\triangle(q_l, \mathbf{G}, \ell)$, the query cannot be safely answered since the answer will not satisfy ℓ-uncertainty

property. Therefore, a denial will be given. Notice that this denial is different from the previous one in what causes the denial. We define the ℓ-uncertainty simulatable strategy in the following.

Definition 4 (ℓ-Uncertainty Simulatable Strategy) *Given the knowledge dictionary* $\mathbf{G}$ *and a new query* q_l, q_l *will be answered unless if*

- *it is not* ℓ*-uncertainty safe query, or*
- F_l *violates* ℓ*-uncertainty property*

As shown in Figure 4, a query may be denied at two different stages. First, a query may be denied at the simulatable auditing stage if the denying the query may lead to privacy breaches. Notice that the decision of this denial is independent of the actual database. That is, a query may be denied, if though its answer computed based on the actual database may be safe. This auditing is simulatable since adversaries may obtain the same decision by themselves. Second, the actual database is used to make the decision, and a query will be denied if the answer itself will lead to privacy breaches. This step is not simulatable since the decision will depend on the actual database. Therefore, a denial at this stage will disclose information to adversaries, and the knowledge dictionary will need to be updated (we will discuss the update of knowledge dictionary in next chapter).

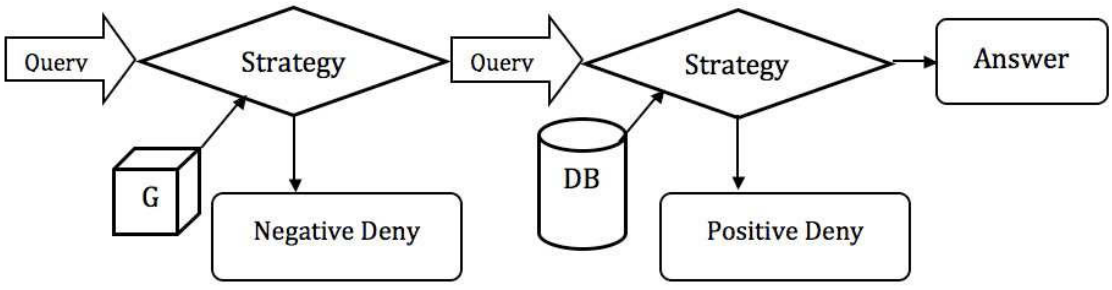

Figure 4: The Scenario of ℓ-uncertainty Simulatable Strategy

The strategy can be more precisely described as the algorithm below. It takes a knowledge dictionary and new query as inputs, and outputs either a denial or answer to the query. The algorithm will return a denial at either line 3 or line 8, corresponding the two cases discussed above. The knowledge dictionary will be updated in the latter case since the denial does disclose information to adversaries. Finally, if the query can be safely answered, the answer will be returned at line 10, and the knowledge dictionary will be updated with the released fact.

Algorithm 2 ℓ-Uncertainty Simulatable Strategy

Input: Knowledge dictionary $\mathbf{G}$ and a new query q_l

Output: Denial, or an answer to q_l

1: determine whether q_l is ℓ-uncertainty safe using Algorithm 1;
2: if $\neg \ell SQ(q_l, \mathbf{G})$
3: return denial;
4: else
5: //access the actual database to compute the answer
6: $F_l = q_l$(T);
7: if F_l violates ℓ-uncertainty
8: return denial and update $\mathbf{G}$;
9: else
10: return F_l and update $\mathbf{G}$;
11: end if
12:end if

Chapter 5

Extensions

In this chapter, we describe some extended work. First, we study how to compute and maintain the knowledge dictionary. Then, we investigate how queries may be delayed or denied in a combined manner in order to improve data utility in the sense that more queries may be safely answered.

5.1 Knowledge Dictionary

In this section, we discuss how to generate the knowledge dictionary and update it while more queries are either denied or answered.

We need a knowledge dictionary to capture adversarial knowledge about possible instances when making decisions on new queries. To consider computing the knowledge dictionary, we consider three states of the auditing system, that is, system initial, the first query, and the following queries.

- When the auditing system first initiates and no query has been received, the knowledge dictionary can be used to model public domain knowledge of the database. For example, such domain knowledge can be modeled as the Cartesian product of the domain of public and sensitive attributes, respectively. For a medical database, this

may be the product of a population list and a list of disease names. This means that nothing has been given out at the system initial time so an adversary would not be able to make any inferences. The knowledge dictionary will include all possible pairings between of each public and sensitive attribute value.

- Now consider the first received query. Since the knowledge dictionary at the system initial will typically not breach the data privacy, the first query can be answered as long as the corresponding fact is safe, that is, the number of unique sensitive value it contains is greater than ℓ. The adversary knows the strategy and knows that if the query asks for a smaller set of values it will get denied and no information will be disclosed. Consequently, after the first answer is given, be it the real answer or a denial, the knowledge dictionary will be either be replaced by a real answer, or remain unchanged in case it is a denial. This case is thus obvious since the answer itself is the only information that is being disclosed to the adversary.

- When following queries are received, how to update the knowledge dictionary will depend on at which stage an answer or denial is given by the strategy. We will study how different updating actions should be taken at different stages of a strategy in the following.

Updating the knowledge dictionary after queries are either denied or answered by the system is the most important step in preventing data privacy disclosure. From last section, we know that the strategy may give a response to a query at three different stages, a denial if the query itself is not safe, a denial if the query's answer is not safe according to the real database, or an answer if the query can be safely answered. For simplicity, we shall call the first denial *negative denial* and the second *positive denial*. Different stages require different updating methods, as we shall enumerate next and illustrate in Figure 5.

- Negative Denial. The negative denial stage is a simulatable stage. When the number

of unique sensitive values associated with each identifier among the set of unsafe instances $\triangle(q, G, \ell)$ is less than ℓ, the strategy will give a denial. This step is simulatable in the sense that an adversary may obtain the same result by himself/herself, with the knowledge of the current knowledge dictionary and the new query. The adversary thus could not infer anything from this denial and the knowledge dictionary will also remain unchanged after a negative denial is given.

For example, suppose an adversary asks a query for those with an age greater than or equal to 50 while he/she has the knowledge F1 shown in Table 4. The system should give a denial since there is only one chance to breach data privacy in which Alice and Bob both have cancer. After this negative denial is given, the adversary cannot infer any information. Hence, the knowledge dictionary will remain unchanged. As shown in Figure 5.a, the knowledge dictionary (as the circle indicates) and unsafe set (the shadowed portion). The number of sensitive values in the unsafe set is less than ℓ. The negative denial is thus the response. Figure 5.d shows the updated knowledge dictionary, which is not changed. The real database state may or may not be in $\triangle(q, G, \ell)$.

- Positive Denial. The positive denial stage uses the real database to compute the answer to a new query in order to determine whether this answer violates ℓ-uncertainty property. To reach the decision of checking for the real answer of this query, the prerequisite is that $\lfloor \triangle(q, G, \ell) \rfloor$ greater or equal to ℓ. From the adversary's point of view, after this denial is seen, he/she knows the real database must be inside set $\triangle(q, G, \ell)$, The knowledge dictionary will thus change to $\triangle(q, G, \ell)$.

  For example, suppose the adversary has the knowledge F1 (as shown in Table 8) and asks for a query about Alice and Bob. Suppose ℓ=2 and the system gives a denial. After the adversary sees this denial, he/she knows that it must be a positive denial since there have are possibilities for breaching the data privacy, that is, either

Alice and Bob both have cancer, or Alice and Bob both have headache. This denial must thus come from the real database, and one of these possibilities corresponds to the real database. Therefore, the new knowledge dictionary for Alice and Bob will change to these two cases. Figure Figure 5.b and Figure 5.e show this case. Figure 5.b is the knowledge dictionary and the shadowed portion is the unsafe set for the new query. Figure 5.e is the updated knowledge dictionary.

- Answer. If an answer does not breach data privacy at the previous stage, then the real answer will be given out. The knowledge dictionary will be updated according to the answer together with the previously knowledge dictionary.

F_1

Alice	cancer
Bob	cancer
Clark	flu
Diana	headache
Ellen	headache
Fen	pneumonia

V_1	Alice	cancer
	Bob	flu
V_2	Clark	cancer
	Diana	headache
	Ellen	headache
	Fen	pneumonia

Table 8: Updating knowledge dictionary

For the same example as above, if the answer {cancer, flu} is given for Alice and Bob, the knowledge dictionary will be updated such that Alice and Bob can only have cancer or flu. Note that an answer may affect other parts of the knowledge dictionary, since in F_1 all identities are equally likely to have one of the six disease, while after the answer is given, two of the six diseases can no longer be associated to the rest of the identities. The new knowledge dictionary is shown as V_1 and V_2. Note that we are using a simplified example here; for more general cases, the updating will not be so straightforward. We will discuss more details later. Figure Figure 5.c and Figure 5.f illustrate this case. Figure 5.c is the knowledge dictionary and the shadowed portion is the unsafe set corresponding to the new query. Figure 5.f is the updated knowledge dictionary.

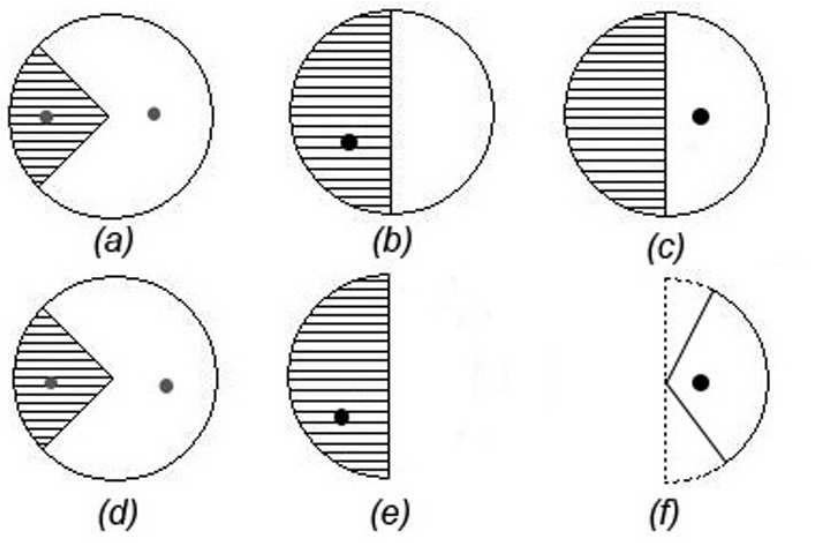

Figure 5: Updating knowledge dictionary

Since a response from the auditing system, which is either an answer to the query or a denial, can only cause some possible instances in the current knowledge dictionary to become impossible and removed, we have the following straightforward result.

Lemma 3 *The size of the knowledge dictionary will decrease monotonically as more and more queries are processed by the auditing system.*

To update the knowledge dictionary with a new fact, there exist three cases. In the first, the identities in the new fact do not appear in the knowledge dictionary. For updating the latter, we shall add the new fact to the knowledge dictionary. In the second case, the identities in the new fact correspond to part of one of the views (the concept of a view will be defined shortly) in the knowledge dictionary, such as in the examples shown below. To update the knowledge dictionary, we shall separate that view to two views. In the third case, the identities in the new fact involve part of (multiple) views, which will be discussed in the following. The algorithm for updating knowledge dictionary is shown below.

Algorithm 3 Updating Knowledge Dictionary

Input: knowledge dictionary $\mathbf{G}$, response(ND, PD, F_l)

Output: New knowledge dictionary $\mathbf{G}$

1: if ND; //Negative denial
2: return G;
3: end if
4: if PD; //Positive denial
5: return $\triangle(q, G, \ell)$;
6: end if
7: if F_l is given;
8: if $F_l[id]$ overlapped with $\mathbf{G}[id]$;
9: Updating overlapped fact with G;
10: else;
11: return $F_l \bigcup \mathbf{G}$;
12: end if
13: end if

To incorporate a new fact whose identities overlap with multiple facts in the knowledge dictionary, we need to separate the intersections to separate views. Here a *view* is similar to a fact, but specifically refers to one in the knowledge dictionary. We shall use *view set* to denote the set of views in a knowledge dictionary whose lists of identities are always mutually disjoint. If we see each view as two separate lists, one of identity values and the other of sensitive values, then identity value lists of all the views in a view set are always disjoint. Intuitively, we can regard views as smaller facts with their intersections separated into new views. Similar to a fact, a view means each sensitive value may be associated to any identity in the view. A straightforward result is that the size of a view set will always

be larger than or equal to the size of the corresponding fact set.

F_1

Alice	cancer
Bob	cancer
Clark	flu
Diana	headache
Ellen	headache
Fen	pneumonia

F_2

Diana	cancer
Ellen	flu
Fen	flu
George	HIV
Helen	headache
Ian	headache

Table 9: Updating knowledge dictionary for multiple facts

An example is given in Table 9. In the given two facts F_1 and F_2, the two lists of identities have 3 common values {Diana, Ellen, Fen}, while those of the sensitive attribute have four common values {flu, cancer, headache, headache}. This phenomenon is caused by the identical sensitive values that appear in the two facts. Obviously, only three out of the four common sensitive values can be associated with the three common identity values. We shall call the sensitive values {flu, cancer, headache, headache} the *possible values* for {Diana, Ellen, Fen} since we cannot determine which of those values can really be associated with the identities. However, by examining the possible combinations of values, we can see that 'headache' here will always be associated with one of the three identities. We say such a value is *determined*. In a special case, if the intersection of the identity lists and that of the sensitive attribute both have the same size, then we can easily see that all the sensitive values are determined. We now convert the two facts shown above into the view set shown in Table 10.

V_1

A	cancer
B	pneumonia
C	*flu*
	cancer
	headache

V_2

D	headache
E	*flu*
F	*cancer*
	headache

V_3

G	flu
H	HIV
I	*flu*
	cancer
	headache

Table 10: Views in a knowledge dictionary

There are three views in this view set. The italic font denotes that the sensitive values

are only possible values in the view. In V_1 and V_3, one of the three sensitive values will belong to this view. In V_2, two of the three sensitive values will belong to it. These possible values inside each view are interdependent. For example, if the adversary has any reason to believe that the value flu should be associated with one of $\{D, E, F\}$, then he can say that flu is determined in V_2, while for V_1 and V_3, flu will be removed from the list of possible values.

When a new fact (the answer to a new query) is released, recall that we need to intersect it with every view in the knowledge dictionary in order to update the latter, as shown in Table 9. However, the aforementioned dependency relationship between views would prevent such an update since the sensitive values in different views will depend on each other. Therefore, we first *break* this relationship by converting the views that contain possible values into sets of views that only contain determined values, by enumerating all possible combinations. For example, in Table 10, the result of breaking the dependent relationship between the three views will lead to $\binom{3}{2}$ different sets of views, as shown in Table 11. Since now all views inside each set only contain determined vales, we can then intersect the new fact with each set of views as before.

A	cancer	A	cancer	A	cancer
B	pneumonia	B	pneumonia	B	pneumonia
C	flu	C	cancer	C	headache
D	headache	D	headache	D	headache
E	cancer	E	flu	E	flu
F	headache	F	headache	F	cancer
G	flu	G	flu	G	flu
H	HIV	H	HIV	H	HIV
I	flu	I	cancer	I	headache

Table 11: Sets of Views

To intersect a new fact (the answer to a new query) with each set of views, it is convenient to organize these views by levels as follows. Suppose currently the knowledge dictionary is the result of releasing totally n facts. Then, we place the view that is the result

of intersecting all the n facts at the top level. Then, the views placed at the second level will be the intersection between each set of $(n-1)$ facts, and there are at most $\binom{n}{n-1}$ such views at this level. Clearly, there can be at most n levels and $\sum_{i=1}^{n}\binom{n}{i}$ views. As shown in Figure 6, we denote those views using integers with larger numbers at a higher level (the order between views at the same level is not important) where $\rho(f)_i^n$ denotes the first number at level i (which is equal to $\sum_{j=1}^{i-1}\binom{n}{j}+1$) and $\rho(l)_i^n$ the last at that level (which is equal to $\rho(l)_i^n = \sum_{j=1}^{i}\binom{n}{j}$). Note that there could be much less views in practice since many facts may have an empty intersection.

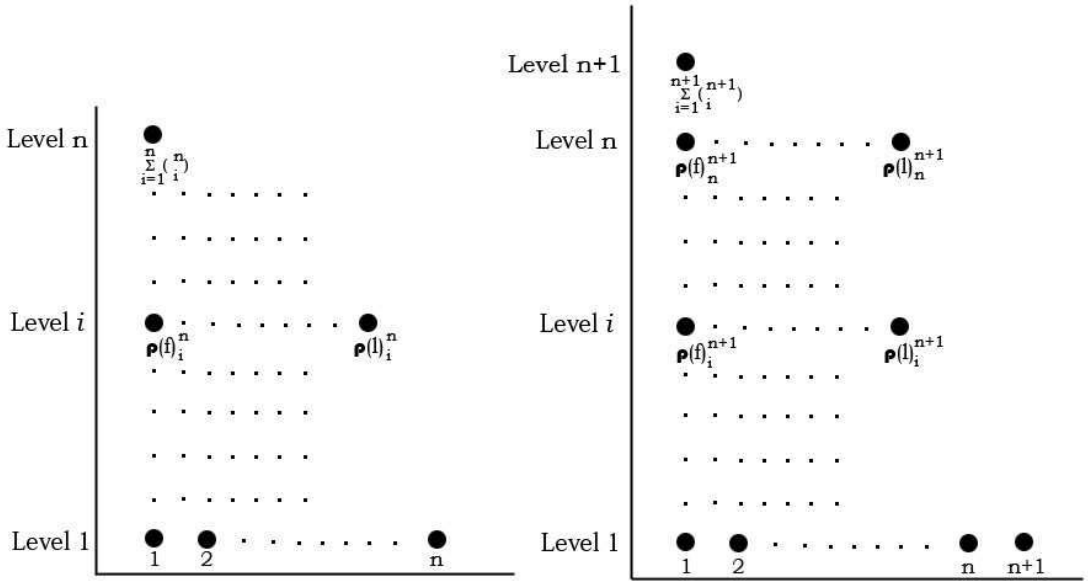

Figure 6: Organizing views by levels

When we intersect a new fact with the views organized as above. We can compute the intersections level by level, from top to the bottom, and the result can then be easily organized in a similar way, as depicted more clearly in the algorithm below.

Algorithm 4 Updating Overlapping Views

Input: The knowledge dictionary $\mathbf{G}$, a new fact F_{l+1}

Output: The updated knowledge dictionary $\mathbf{G}$

1: generate sets of views by breaking the dependency relationship among views in $\mathbf{G}$;

2: organize each set of views $\sum_{i=1}^{l}\binom{l}{i}$ to $\sum_{i=1}^{l+1}\binom{l+1}{i}$

3: for each set of views;

4: for $i = l$ to 1

5: for $j = \rho(f)_i^l$ to $\rho(l)_i^l$

6: $V_{\rho(l)_{i+1}^l+k}$=$F_{l+1} \bigcap V_j$

7: Update F_{l+1} and $V_{\rho(f)_i^{l+1}+k}$

8: $k = k++$

9: end for

10: end for

11: V_{l+1}=F_{l+1}

12:end for

13:return G

5.2 Concurrent Queries

In this section, we study the issue of auditing multiple queries received at the same time, namely, *concurrent queries*. Concurrent queries can be checked in any order, since there is no inherent order among those queries. The auditing system should select an order among queries in order to answer as many queries as possibly allowed by the privacy requirement under a strategy. Therefore, we need to answer following two questions.

- Does the order of checking concurrent queries affect their answers, and if so, can this dependency relationship breach the data privacy?

- How can we maximize the number of concurrent queries that can be safely answered?

First, consider an example of answering concurrent queries by following the ℓ-uncertainty strategy. Assuming two concurrent queries(q_1, q_2) are received at the same time. q_1 asks for the disease information about those whose age is less then 35, whereas q_2 asks for information about those whose age is larger and equal to 40. The adversary has already the knowledge about F_1 shown in Table 12. Applying ℓ-uncertainty strategy here, we consider first checking query q_1 and then q_2. The query q_1 gets a negative denial since Ellen and Fen may have the same disease. Then, query q_2 gets answered with {cancer, cancer, flu} (assuming the privacy requirement is ℓ=2).

However, if we take the reversed order in checking these two queries, q_2 and then q_1, then these two queries will both get answered. First, query q_2 will get answered with {cancer,cancer,flu}, and the knowledge dictionary changes to G_2 as shown in Table 12. Then, query q_1 will get answered with {pneumonia, gastritis}, since it is impossible that Ellen and Fen would have the same disease cancer according to G_2. The knowledge dictionary will then be updated to G_3 (shown in Table 12). This example shows that different orders in checking concurrent queries may indeed lead to different results.

The above example shows that different orders may affect the result of auditing concurrent queries. The reason is that an answer not only depends on the query itself, but also the knowledge dictionary, which will in turn depend on the queries that have been answered before the present one, as described in Lemma 4. However, the order will not matter in a special case where only negative denials are given. This is because a negative denial will not cause the knowledge dictionary to be updated, as we mentioned before. Therefore, if all queries get a negative denial as the answer, then an answer will not depend on how many other such queries have been answered before this query.

Lemma 4 *Different orders in checking a set of concurrent queries may lead to different results of each query, unless all those queries get a negative denial as the answer in which*

F_1

Name	Disease
Alice	cancer
Bob	cancer
Clark	flu
Diana	gastritis
Ellen	headache
Fen	pneumonia
George	HIV

F_2

Name	Disease
Alice	cancer
Bob	cancer
Clark	flu

G_2

V_1	Alice	cancer
	Bob	cancer
	Clark	flu
V_2	Diana	gastritis
	Ellen	headache
	Fen	pneumonia
	George	HIV

G_3

V_1	Alice	cancer
	Bob	cancer
	Clark	flu
V_2	Diana	gastritis
	Ellen	headache
V_3	Fen	pneumonia
	George	HIV

Table 12: Example of Concurrent Queries

case the order does not matter.

Next, we consider whether a denied query can be safely answered by changing the order of auditing concurrent queries. First, we have the following result for positive denials.

Lemma 5 *If q_i gets a positive denial as the answer, then q_i will still be denied even if more queries are checked before it.*

Proof: Consider a set of concurrent queries Q=$\{q_1, .., q_n\}$. First of all, a positive denial implies the following.

- $\lfloor \triangle(q_i, \mathbf{G}, \ell) \rfloor > \ell$, that is, the number of sensitive values in the unsafe set of query q_i must be larger than ℓ;
- DB $\in \triangle(q_i, \mathbf{G}, \ell)$, that is, the real database must belong to the unsafe set;
- The new knowledge dictionary G_i will be updated to $\triangle(q_i, \mathbf{G}, \ell)$.

We know that the above second fact will not change even if more queries are checked before q_i, whereas the first fact may or may not change. Therefore, q_i will either lead to a positive denial (if the first fact becomes untrue) or a negative denial. □

Before we examine the case of negative denials, we build intuitions through an example shown in Table13. Suppose the fact F_1 shown in Table12 is released and a query q_1 is asked about Ellen and Fen. There are 16 (selecting 2 out of 7 then minus 5 since there are two identical values) possible answers for q_1. In this set of possible answers, only one answer could breach the assumed 2-uncertainty property. Set $\triangle(q_1, \mathbf{G}, \ell)$ that includes 120 instances could lead to this unsafe answer in which both Ellen and Fen have cancer. Therefore, the auditor gives a negative denial, and the knowledge dictionary will remain the same. Now suppose another query q_2 is asked, the answer will be given, since no unsafe answer exists, and then the knowledge dictionary is updated as shown in Table 13. Then, suppose the same query q_1 is asked again. With the updated knowledge dictionary, the unsafe answer will actually no longer be a possible instance. The number of possible answers for q_1 are thus reduced to 6, and all these answers turn out to be safe with respect to 2-uncertainly property. Therefore, q_1 can be answered now.

Lemma 6 *When a query q_i leads to a negative denial, q_i may be answered if more queries are checked before q_i.*

Proof: Suppose query q_i gets a negative denial as the answer. This negative denial implies that among instances in set $\triangle(q_i, \mathbf{G}, \ell)$, there must exist at least one identity ID that is associated to $\ell'(\ell' < \ell)$ sensitive values $S_1, S_2, \ldots, S_{l'}$. Also, we know that the knowledge dictionary will remain the same after this response. If another query q_j is asked before q_i and gets answered, then we know that the knowledge dictionary will be updated to include less instances, that is, some possible instances would need to be removed due to the answer. It is thus possible that those instances that contain the association between ID and sensitive values $S_1, S_2, \ldots, S_{l'}$ are altogether removed. Therefore, q_i would now be answerable. □

	$q_1(Diana, Ellen)$	$q_2(Alice, Bob, Clark)$	$q_1(Diana, Ellen)$
# of Possible answers	16	25	6
Unsafe answer(s)	Diana, cancer Ellen, cancer		
$\triangle(q_i, \mathbf{G}, \ell)$	120	$\emptyset$	$\emptyset$
$\lfloor \triangle(q_i, \mathbf{G}, \ell) \rfloor$	1 (Diana has cancer Ellen has cancer)	2	2
Auditor's response	Negative Denial	Answer	Answer
Updated G	Alice, cancer Bob, cancer Clark, flu Diana, gastritis Ellen, headache Fen, pneumonia George, HIV	Alice, cancer Bob, cancer Clark, flu Diana, gastritis Ellen, headache Fen, pneumonia George, HIV	Alice, cancer Bob, cancer Clark, flu Diana, gastritis Ellen, headache Fen, pneumonia George, HIV

Table 13: Examples of Negative Denial ($\ell = 2$)

Now we know that only negative denials will cause the results of queries checked in different orders to vary. From our discussions in previous sections, we also know that negative denials are the result of a simulatable auditing algorithm, that is, an adversary may obtain the same results by himself/herself. If the auditing system checks a set of concurrent queries in an order that maximizes the number of answered queries, then the results would also be simulatable, and therefore, safe. We thus have the following straightforward result.

Theorem 7 *Given a set of concurrent queries Q, it is safe to answer $Q' \subseteq Q$ such that $\mid Q' \mid$ is the maximum possible for satisfying the privacy requirement.*

The naive way for maximizing the number of answerable queries is to check the queries in all possible orders. However, based on the previous discussions, this process can be optimized as shown in the below algorithm. First, we separately check each query using the current knowledge dictionary, and we divide these queries into three groups based on their answers, that is, the answerable, negative denials, and positive denials. We then check queries in different orders within the answerable group and choose the optimal order which

yields the greatest number of answers. Then, we process the positive denial group in which queries shall all get a denial as the answer but will update the knowledge dictionary. Finally, with the updated knowledge dictionary, we then process the negative denial group and output the answers.

Algorithm 5 Checking Concurrent Query

Input: a set of concurrent queries $\{q_1, .., q_n\}$, knowledge dictionary $\mathbf{G}$, ℓ

Output: Query answers

1: Check all queries using the ℓ-uncertainty strategy and group them into three groups A (Answerable), ND (Negative denials), and PD (Positive denials)

2: Determine the optimal order for answering A and place queries leading to denials into ND and PD;

3: Check PD and update G

4: While $ND \neq \phi$

5: Check ND

6: End while

5.3 ℓ-Uncertainty Delay and Joint Strategy

Now we consider possible variations of the basic strategy that can potentially improve data utility. The key observation is that within the aforementioned strategy, the negative denial stage employs a simulatable method in making the denial decision for a query, which means the decision may not be justified by the real database. This simulatable method certainly preserves data privacy but it will also have a greater impact on data utility.

By Lemma 6, we can then come up with a possible way for improving data utility. That is, when a query gets a negative denial as the answer, it may become answerable if some other queries are answered first since those instances that cause this query to be denied may

be removed from set $\triangle(q, G, \ell)$ when additional answers cause the knowledge dictionary to be updated. This is of course under a necessary condition that is the actual database state must not be inside set $\triangle(q, G, \ell)$. Otherwise, if the actual database state is inside set $\triangle(q, G, \ell)$, then by previous discussions we know that the real database always stays in G and $\triangle(q, G, \ell)$ no matter how many queries are answered.

For improving the data utility, we extend the basic strategy to a new strategy called *ℓ-uncertainty delay strategy*, as described in below definition and illustrated in Figure 7.

Definition 5 (ℓ-Uncertainty-Delay Strategy) *Given the knowledge dictionary* $\mathbf{G}$ *computed from previously released facts* $\mathbf{F}\{F_1, .., F_{l-1}\}$*, a new query* q_l *and a delayed query* $q_{l'}$*, the ℓ-uncertainty delay strategy will*

- *delay* q_l*, iff* $\neg \ell SQ(q_l, G)$*;*
  deny q_l*, iff DB* $\in \triangle(q_l, G, \ell)$*;*
  answer q_l*, otherwise*.

- *answer* $q_{l'}$*, iff* $\triangle(q_{l'}, G, \ell) = \emptyset$*;*
  delay $q_{l'}$*, otherwise*.

As shown in Figure , the scenario of ℓ-uncertainty delay strategy is similar to that of ℓ-uncertainty simulatable strategy. There are also three stages for giving a response to a new query. However, instead of giving a negative denial, the auditing system will place the query into a queue and mark it as being delayed. When the knowledge dictionary has been updated after answering some other queries, the delayed queries will be checked again for possible answers.

Taking the adversary's point of view, when a denied query is answered after several other queries are answered, the adversary cannot breach data privacy since the strategy already take this into consideration. The only information that this answer will convey to the adversary is that set $\triangle(q_l, G, \ell)$ does not contain the real database instance. Also,

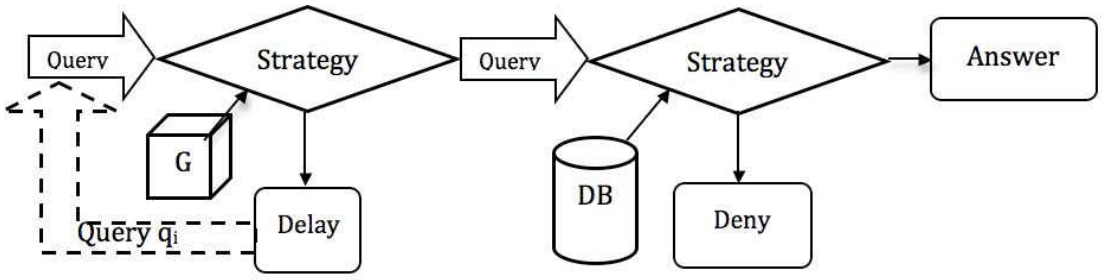

Figure 7: The Scenario of ℓ-uncertainty-Delay Strategy

since this new strategy may eventually answer some queries that would be denied under the previous strategy, the data utility can potentially be improved. Therefore, we have the following straightforward result.

Lemma 8 *ℓ-uncertainty-delay strategy satisfies the ℓ-uncertainty property and has no worse data utility than ℓ-uncertainty strategy.*

On the other hand, waiting for one or more queries whose answers will remove unsafe instances from set $\triangle(q_l, G, \ell)$ is a passive approach. We now consider a more active approach for the same goal of improving the data utility. We explain the approach through an example in the following.

Suppose query q_a has received a negative denial as the answer, and q_a has been denied because its answer will allow adversaries to know $a = s1$ or $a = s2$ (assuming ℓ=3). Also suppose some queries have been processed since then, and the knowledge dictionary has been updated; however, the instances unsafe for q_a have not been removed by those answers. At this point, the auditing system cannot allow q_a to check its answer against the real database by the ℓ-uncertainty delay strategy; the system will thus continue to delay the query.

However, suppose among those queries, another query q_b also gets a negative denial since it allows the adversary to know that $a = s2$ and $a = s3$. Since each of these two

negatively denied queries associates two possible sensitive values to the same identity, a natural question to ask is whether we can *combine* them to yield a scenario where the identity is associated with three possible sensitive values, which are $a = s1$, $a = s2$, and $a = s3$, and thus the data privacy requirement is satisfied (ℓ=3) and both q_1 and q_2 can be answered. This approach would further improve data utility.

Based on this intuition, we define a strategy, called the *ℓ-uncertainty joint strategy*. In this strategy, we create a *waiting pool* to store all the negatively denied queries. In the waiting pool, every query would have an answer that associates less than ℓ sensitive values with an identity. However, once the total number of $\lfloor \triangle(q_i, G, \ell) \rfloor$ for some of the queries in the waiting pool becomes greater than ℓ (we shall call such a set of queries *joint queries*), we forward them to the real database to check for the answer. We give all the answers altogether only if they *all* satisfy ℓ-uncertainty property at the same time; we deny all the joint queries, otherwise. Taking the adversary's point of view again, if we give a denial to one of the joint queries whose answer does not satisfy the privacy property, the adversary can infer that the actual database state must be in set $\triangle(q_i, G, \ell)$ whose size is less then ℓ. Therefore, in this case, we should deny all the joint queries if only at least one of them gets a denial.

To see why the data privacy is not breached when the auditing system denies a set of joint queries as described above, consider Figure 8. In the figure, queries q_1, q_2, and q_3 are joint queries that can collectively pass the negative denial check. If we either deny or answer all the queries at the same time, the knowledge dictionary will change to the illustrated union in which the privacy property is satisfied.

One subtlety is that in determining whether a set of joint queries satisfy a privacy property, we must ensure all the queries are considered at the same time against each instance. For example, suppose $a = s1$ is true for instance I_1 under query q_a, $a = s2$ for instance I_2 under query q_a, $b = s1$ for instance I_1 under query q_a, and $b = s2$ for instance I_2 under

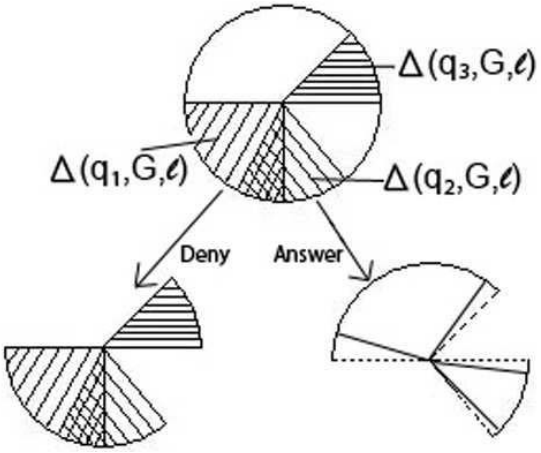

Figure 8: Knowledge dictionary for joint queries

query q_b. Suppose these two queries are joint queries, and we need to determine whether they satisfy the privacy property of ℓ=3. It might seem that the property is satisfied since altogether there are four different possiblities for a and b to be associated with sensitive values. However, a closer look will reveal the opposite, since either $a = s1$ and $b = s1$ is true with I_1 being the real database, or $a = s2$ and $b = s2$ is true with I_2. Therefore, the privacy property is not satisfied with $\ell = 3$.

Another issue is that since an originally denied query may benefit from being pushed into the waiting pool, it might seem probable that a query that is to be denied by the delay strategy may be pushed into the waiting pool such that it would have a second chance. However, delaying such a query may breach the data privacy. Assume a series of queries $q_a, .., q_{a+x}, q_b, .., q_{b+y}$ are already in the waiting pool and we have $\lfloor \triangle(q_{a+i}, G, \ell \rfloor) \leqslant \ell$ hold for $0 \leq i \leq x$, and also $\sum_{i=0}^{x} \lfloor \triangle(q_{a+i}, G, \ell) \rfloor + \sum_{j=0}^{y} \lfloor \triangle(q_{b+j}, G, \ell) \rfloor > \ell$ is true.

Now suppose these queries have been denied after check with the real database. We thus have $DB \in \triangle(q_a, G, \ell) \cup .. \cup \triangle(q_{a+x}, G, \ell) \cup \triangle(q_b, G, \ell) \cup ., \cup \triangle(q_{b+y}, G, \ell)$. Suppose now we again push these queries into the waiting pool, and another series of queries $\{q_c, .., q_{c+z} q_c, .., q_{c+z}\}$ are then pushed into the waiting pool. The auditing system finds

that the set of queries $q_a, .., q_{a+x}, q_c, .., q_{c+z}$ are safe to check the real database. Now, a denial will not breach the data privacy, since either DB $\in \triangle(q_a, G, \ell) \cup .. \cup \triangle(q_{a+x}, G, \ell)$ or DB $\in \triangle(q_c, G, \ell) \cup .. \cup \triangle(q_{c+z}, G, \ell)$ must true. However, answering the queries will breach the data privacy, since DB $\in \triangle(q_b, G, \ell) \cup .. \cup \triangle(q_{b+y}, G, \ell)$ implies that $\lfloor \triangle(q_b, G, \ell) \rfloor \cup .. \cup \lfloor \triangle(q_{b+y}, G, \ell) \rfloor \leqslant \ell$ must be true. Therefore, pushing already denied queries back into the waiting pool may breach the data privacy.

Based on the above discussions, we now define the ℓ-uncertainty joint strategy as shown below and illustrated in Figure 9.

Definition 6 (ℓ-uncertainty-Joint Strategy) *Given the knowledge dictionary* **G** *computed from previously released facts* $\mathbf{F}\{F_1, .., F_{l-1}\}$*, a new query* q_l*, and the waiting pool* $\{q_a, .., q_b\}$*, the* ℓ*-uncertainty-Joint Strategy will*

- *push* q_l *into the waiting pool, iff* $\neg \ell SQ(q_l, G)$*;*
  deny q_l*, iff DB* $\in \triangle(q_l, G, \ell)$*;*
  answer q_l*, otherwise.*

- *leave all the queries in the waiting pool, if* $\sum \lfloor \triangle(q_i, G, \ell) \rfloor < \ell$*;*
  remove them, if there exists $q_i(T)$ *not satisfying* ℓ*-uncertainty property;*
  answer them, otherwise.

With a similar reason as in the proof of Lemma 8, we have the following result.

Lemma 9 *The* ℓ*-uncertainty joint strategy satisfies* ℓ*-uncertainty property, and it has no worse data utility than the* ℓ*-uncertainty strategy*

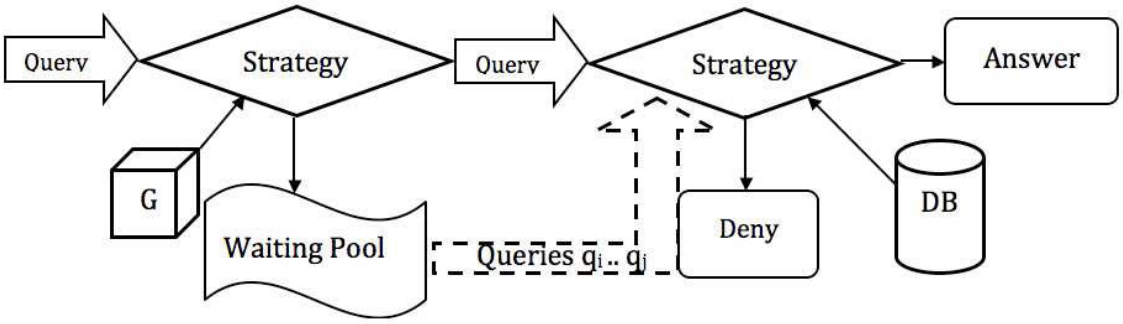

Figure 9: The Scenario of ℓ-Uncertainty Joint Strategy

Chapter 6

Conclusions

In this thesis, we addressed the micro-data auditing issue in revealing useful information about a secret table while preserving private information of individual records. We have proposed a novel ℓ-uncertainty strategy for micro-data auditing under the ℓ-uncertainty privacy property. We have shown how to make decisions for preventing data privacy breaches resulted from answering or denying a query. We have shown that our strategy can prevent not only real answers but also denials from breaching the data privacy. In extended work, we also studied the case of concurrent queries where no inherent order exists among the queries. We have shown that checking concurrent queries in different orders does not breach the data privacy. An optimal order can thus be chosen to improve data utility. Based on this result, we extend the basic strategy to ℓ-uncertainty delay and ℓ-uncertainty joint strategies both of which aim to reduce unnecessary impact on data utility. The first strategy waits for queries to render a denied query safe, and the second strategy combines the answers to multiple denied queries such that all of them may get a chance to be answered. As a first step towards the study of auditing microdata disclosure, we have not considered the computational complexity of our proposed strategies. This will be one of the main focuses of our future work. We also plan to further study the data utility issue in such a context.

Bibliography

[1] 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.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT'05*, pages 246–258, 2005.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, November 2005.

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

[5] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[6] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 217–228, 2005.

[7] F. Chin and G Ozsoyoglu. Auditing for secure statistical databases. *Proceedings of the ACM '81 conference*, pages 53–59, 1981.

[8] C.Yao, L.Wang, S.Wang, and S.Jajodia. Indistinguishability: The other aspect of privacy. In *3rd VLDB Workshop on Secure Data Management*, pages 1–17, 2006.

[9] C.Yao, X.Wang, and S.Jajodia. Checking for k-anonymity violation by views. In *In VLDB*, pages 910–921, 2005.

[10] A. Deutsch. Privacy in database publishing: a bayesian perspective. In Michael Gertz and Sushil Jajodia, editors, *Handbook of Database Security: Applications and Trends*, pages 464–490. Springer, 2007.

[11] A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In *ICDT*, pages 230–245, 2005.

[12] D.Kifer and J.Gehrke. Injecting utility into anonymized datasets. In *SIGMOD*, 2006.

[13] D.P.Dobkin, A.K.Jones, and R.J.Lipton. Secure databases: Protection against user influence. *ACM TODS*, 4(1):76–96, 1979.

[14] Y. Du, T. Xia, Y. Tao, D. Zhang, and F. Zhu. On multidimensional k-anonymity with local recoding generalization. In *ICDE*, pages 1422–1424, 2007.

[15] F.Chin. Security problems on inference control for sum, max, and min queries. *J.ACM*, 33(3):451–464, 1986.

[16] 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.

[17] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 758–769, 2007.

[18] G.Miklau and D.Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, pages 575–586, 2004.

[19] 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.

[20] I.P.Fellegi. On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337):7–18, 1993.

[21] J.Byun and E.Bertino. Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges. *SIGMOD Record*, 35(1):9–13, 2006.

[22] J.Kleinberg, C.Papadimitriou, and P.Raghavan. Auditing boolean attributes. In *PODS*, pages 86–91, 2000.

[23] J.Schlorer. Identification and retrieval of personal records from a statistical bank. In *Methods Info. Med.*, pages 7–13, 1975.

[24] J. Kam and J. Ullman. A model of statistical databases and their security. In *ACM Transactions on Databases Systems*, pages 403–418, 2000.

[25] K.Kenthapadi, N.Mishra, and K.Nissim. Simulatable auditing. In *PODS*, pages 118–127, 2005.

[26] K.LeFevre, D.DeWitt, and R.Ramakrishnan. Incognito: Efficient fulldomain k-anonymity. In *SIGMOD*, pages 49–60, 2005.

[27] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 25, 2006.

[28] 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*, 1982.

[29] L.H.Cox. New results in disclosure avoidance for tabulations. In *International Statistical Institute Proceedings*, pages 83–84, 1987.

[30] L.H.Cox. Suppression, methodology and statistical disclosure control. *J. of the American Statistical Association*, pages 377– 385, 1995.

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

[32] L.Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[33] L.Zhang, S.Jajodia, and A.Brodsky. Simulatable binding: Beyond simulatable auditing. In *Secure Data Management, 5th VLDB Workshop, SDM 2008, Auckland, New Zealand, August 24, 2008, Proceedings*, pages 16–31, 2008.

[34] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[35] Sergio Mascetti. Privacy protection through anonymity in location-based services. *Ph.D. Thesis*, 2006.

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

[37] N.R.Adam and J.C.Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[38] P.Diaconis and B.Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Annals of Statistics*, 26:363–397, 1995.

[39] P.Kalnis, G. Ghinita, K.Mouratidis, and D.Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, 2006.

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

[41] S. Reiss. Security in databases: A combinatorial study. *ACM*, 26(1):45–57, 1979.

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

[43] P. Samarati. Protecting respondents' privacy in microdata release. *IEEE Transaction on Knowledge and Data Engineering*, 13(6), 2001.

[44] P. Samarati and L. Sweeney. Generalizing data to privide anonymity when disclosing information (abstract). *Proceeding of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, page 188, 1998.

[45] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

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

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

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

[49] T.M.Truta and B.Vinay. Privacy protection: p-sensitive k-anonymity property. In *Data Engineering Workshop 2006*.

[50] Ke Wang and Benjamin C. M. Fung. Anonymizing sequential releases. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 414–423, 2006.

[51] W.Liu, L.Wang, and L.Zhang. K-jump strategy for privacy preserving micro-data disclosure. In *ICDT '10: Proceedings of the 13th International Conference on Database Theory*, 2010.

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

[53] R.C. Wong, J. Li, A. Fu, and K. Wang. alpha-k-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *KDD*, pages 754–759, 2006.

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

[55] X.Xiao and Y.Tao. Personalized privacy preservation. In *SIGMOD*, pages 229–240, 2006.

[56] L. Zhang, S. Jajodia, and A. Brodsky. Information disclosure under realistic assumptions: privacy versus optimality. In *CCS*, pages 573–583, 2007.

[57] L. Zhang, L. Wang, S. Jajodia, and A. Brodsky. Exclusive strategy for generalization algorithms in micro-data disclosure. In *Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*, pages 190–204, 2008.