Aggregate Query Answering on Anonymized Tables

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Divesh Srivastava
AT&T Labs - Research
divesh@research.att.com

Ting Yu Qing Zhang
North Carolina State University
{tyu,qzhang4}@ncsu.edu

ABSTRACT

Privacy is a serious concern when microdata need to be released for ad hoc analyses. Simple de-identification has been shown to be inadequate, since privacy can be compromised when quasi-identifiers in a de-identified database are linked with publicly available information. To mitigate the problem, generalization and suppression based approaches (such as k-anonymity and ℓ-diversity) have been proposed to weaken the linkage between the quasi-identifiers of a record and its sensitive attributes in a microdata database. The privacy protection goals of these approaches are only suitable for categorical sensitive attributes. Directly applying them to numerical sensitive attributes (e.g., salary) may result in undesirable information leakage. The first contribution of this paper is to propose privacy goals to better capture the need of privacy protection for numerical sensitive attributes.

Complementing the desire for privacy of microdata is the need to support ad hoc aggregate analyses that select subsets of records based on arbitrary conditions on the quasi-identifiers and compute aggregates over sensitive attributes (e.g., what is the average salary of men over age 50 in Texas?). Approaches based on generalization and suppression cannot, in general, answer such aggregate queries with any reasonable accuracy, thereby reducing the utility of released microdata. The second contribution of this paper is a general framework of permutation-based anonymization to support accurate answering of aggregate queries. We show that, for a specific privacy protection goal, permutation-based anonymization techniques can always answer aggregate queries more accurately than generalization-based approaches. We further propose several criteria to optimize permutations for accurate answering of aggregate queries, and develop efficient algorithms for each criterion. We conduct comprehensive experiments on both real and synthetic data sets to demonstrate the advantages of our proposed techniques.

1. INTRODUCTION

Compared with traditional data dissemination in pre-aggregated or statistical forms, the release of microdata offers significant advantages in terms of information availability, which make it particularly suitable for ad hoc analyses in a variety of domains such as public health, population studies and financial research.

On the other hand, the release of microdata incurs apparent privacy concerns. While supporting ad hoc analyses, it is paramount to prevent private information of individuals from being revealed. Existing privacy practice relies on de-identification, i.e., removing explicit identification information (e.g., name, SSN, home address and telephone numbers) from microdata. However, it has been well recognized that simple de-identification is not sufficient to protect an individual's privacy. One's other attributes (so-called *quasi-identifiers*, such as age, zip code, date of birth and race) are usually needed for data analyses, and thus are kept after de-identification. Individuals' sensitive information may often be revealed when microdata are linked with publicly available information through quasi-identifiers.

k-anonymization is a technique that has been proposed to address the above privacy problem. Through domain generalization and record suppression, k-anonymity guarantees that publicly available information cannot be related with less than k records in a microdata database. In other words, given a sensitive attribute value in microdata, an attacker can at most relate it to a group of no less than k entities instead of any specific individual. The concept of ℓ-diversity was recently proposed to further protect privacy in microdata. It is based on a stronger attack model where an attacker is assumed to have the knowledge that both the record corresponding to an individual and some values of its quasi-identifiers appear in a microdata database.

The privacy goal of k-anonymization and ℓ-diversity is suitable for categorical sensitive attributes, such as the disease attribute in a patient record table. It assumes that different attribute values are incomparable. In ℓ-diversity, for example, as long as it is ensured that an individual's sensitive attribute value can at most be narrowed down to a group of no less than k tuples with no less than ℓ distinct values, one's privacy is protected. In practice, however, besides categorical attributes, many sensitive attributes in microdata databases are in fact numerical data, e.g., one's salary, investment gains or losses. Applying existing privacy goals of k-anonymity and ℓ-diversity is often not sufficient to protect numerical attributes. For example, even when ℓ-diversity is satisfied, if the group of salary values falls into a narrow range, an attacker can still obtain sensitive financial information of an individual. Therefore, it is important to define new privacy goals for the protection of numerical sensitive attributes in microdata.

Complementing the desire for privacy of microdata is the need to support ad hoc aggregate analyses that select subsets of records based on arbitrary conditions on the quasi-identifiers and compute aggregates over sensitive attributes (e.g., what is the average salary of men over age 50 in Texas?). Since most existing approaches achieve privacy through generalization of quasi-identifiers, they

cannot answer aggregate queries with any reasonable accuracy, thereby reducing the utility of the released microdata. In this paper, we investigate effective techniques to support accurate aggregate query answering on microdata while preserving privacy.

The goal of privacy protection in microdata is essentially to break the association between the identifiers in publicly available information and the sensitive attributes in microdata, due to quasi-identifiers. We observe that there is more than one way to eliminate or reduce such associations. Existing generalization-based approaches aim to weaken the link between quasi-identifiers in publicly available databases and microdata. In this paper, we propose *permutation*-based approaches to reduce the association between quasi-identifiers and sensitive attributes in microdata. In particular, the contributions of the paper include the following:

- We extend the concept of k-anonymity, and propose a new privacy goal to better capture privacy protection for numerical sensitive attributes. Besides requiring a group of sensitive attribute values to have no less than k distinct values, the proposed privacy goal further requires the range of the group to be larger than a certain threshold e; such a threshold prevents an attacker from accurately deriving the range of sensitive attribute values for an individual record.

- We propose a general framework of permutation-based techniques to support accurate answering of aggregate queries, while protecting privacy. Given the same privacy objective, we show that permutation-based anonymization techniques can always answer aggregate queries more accurately than generalization-based approaches.

- We design query-rewriting algorithms to ensure that existing database management systems can be used directly to support aggregate query answering in permuted microdata. The auxiliary relations can be completely derived from an anonymized microdata database. Therefore, they do not compromise the privacy of microdata.

- Building on the proposed privacy objective and the anonymization framework, we identify several alternative criteria to optimize permutations for accurate answering of aggregate queries, and develop efficient algorithms for each criterion. Since the proposed permutation-based approach is not constrained by domain generalization hierarchies, our optimization algorithms obtain partitions of microdata tables that can be used to compute very accurate answers to aggregate queries.

- We conduct comprehensive experiments on both real and synthetic data sets to demonstrate the advantages of the proposed techniques.

The rest of the paper is organized as follows. In section 2, we present a detailed example illustrating our approach and its benefits. In section 3, we describe existing privacy goals, and show their inadequacy to protect numerical sensitive data. We then formally define a new privacy goal for numerical sensitive attribute protection. A permutation-based anonymization framework is proposed in section 4. Section 5 presents the query rewriting algorithm and the design of auxiliary relations for aggregate query answering. Criteria for permutation optimization and algorithms for each are described in section 6. Section 7 shows our experiments. In section 8, we report closely related work in privacy protection in the release of microdata. We conclude this paper in section 9.

	ID	Quasi-identifiers			Sensitive
tuple ID	name	age	zipcode	gender	salary
1	Alex	35	27101	M	$54,000
2	Bob	38	27120	M	$55,000
3	Carol	40	27130	M	$56,000
4	Debra	41	27229	F	$65,000
5	Evan	43	27269	F	$75,000
6	Frank	47	27243	M	$70,000
7	Gary	52	27656	M	$80,000
8	Henry	53	27686	F	$75,000
9	Ina	58	27635	M	$85,000

Figure 1: An example microdata table

2. EXAMPLES

Consider the population table shown in figure 1, which needs to be shared for business and economic research. Among the attributes of the table, "name" can be used to directly identify an individual in real life. To prevent the sensitive information ("salary" in this example) of individuals from being disclosed, such explicit identity attributes need to be removed before the table is disclosed. Other attributes of individuals, such as "age", "zip code" and "gender", often need to be disclosed, since valuable statistical analysis relies on these attributes. For this example, we consider the following aggregate queries:

- **Query 1.** The average salary of those with age over 50.
- **Query 2.** The sum of salaries of those with age between 35 and 55.
- **Query 3.** The minimum salary of females.

Clearly, after removing the identity attribute "name" from the table in figure 1, we can still accurately answer these queries.

On the other hand, some public databases may also contain the attributes "age", "zip code" and "gender". They may also further be associated with people's explicit identities. Due to this reason, these attributes are called "quasi-identifiers" since they may be used to reveal one's identity when combined with public databases.

Current approaches to addressing potential privacy violations caused by quasi-identifiers generalize the domains of quasi-identifiers so that many tuples will have the same quasi-identifiers. Figure 2 shows such a generalization, where "age" is generalized to a range of width 10, "zip code" only keeps the first 3 digits, and "gender" is totally suppressed. The resulting table satisfies 3-anonymity, which means that after generalization each tuple can find at least two other tuples with the same values of quasi-identifiers. It also satisfies 3-diversity since among those tuples with the same values of quasi-identifiers there are at least 3 different sensitive attribute values.

In this paper, we propose a permutation-based approach to anonymization. In our approach, tuples in the table are partitioned into several groups such that each group has at least k different sensitive attribute values. We then perform a permutation between the tuples' quasi-identifiers with their sensitive attribute inside each group. Figure 3 shows the table of figure 1 after permutation, where each group has 3 different sensitive attribute values. In fact, the partition used in figure 3 is the same as the one resulting from the generalization in figure 2.

The privacy objective of microdata anonymization is to prevent attackers from knowing sensitive attribute values for an individual. In section 4, we will show that the permutation-based approach will achieve the same privacy protection as existing generalization-based approaches. One benefit of our permutation-based approach is that it will provide more accurate answers to aggregate queries.

	Quasi-identifiers			Sensitive
tuple ID	age	zipcode	gender	salary
1	$[31-40]$	271*	*	$56,000
2	$[31-40]$	271*	*	$54,000
3	$[31-40]$	271*	*	$55,000
4	$[41-50]$	272*	*	$65,000
5	$[41-50]$	272*	*	$75,000
6	$[41-50]$	272*	*	$70,000
7	$[51-60]$	276*	*	$80,000
8	$[51-60]$	276*	*	$75,000
9	$[51-60]$	276*	*	$85,000

Figure 2: An example 3-anonymity microdata table after generalization. Note that this table also satisfies 3-diversity

	Quasi-identifiers			Sensitive
tuple ID	age	zipcode	gender	salary
1	40	27130	M	$54,000
2	38	27120	M	$55,000
3	35	27101	M	$56,000
4	41	27229	F	$65,000
5	43	27269	F	$70,000
6	47	27243	M	$75,000
7	52	27656	M	$75,000
8	53	27686	F	$80,000
9	58	27635	M	$85,000

Figure 3: An example 3-anonymous microdata table after permutation

Let's consider the three queries previously mentioned in this section. Since both approaches introduce imprecise information into the original table, they cannot always get the correct answer for aggregate queries. Instead, for each approach, we can get deterministic lower and upper bounds of the correct answer. We compare the accuracy of the bounds of each approach.

Query 1: The average salary of those with age over 50. This query covers all the tuples in the third group of both figures 2 and 3. Therefore, by using either table, we can get the exact result for the query, which is $80,000.

Query 2: The sum of salaries of those with age between 35 and 55. In both tables, all the tuples in the second group should be included in the aggregation. But, by using the generalized table of figure 2, we face a difficulty when dealing with tuples in the first and the third groups. Since only generalized ranges [31-40] and [51-60] of ages are available to get a correct lower/upper bound, we have to assume that none/all of the tuples participate in the aggregation. Therefore, the bounds based on figure 2 can only be [$210K, $615K].

On the other hand, in our permutation-based approach, we know exactly how many tuples are included in the aggregation from each group of the partition, which aids in the derivation of more accurate bounds. For this query, based on figure 3, we know that 3 and 2 tuples in groups 1 and 3 respectively participate in the aggregation. Therefore, the bounds will be [$530K, $540K], which are much more accurate than those derived from figure 2.

Query 3: The minimum salary of females. There is a great difficulty in answering this query using figure 2, since the "gender" attribute is totally suppressed in order to achieve 3-anonymity. We do not even know whether there exists a tuple with gender female in the original table. Even if we assume that there is at least one female in the table, the best bound we can get is [$54K, $85K].

Based on the permuted table, we know that there is a female in each of groups 2 and 3, but none in group 1. Since all the salaries in group 3 are higher than those in group 2, we can conclude that the minimum salary of females in this table is between [$65K, $75K]. Again, this is more accurate than the bounds derived from figure 2.

group ID	hits	sum-l-b	sum-u-b	min-l-b	min-u-b
1	1	$54K	$56K	$54K	$56K
1	2	$109K	$111K	$54K	$55K
1	3	$165K	$165K	$54K	$54K
2	1	$65K	$75k	$65K	$75K
2	2	$135K	$145K	$65K	$70K
2	3	$210K	$210K	$65K	$65K
3	1	$75K	$85K	$75K	$85K
3	2	$155K	$165K	$75K	$80K
3	3	$240K	$240K	$75K	$75K

Figure 4: An example help table

In section 4, we will show that, given the same partitions, the permutation-based approach always produces more accurate bounds for aggregation queries than the generalization-based approach.

In fact, since we know the exact number of tuples included in the aggregation in each group (we refer to it as the "number of hits"), more statistics about the query result can be derived. For example, besides the lower and upper bounds for SUM and AVERAGE, we may also compute the mean and variance among all the possible answers. Such statistics provide more information. This is not possible for generalized tables.

One nice property is that the lower and upper bounds for an aggregation operation over the whole table can be computed efficiently by combining bounds over each group of the partition. To facilitate efficient query answering over a permuted table, we propose to use a *help table*, which pre-computes the bounds for each group and all the possible numbers of hits. Given an aggregate query, we can simply rewrite it to query both the permuted table, determining the number of hits of each group, and then join this information with the help table to quickly get the bounds for the whole query result. An example help table, for the table of figure 1, is shown in figure 4. Due to space limitations, we only show the bounds for SUM and MIN in the table. It can certainly include those for other aggregation operations.

Both the generalized and the permuted tables satisfy 3-diversity. In fact, they produce the same partition of tuples for anonymization. However, this partition has a problem in terms of privacy. The range of sensitive attribute values in the first group in the partition is only $2,000 while that of others are $10,000. As pointed out in [11], attackers often have external background information about quasi-identifiers of an individual, which enables inferences for the existence of records in the microdata database. In the above examples, if an attacker knows that the age of Alex is 35 and his zipcode is 27101, he is able to derive that Alex's salary is between $54K and $56K. Though the attacker cannot know the exact salary of Alex, this range might be narrow enough to be considered as sensitive. This example shows that for numerical sensitive attributes, besides distinct values in each group of a partition, we also need to consider the range of each group to prevent type of inference as the one described. In this paper we introduce another privacy parameter e, and further require the range of the k distinct values in a group to be no less than e. We call this privacy objective (k, e)-anonymity. For instance, the above generalized and permuted tables only satisfy $(3, 2000)$-anonymity but violate $(3, 10000)$-anonymity.

We note that several techniques already exist to protect the privacy of numerical sensitive attributes. In perturbation-based approaches [1, 4, 6], noise following a certain distribution is added to sensitive attribute of each tuple. Such an approach inevitably

changes important statistics of the marginal distributions of sensitive attributes (e.g., vairance) [14]. Further, depending on the distribution of added noise (e.g., Gaussian Distribution), it is often difficult to derive deterministic bounds for answering aggregate queries.

To some extent, permutation-based anonymization is similar to data swapping techniques where privacy is achieved by exchanging the sensitive attributes of pairs of randomly selected records [5, 7]. Since no noise is introduced, both approaches preserve the marginal distributions of sensitive attributes. However, data swapping is done globally, which has a much larger impact on microdata utility. Even when done in a controlled manner (e.g., rank-based data swapping [13]), it will still produce big errors for aggregate queries. Our experimental results show this point clearly (see section 7).

In the following sections, we formally describe (k, e)-anonymity and a permutation-based approach to anonymization of microdata.

3. BACKGROUND

3.1 Privacy in the Release of Microdata

We introduce notations and concepts to facilitate our discussion on potential privacy vulnerabilities in the release of microdata.

Microdata. There are three types of attributes in an *original* microdata table $\mathcal{M}$: identifiers, quasi-identifiers and sensitive attributes. An identifier ID is an attribute, whose value, if known, can *always* be used to *uniquely* identify an individual in real life. In practice, there may exist multiple identifiers such as one's SSN and telephone number. Quasi-identifiers $\{QI_1, \ldots QI_k\}$ are a set of attributes associated with tuples that not only appear in the microdata table but may also appear in other publicly available databases. Example quasi-identifiers include age, date of birth, zip code, etc. Sensitive attributes, on the other hand, are only contained in the microdata table, and do not appear in public databases. The goal of privacy protection is thus to prevent attackers from knowing the specific values of sensitive attributes associated with individual tuples. Without loss of generality, we assume there is only one sensitive attribute S in a microdata table. We further assume that the domain of the sensitive attribute is numeric, which is widely true in a variety of microdata databases. Examples include salary in population databases, credit score in financial databases, and white blood cell count and other diagnosis indicators in public health databases. Figure 1 is an example of an original microdata table where the identifier, quasi-identifiers, and sensitive attribute are shown.

A de-identified microdata table $\mathcal{D}$ is a projection of $\mathcal{M}$ over quasi-identifiers and sensitive attributes. We call the projection of $\mathcal{M}$ over ID and S the sensitive information table, denoted $\mathcal{S}$.

Note that in some situations the mere fact that there exists a record for a specific individual Alice in the microdata table may also be considered sensitive, even though Alice's sensitive attribute is unknown. For example, the fact that Alice has a record in a microdata table released by a psychiatric hospital may seem quite sensitive. However, as stated in [11], besides public databases, attackers may often have external background knowledge. For example, Bob may physically see that Alice checked into a hospital. Thus, it will be very hard to prevent such information leakage. In this paper, revealing one's sensitive attribute values is considered a privacy violation, but revealing the existence of a record with specific quasi-identifiers is not.

Public information. Attackers may often gain access to publicly available information related to individuals. We model publicly available information as a table $\mathcal{P}$ with the following attributes $\{ID, QI_1, \ldots, QI_k\}$. In practice, there may exist multiple sources of public information, such as county real estate databases and voter registration records. The above model represents the overall public information that an attacker may derive when combining information from multiple sources.

Aggregate queries. We consider queries that select subsets of records from a microdata table based on arbitrary conditions on the quasi-identifiers and compute aggregates over the sensitive attribute. Such aggregate queries are important during microdata analysis in a variety of domains. Since the domain of the sensitive attribute is assumed to be numeric-valued, a variety of SQL aggregation operations, such as COUNT, SUM, AVERAGE, MIN and MAX, can be used in aggregate queries.

Privacy is violated when an attacker successfully recovers one or more tuples in the sensitive information table $\mathcal{S}$. Formally, we have the following privacy definition, which is based on the one proposed in Yao et al. in [20].

DEFINITION 3.1. *Each tuple on (ID, S) is called an* association. *A set $\mathcal{A}$ of associations on (ID, S) is called an* association cover *if all the tuples in A have the same ID value and $\mathcal{A} \cap \mathcal{S} \neq \emptyset$. An association cover of size k is called a k-*association-cover.

For example, considering the microdata table in figure 1, $\{(Alex, \$54,000), (Alex, \$55,000), (Alex, \$56,000)\}$ is a 3-association cover.

DEFINITION 3.2. *A de-identified microdata database $\mathcal{D}$ satisfies k-*Anonymity *if from $\mathcal{D}$ and any given public database $\mathcal{P}$, an attacker cannot derive any association cover with size less than k.*

The above definition captures the essence of privacy in microdata, i.e., preventing the association between an individual's ID and its sensitive attribute value. The originally proposed concept of k-anonymity was defined specifically for generalization based approaches. It required that, after generalization, for each tuple t in the table, there should exist no less than $k-1$ other tuples having quasi-identifiers equal to those of t. This original definition can be viewed as the goal for generalization in order to achieve k-anonymity. Definition 3.2, on the other hand, is declarative and independent of specific techniques for anonymization. Therefore, it serves as a good privacy definition for the comparison of different anonymization techniques.

As shown in section 2, for numeric-valued attributes, preventing attackers from deriving an association cover of size less than k may not be enough to protect one's privacy, especially when the range of attribute values in the association cover is small. Therefore, we propose the following extended definition for the protection for numeric-valued sensitive attributes:

DEFINITION 3.3. *A de-identified microdata database $\mathcal{D}$ satisfies (k, e)-*anonymity *if given $\mathcal{D}$ and any given public database $\mathcal{P}$, any association cover that an attacker can derive satisfies: (1) the size of the association cover is no less k; and (2) the range of the sensitive attribute values in the association cover is no less than e.*

3.2 Anonymization Through Generalization

Most existing works achieve k-anonymity through domain generalization of quasi-identifiers. That is, instead of releasing the exact values of quasi-identifiers, the values are generalized in a way that many tuples appear to have the same values for quasi-identifiers. For example, instead of disclosing one's exact age, the microdata only shows that the age falls into a certain pre-defined range. Thus, an individual identifier can only be associated with those who have the same quasi-identifiers after generalization.

More specifically, following the notation introduced in [11], let D be a domain. A *generalized domain* D^* of D is a domain $\{D_1, \ldots, D_r\}$, such that each D_i is a subset of D, $D_i \cap D_j = \emptyset$ when $i \neq j$, and $\bigcup D_i = D$. Let $x \in D$. The generalization of x under D^*, denoted $g(x, D^*)$, is $D_i \in D^*$, where $x \in D_i$.

Let $D^*_{QI_i}$ be a generalized domain for each quasi-identifier QI_i. Then $D^*_{QI} = D^*_{QI_1} \times \cdots \times D^*_{QI_k}$ forms a generalized domain for quasi-identifiers. Given a tuple $t \in \mathcal{D}$, its generalization under D^*_{QI}, denoted $g(t, D^*_{QI})$, is a tuple t' such that $t'[QI_i] = g(t[QI_i], D^*_{QI_i})$ and $t'[S] = t[S]$. The generalization of $\mathcal{D}$, denoted $g(\mathcal{D}, D^*_{GI})$, is thus $\{g(t, D^*_{QI}) \mid t \in \mathcal{D}\}$.

In the example shown in figure 2, the generalization domains for the three quasi-identifiers are $\{[1-10], [11-20], \ldots\}$, the first 3 digits of zip codes, and $\{*\}$ (denoting any gender), respectively.

Given a microdata table $\mathcal{D}$, a generalized domain D_{QI} in fact defines a partition $\{\mathcal{D}_1, \ldots, \mathcal{D}_r\}$ of $\mathcal{D}$ such that any two tuples t_1 and t_2 in $\mathcal{D}$, belong to the same $\mathcal{D}_i$ if and only if they have the same generalized quasi-identifiers under D_{QI}. Let $r = (qi_1, \ldots, qi_k)$ be a value of quasi-identifiers. We say r belongs to $\mathcal{D}_i$ if the generalization of r under D^*_{QI} is the same as that of tuples in $\mathcal{D}_i$.

Most existing works on k-anonymity only require that, after generalization, each $\mathcal{D}_i$ has no less than k tuples. However, as pointed out in [11], this requirement may not always satisfy k-anonymity when the attacker has background knowledge regarding the appearance of individual records in a microdata table. To fix the problem, in [11], it is further required that there should be at least ℓ distinct sensitive attribute values in each $\mathcal{D}_i$ after generalization. When ℓ is set to be k, the partition obtained by using ℓ-diversity techniques will then guarantee k-anonymity. It is not hard to see that domain generalization can also be used to achieve (k, e)-anonymity when handling numeric-valued sensitive attributes. After each domain generalization, besides checking whether every partition has no less than k tuples, we simply further check whether the partition has a range larger than or equal to e.

However, as shown in section 2, generalization based techniques impose challenges when answering ad hoc aggregate queries. In general, given an aggregate query whose condition over quasi-identifiers is c, and a microdata table generalized under D^*_{QI}, let $\mathcal{D}_i$ be one subset defined by D^*_{QI}. If for every possible $r = (qi_1, \ldots, qi_k)$ that belongs to $\mathcal{D}_i$ under D^*_{QI}, $c(r)$ is true/false, then obviously all the sensitive attribute values in the partition should be included/excluded when computing the aggregate. Otherwise, to get correct lower/upper bounds of the query result, we have to act conservatively, and include none/all of the sensitive attribute values to compute the aggregate, which can be inaccurate.

4. PERMUTATION ANONYMIZATION

The essential reason that an attacker may recover an individual's sensitive attribute value is the existence of the following three links: (1) the link between the identifier and quasi-identifiers in the public database $\mathcal{P}$; (2) the link between the quasi-identifiers in $\mathcal{P}$ and those in the de-identified microdata $\mathcal{D}$; and (3) the link between quasi-identifiers and the sensitive attribute in $\mathcal{D}$. Figure 5 shows the association between identities and sensitive attributes through quasi-identifiers. Breaking or weakening the associations of any of the above links will help protect privacy. Domain generalization actually weakens the second and the third links.

In this paper, we propose to *only* break the third link through permutation. Given a set of tuples in a de-identified microdata table, we randomly permute the association between quasi-identifiers and the sensitive attribute instead of using domain generalization on the quasi-identifiers. Intuitively, even if an attacker can link an individual's identifier with a tuple's quasi-identifier (for example through background knowledge), he will not be able to know with certainty the exact value of the individual's sensitive attribute.

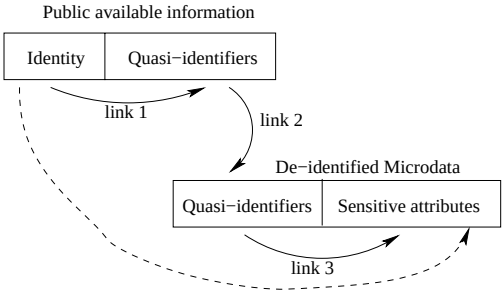

Figure 5: The association between identities and sensitive attributes through quasi-identifiers

DEFINITION 4.1. *Let $T = \{t_1, \ldots, t_n\}$ be a table with attributes $\{a_1, \ldots, a_m\}$, and p be a random permutation over $\{1, \ldots, n\}$. We define the permutation of T, denoted $p(T, \{a_1, \ldots, a_l\}, \{a_{l+1}, \ldots, a_m\})$ as the set of tuples $\{t'_i \mid \forall j, 1 \leq j \leq l, t'_i[a_j] = t_i[a_j]$ and $\forall j, l+1 \leq j \leq m, t'_i[a_j] = t_{p(i)}[a_j]\}$.*

DEFINITION 4.2. *Let $\mathcal{D}$ be a de-identified microdata table with attributes $\{QI_1, \ldots, QI_k, S\}$, and $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a partition of $\mathcal{D}$. A group $\mathcal{D}_i$ is* (k, e)-anonymous *if the projection of $\mathcal{D}_i$ over the sensitive attribute S contains no less than k different values, and the range of these different values in $\mathcal{D}_i$ is no less than e. We say the partition is* (k, e)-anonymous *if every $\mathcal{D}_i$ in the partition is (k, e)-anonymous. We denote $\mathcal{D}'_i = p(\mathcal{D}_i, \{QI_1, \ldots, QI_k\}, \{S\})$. $\mathcal{D}_p = \bigcup_{i=1,\ldots,n} \mathcal{D}'_i$ is a (k, e)-anonymous permutation of $\mathcal{D}$.*

As an example, figure 3 shows a (3,2000)-anonymous permutation of the table in figure 1.

THEOREM 1. *Given a (k, e)-anonymous permutation $\mathcal{D}_p$ and a public database $\mathcal{P}$, any association cover that an attacker can derive satisfies (k, e)-anonymity.*

PROOF. (Sketch) Prove by contradiction that, if attacker derives an association cover $\mathcal{A}$ of size less than k or with range less than e, then we can construct a microdata table $\mathcal{M}$ such that its (k, e)-anonymous permutation is the same as $\mathcal{D}_p$, and the sensitive attribute of that identifier is not in the association cover. □

5. AGGREGATE QUERY ANSWERING

Given a (k, e)-anonymous permutation $\mathcal{D}_p$ and an arbitrary query condition over quasi-identifiers, since the quasi-identifiers of a tuple are unchanged, we know exactly how many tuples satisfy the condition in each group $\mathcal{D}_i$ of the partition. Suppose this number is m_i. Due to the random permutation between quasi-identifiers and the sensitive attribute in $\mathcal{D}_i$, the actual result of the aggregate in $\mathcal{D}_i$ may be over the sensitive attribute of any m_i tuples in the group. Thus, in the worst case, there may be totally $C(|\mathcal{D}_i|, m_i)$ different results for the aggregate. It would be too expensive to enumerate all the possible results when the size of the group is large. Instead, we are interested in efficiently computing important statistics, such as the lower and upper bounds, mean, variance, of all the possible aggregates. Such statistics will be very useful for ad hoc analyses.

5.1 Lower and Upper Bounds

Let m_i be the number of tuples in $\mathcal{D}_i$ that satisfy the condition of an aggregate query Q. If the aggregation operation is monotonic, then the lower/upper bound of the result of Q in $\mathcal{D}_i$ is the aggregation of the m_i smallest/largest sensitive attribute values in $\mathcal{D}_i$. Standard SQL aggregation functions, such as SUM, AVERAGE, MIN, MAX and COUNT, are all monotonic. Therefore, the lower and upper bounds of the result of an aggregate query in each $\mathcal{D}_i$ can be efficiently computed.

The lower and upper bounds of the query over the whole microdata table can be derived from those of each $\mathcal{D}_i$, depending on different aggregation functions. For sum, the overall lower/upper bound is the summation of the lower/upper bound of each $\mathcal{D}_i$, respectively. Since we know the total number of tuples satisfying the query condition, the lower/upper bounds of the average function can be directly computed from those of the sum function. It is also easy to see that the overall lower/upper bounds of the MIN function is the minimum among the lower/upper bounds of all the $\mathcal{D}_i$. The bounds for the MAX function can be obtained similarly.

THEOREM 2. *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_r\}$ be a partition of $\mathcal{D}$ defined by a generalization. Given any aggregate query, the lower and upper bounds given by the generalized table always include that given by the permuted table using the same partition.*

For sum and average, other statistics besides lower and upper bounds can also be computed by combining those of each group in a partition. We omit details for reasons of space.

5.2 Auxiliary Relation and Query Rewriting

We observe that, for the same aggregation operation, no matter what the query condition is, as long as the number of tuples satisfying the condition in each $\mathcal{D}_i$ is the same, the bounds of the aggregation in $\mathcal{D}_i$ remain unchanged. Therefore, we do not need to compute the bounds for each $\mathcal{D}_i$ on the fly when answering a query. Instead, we propose to create a *help table* to facilitate efficient query answering.

The primary key of the help table is "group ID" and "hits", where the former indicates a group in the partition, and the latter represents the number of tuples in the group satisfying a query condition. For each group in the partition and the number of hits, the table lists the lower and upper bounds for each aggregation operation on the sensitive attribute. Figure 4 shows the help table for the permuted table in figure 3. It contains the bounds for SUM and MIN, though in practice it should also contain the bounds of AVERAGE, MAX and other aggregation operations supported by SQL. The number of tuples in the help table is the same as in the microdata table.

Besides the help table, we also create a binary *mapping table* that indicates which tuples the groups of a partition contain.

Given an aggregate query of the form "select agg(sensitive-attribute) from permuted-table where C", we rewrite it to get the bounds of the query result. The rewritten query first selects the tuple IDs of tuples that satisfy condition C from the permuted table PT. The result in the first step is then joined with the mapping table MT to count the hits of each group of the partition. Once this information is available, it is further joined with the help table HT, and the bounds of the aggregation of each group are combined to compute the bounds for the final query result.

More specifically, for the SUM aggregation operation, we rewrite the query as follows.

1. R1 = "select groupID, count(tupleID) AS hits
   from MT, R1
   where MT.tupleID in (select tupleID from PT where C)
   group by groupID"
2. select sum(sum-lb), sum(sum-ub)
   from HT, R1
   where HT.groupID = R1.groupID and HT.hits = R1.hits

The processing of other aggregate queries is similar.

We emphasize that the help table can be constructed directly from the permuted table (using the mapping table), without requiring any access to the original microdata table. Therefore, the use of the help table does not compromise the privacy of microdata in any way. Further, we only need to compute the help table and the mapping table once for a released microdata database. It can be done offline, which will not affect the performance of ad hoc analyses.

6. CRITERIA FOR (K, E)-ANONYMOUS PARTITION

We have shown that, given the same partition, the anonymized table obtained through permutation will always answer aggregate queries more accurately than that obtained through domain generalization. However, given an arbitrary partition, even with the permutation-based approach, it is unlikely to get satisfactory answers to aggregate queries. Thus, in this section, we turn our discussion to the problem of generating "good" (k, e)-anonymous partitions which are likely to produce accurate answers to aggregate queries.

Formally, let D denote a total order of the multiset of sensitive attribute values $D = x_1, x_2, \ldots, x_n$, and $P = \{G_1, \ldots, G_m\}$ be a partition of D. Since D is a totally ordered multiset, we denote the indices of the first and the last data point in a group G_i as min_i and max_i respectively. Thus, the range of G_i is obtained by $[x_{min_i}, x_{max_i}]$. Let $E(G_i)$ denote an error measure defined on a group D_i, and F be a point-wise additive function. We can now formally define the optimal partition problem:

PROBLEM 6.1 (OPTIMAL PARTITION). *Given a total order D of a sensitive attribute, obtain a (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ that minimizes $F(E(G_1), E(G_2), \ldots E(G_m))$ for suitable choices of F and $E()$.*

Given a point query x_q, if $x_q \in G_i$, our scheme will return any point inside G_i as an answer. As a result, the *maximum* error incurred for any point query inside a group G_i is $E(G_i) = x_{max_i} - x_{min_i}$. Therefore, intuitively, the smaller is the range of each group, the smaller error will be introduced to the answer of aggregate queries. Since all the groups of P may be used for querying, it seems imperative to define the function F in a way that the error across all groups, assuming a uniform random workload of point queries, is minimized. It is natural to aim to minimize the *additive* error or the *max* error across all groups. Thus candidate point-wise additive functions are sum or max. We call the optimization problems using the sum and the max functions the *minimum sum-of-error problem* and the *minimum max-of-error problem* respectively. We denote the sum and the max of errors of all the groups in a partition P as $sum\_of\_error(P)$ and $max\_of\_error(P)$ respectively.

6.1 The Minimum Sum-of-Error Problem

Recall that the goal of this problem is to find a (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ of D such that $sum\_of\_error(P)$ is minimized. Without loss of generality, we assume that $G_1, \ldots, G_m$ are ordered according to the index of the minimum value of each group, i.e., $i < j$ implies that $min_i < min_j$, which also means that $x_{min_i} \leq x_{min_j}$.

LEMMA 6.1. *There exists an optimal partition* $P = \{G_1, \ldots, G_m\}$ *to the minimum sum-of-error problem such that for any groups* G_i *and* G_{i+1}*, we have* $max_i < min_{i+1}$.

PROOF. (sketch) We first observe that $max_i < max_{i+1}$. Otherwise, since $min_i < min_{i+1}$, we can simply merge G_{i+1} and G_i into one group, whose maximum error is still $x_{max_i} - x_{min_i}$, and obtain another (k, e)-anonymous partition P'. We have $sum\_of\_error(P') = sum\_of\_range(P) - (x_{max_{i+1}} - x_{min_{i+1}})$, contradicting the fact that P is optimal.

Second, if $max_i > min_{i+1}$, we can still merge G_i and G_{i+1}. The maximum error of the new group is $x_{max_{i+1}} - x_{min_i}$, which is less than or equal to the sum of the maximum errors of G_i and G_{i+1}. □

Lemma 6.1 shows that the ranges of groups in an optimal partition are disjoint. It suggests that this problem has the optimal substructure property, thus it is amenable to dynamic programming solutions. Let $f(i)$ denote the minimum cost way to partition $x_1, \ldots, x_i$ into a number of groups, say m^*, such that the partition is (k, e)-anonymous for $x_1, \ldots, x_i$. Then

$$f(i) = min_{1 \le d \le i} F(f(d-1), E(\{x_d, \ldots, x_i\})) \qquad (1)$$

Thus, the optimal solution for partitioning the data points $x_1, \ldots x_i$ into m^* groups is equal to the minimum cost way of extending (according to the point wise additive function F) the optimal $m^* - 1$ partitioning of $x_1, \ldots x_{d-1}$ (for some d, $1 \le d \le i$) with the group $\{x_d, \ldots, x_i\}$. Algorithm 1 presents the dynamic programming solution for the sum-of-error problem. The algorithm considers all values of $i, 1 \le i \le n$, and for each value of $d, 1 \le d \le i$, assesses the sum of errors using equation 1. The index of the minimum item of each group is stored in the array *partition*. A linear scan of this array at the end of the algorithm (starting from partition[n] and going backwards) will extract the optimal group descriptions. The algorithm makes use of the array $distinct$ that returns in $O(1)$ the number of distinct elements in D between the arguments supplied. The size of this array is $O(n^2)$ and can be populated in a preprocessing step in $O(n^2)$ time, so that access to it remains $O(1)$. It is evident that Algorithm 1 runs in $O(n^2)$.

Algorithm 1 Optimal partition for the minimum sum-of-error problem

```
f[0] = infinity
partition[0] = 0
for i = 1 to n do
   f[i] = infinity
   partition[i] = partition[i-1]
   for d = 1 to i do
      if distinct({x_d, ..., x_i}) ≥ k and x_i − x_d ≥ e then
         error = x_i − x_d
      else
         error = infinity
      end if
      temp = max(f[d-1], error)
      if temp < f[i] then
         f[i] = temp
         partition[i] = d
      end if
   end for
end for
```

THEOREM 3. *Given a total order of a sensitive attribute* D *of* n *items, with* $O(n^2)$ *preprocessing and* $O(n^2)$ *space, we can compute in* $O(n^2)$ *time, the optimal partition of* D *for the sum-of-error problem.*

6.2 The Minimum Max-of-Error Problem

Though seemingly similar to the minimum sum-of-error problem, the minimum max-of-error problem turns out to be much more complex. We have shown that the groups of an optimal partition in the sum-of-error problem are disjoint, i.e., the ranges of different groups are not overlapping (except possibly the boundary where $x_{max_i} = x_{min_{i+1}}$). Therefore, each group can be described completely by using the indices of its first and last attribute values min_i and max_i, and every attribute value with index between them belongs to the group. This property significantly reduces the search space for an optimal partition.

The non-overlapping property, however, does not hold for optimal partitions for the minimum max-of-error problem. As a simple example, consider the following set of sensitive attributes $\{1, 2, 3, 5, 5, 6, 6, 8\}$, the only optimal partition for $(4, 5)$-anonymity is $G_1 = \{1, 2, 5, 6\}$ and $G_2 = \{3, 5, 6, 8\}$, where G_1's range $(1, 6)$ overlaps with G_2's range $(3, 8)$.

On the other hand, we observe that the minimum max-of-error problem has the following property.

LEMMA 6.2. *There exists an optimal* (k, e)*-anonymous partition* $P = \{G_1, \ldots, G_m\}$ *for* D *such that there are no more than two groups whose ranges overlap with each other. In other words, there is no value in the domain of the sensitive attribute such that the value falls into the ranges of more than two groups.*

PROOF. (sketch) First, by a similar proof to that of lemma 6.1, no group's range is included by that of another.

Second, given any optimal partition, suppose there are three groups G_i, G_j and G_l, $i < j < l$, whose ranges overlap with each other. Then we must have $x_{max_j} \ge x_{min_l}$. Otherwise, since $x_{max_i} \le x_{max_j}$, we have the ranges of G_i and G_l are not overlapping. We thus can divide all the items in G_j into two groups, G_{j1} including those less than x_{min_l}, and G_{j2} including those greater than or equal to x_{min_l}. We merge G_{j1} with G_i and G_{j2} with G_l. The ranges of G_i and G_l do not change, which means the new partition is still optimal. By repeating this step, we will get an optimal partition where the ranges of no three groups overlap with each other. □

Let $P = \{G_1, \ldots, G_m\}$ be an optimal (k, e)-anonymous partition that satisfies the above property. Consider the first two groups G_1 and G_2. If the ranges of G_1 and G_2 do not overlap, then G_1 contains all the values from $x_{min_1} = x_1$ to x_{max_1} in D. Then the remaining groups in fact form an optimal partition for the rest of the items $x_{max_1+1}, \ldots, x_n$.

On the other hand, suppose the ranges of G_1 and G_2 overlap, which means $min_2 < max_1$. We divide G_2 into two parts, the former part G_{2f} which includes those items less than or equal to x_{max_1}, and the latter part G_{2l} which includes those items greater than x_{max_1}. Let t be the smallest index of items in D such that $x_t > x_{max_1}$. According to lemma 6.2, G_1 and G_{2f} combined together include all the values in D that are less than x_t. Further, we have $x_t \in G_{2l}$. Otherwise, suppose $x_t \in G_i, i > 2$. Then x_t must be the smallest value of G_i. This allows us to merge G_{2f} with G_1 and G_{2l} with G_i. The resulting partition P' is still optimal, and there is no overlap between the first two groups in P'.

Based on the above observation, we have that G_{2l}, G_3, ..., G_m forms a partition of $x_t, x_{t+1}, \ldots, x_n$. Except G_{2l}, every group is (k, e)-anonymous. For G_{2l}, it has at least $k - d$ distinct values where d is the number of distinct values in G_{2f}. Further, the width of the range of G_{2l} is no less than $e - r$, where $r = x_t - x_{min_2}$. In other words, given any partition $P' = \{G'_1, \ldots, G'_z\}$ of $x_t, \ldots, x_n$, such that G'_1 is $(k - d, e - r)$-anonymous and the rest are (k, e)-anonymous, if the maximum

Algorithm 2 Optimal partition for the minimum max-of-error problem

```
for i = n to 1 do
  for d = 0 to distinct(1, n) do
    for j = 0 to i do
      r = x_i − x_j
      if distinct({x_i, ..., x_n}) < k − d or x_n − x_i < e − r
      then
        g[i][d][r]=INF
        continue
      end if
      g[i][d][r]=x_n − x_i
      for max_1 = i to n − 1 do
        if distinct({x_i, ..., x_{max_1}}) < k − d then
          continue;
        end if
        for min_2 = i + 1 to max_1 + 1 do
          d_1 = #(i, max_1, min_2, d)
          r_1 = x_{max_1+1} − x_{min_2}
          m = max(x_{max_1} − x_i + r, g[max_1 + 1][d_1][r_1])
          if m < g[i][d][r] then
            g[i][d][r] = m
          end if
        end for
      end for
    end for
  end for
end for
```

of $E(G'_1) + r$, $E(G'_2)$, ..., $E(G'_z)$ is minimized, then the partition $\{G_1, G_{2f} \cup G'_1, G'_2, \ldots, G'_z\}$ also forms an optimal (k, e)-anonymous partition of D.

Therefore, we study the following more general optimization problem:

PROBLEM 6.2. *Given d and r, obtain a partition $P = \{G_1, \ldots, G_m\}$ of D, where G_1 is $(k - d, e - r)$-anonymous and the rest groups are (k, e)-anonymous, such that $max(E(G_1) + r, E(G_2), \ldots, E(G_m))$ is minimized.*

Clearly, the minimum max-of-error problem is a special case of the above problem where $d = 0$ and $r = 0$.

The above argument shows that the above problem has the optimal substructure property. Intuitively, for each possible max_1 and min_2, we move as many values between x_{min_2} and x_{max_1} as possible to G_{2f}, the first part of G_2, as long as G_1 still has $k-d$ distinct values. After this step, we denote the number of distinct values in G_{2f} as $\#(min_1, max_1, min_2, d)$, as it is determined by these four parameters. Let $g(d, r, i)$ denote the minimum cost way to partition $x_i, \ldots, x_n$ such that the first group is $(k-d, e-r)$-anonymous and the rest groups are (k, e)-anonymous. Then we have $g(d, r, i) = min_{i \le u \le n, i \le v \le u-1} max(g(d-\#(i, u, v, d), r-(x_{u+1}-x_v), u+1), x_u - x_i + r)$

Algorithm 2 shows the dynamic programming solution to the minimum max-of-error problem. For simplicity, algorithm 2 only returns the maximum error of an optimal partition. With some simple bookkeeping, the algorithm can be easily modified to return the items contained in each group.

The purpose of our discussion so far is to show that the minimum max-of-error problem is in fact tractable. However, with complexity of $O(n^6)$, it is far from practical. Instead, it is very desirable to design efficient approximation algorithms for the problem. For this purpose, we limit our search space to those partitions whose groups do not overlap with each other. In other words, we consider the following problem:

PROBLEM 6.3. *Obtain a (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ of D, where the ranges of any two groups in the partition do not overlap, such that $max(E(G_1), \ldots, E(G_m))$ is minimized.*

We call the above problem the non-overlapping minimum max-of-error problem. With a similar argument to that of the optimal solution to the minimum sum-of-error problem, it is not hard to see that this problem also has the optimal substructure property, and thus can be solved by dynamic programming with $O(n^2)$ in both space and time complexities.

THEOREM 4. *Let P and P' be the optimal partitions of D of the minimum max-of-error problem and the non-overlapping minimum max-of-error problem. Then $min\_max\_error(P') \le 2 \cdot min\_max\_error(P)$.*

PROOF. (sketch) Let $P = \{G_1, \ldots, G_m\}$. According to lemma 6.2, overlapping ranges can only happen between adjacent groups. We examine each group by order. If the range of G_1 does not overlap with that of G_2, we continue to G_2. Otherwise, G_2 can be divided into two groups, G_{2f} which includes those no less than x_{min_3}, and G_{2l} which includes those greater than x_{min_3} (G_{2l} may be empty if the ranges of G_2 and G_3 do not overlap). We merge G_{2f} into G_1 and G_{2l} into G_3. Note that the range of G_3 does not change, while the range of G_1 is at most increased by $max\_of\_error(P)$. We next move to G_3 and check whether it overlaps with G_4. We continue this process until G_m. The resulting partition P' does not contain overlapping groups, and $max\_of\_error(P')$ is at most twice $max\_of\_error(P)$. □

7. EXPERIMENTS

Our experiments are conducted on the Adult Database from the UCI Machine Learning Repository [19]. The database is obtained from the US Census data, and contains 14 attributes and over 48,000 tuples. The same database has been used in previous works on k-anonymity and ℓ-diversity [10, 11]. We choose the same quasi-identifiers (which contain 8 attributes) in our experiments as that used in previous works. Since our approach focuses on numerical-valued sensitive attributes, in the experiments we choose "capital loss" as the sensitive attribute. In particular, we are interested in those people who do have capital loss. Therefore, we remove those tuples whose capital loss attributes are 0 or NULL. That leaves us with 1427 tuples. The range of capital loss in these tuples is from 155 to 3900, with 89 distinct values.

We also conduct experiments on a synthetic data set, so that we can adjust a variety of parameters to comprehensively evaluate the properties of the proposed permutation-based approach. The synthetic data set uses the same schema as the Adult Database. We populate the database with different numbers of tuples, assuming certain distribution of the capital loss attributes. We also consider the correlation between the capital loss attribute and quasi-identifiers. The details of the synthetic data set will be described later when we present the experimental results.

We design three sets of experiments to study the following aspects of different k-anonymity techniques.

Query Answering Accuracy. In this set of experiments, we compare the accuracy of the bounds derived from the generalized table and the permuted table. Specifically, let l and u be a lower and an upper bound of an aggregate query result r respectively. We define $error = (u - l)/r$ to be the relative error of the bounds. The smaller $error$ is, the more accurate the bounds are.

We also compare the accuracy of those bounds when using different optimization criteria to get (k, e)-anonymous partitions.

Query answering overhead. As described in section 5, to answer an aggregate query Q over a permuted database, we first

rewrite it into a query Q' which queries the permuted table, the mapping table and the help table. Our second set of experiments measure the running time of Q', and compare it with the time that it takes to execute Q over the un-permuted microdata table. The difference shows how much overhead our technique introduces to answer aggregate queries.

Scalability of partitioning algorithms. We have shown in section 6 that the optimal algorithm for the minimum sum-of-error problem and the approximation algorithm for the minimum max-of-error problem are of complexity $O(n^2)$. In this set of experiments, we empirically show their scalability when increasing the size of the microdata table.

Next, we describe each set of experiments in detail.

7.1 Accuracy

We first compare the relative error of the bounds derived from the generalized table and the permuted table, given the same partition. Specifically, the partition, which satisfies $(4, 0)$-anonymity (the same as 4-diversity in the ℓ-diversity work), is obtained by using the ℓ-diversity algorithm and the same generalization hierarchies reported in [11]. We note that the experiments in the work of ℓ-diversity actually computed 6-diverse partitions. However, after generalization using those 6-diverse partitions, a majority of quasi-identifiers (6 out of 8) including "age", "race", "native country", etc., are all generalized to "*", which essentially removes these attributes from the microdata table. This significantly limits the type of queries the generalized table can answer. To make the comparison more meaningful and in favor of the generalization-based approach, we instead choose 4-diverse partitions so that interesting attributes such as "age" can be generalized to reasonable domains. In fact, in order to prevent "age" from being generalized to "*" when using the ℓ-diversity algorithm, we also have to remove some of the outliers (6 tuples with age over 80). Otherwise, the "age" attribute will still be suppressed even with 4-diversity.

The resulting 4-diverse partition is composed of 25 groups. We first consider a general model of aggregate queries. Note that no matter what the query condition is, the result of the condition is to select a set of tuples in each group of the partition. Therefore, a query can be viewed as $\{T_1, \ldots, T_r\}$, where each T_i is a subset of $\mathcal{D}_i$. By selecting different tuples in each group, we may model arbitrary aggregate queries.

In the first experiment, we issue queries that randomly touch an arbitrary number of tuples from the table, and compute the average of their sensitive attribute values. We call these queries *arbitrary queries*. Arbitrary queries are representative of various disjunctive queries. From a generalized table, we cannot know for sure how many tuples are actually selected by a query in each group. To get deterministic lower and upper bounds, we have to assume either no tuples are included in the query, or only the tuple with the maximum capital-loss is selected. On the other hand, from a permuted table, we can always know exactly the number of tuples in each group that are selected by the query. Figure 6 shows the relative errors of the bounds derived from the generalized table and the permuted table. We see that the relative errors introduced by the generalized table is significantly higher than that by the permuted table. In fact, the bounds from the generalized table are often over two times of the actual query results. Further, as the total number of tuples selected increases, the relative error introduced by the permuted table drops dramatically, while that introduced by the generalized table does not drop at all.

We recognize that, since the arbitrary query is a very general model, the generalized table will not be able to take advantage of the semantics of a query condition. In the next experiment, we

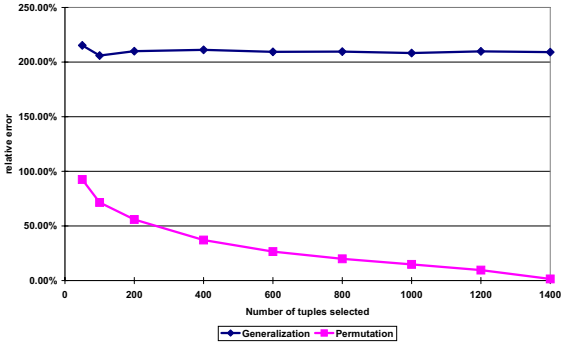

Figure 6: The relative errors of arbitrary queries when using the generalized table and the permuted table respectively

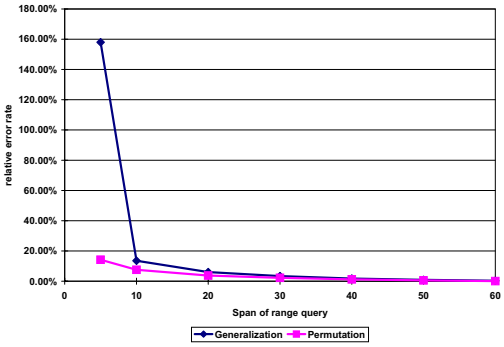

Figure 7: The relative errors of range queries when using the generalized table and the permuted table

issue aggregated queries over a certain range of the age attribute. In particular, we issue a sequence of queries of the form "select avg(capital_loss) from adult-table where age $\geq$ X and age $\leq$ Y", and vary the range $[X, Y]$. Clearly, for the generalized table, if the generalized value of the age attribute of a group falls completely in $[X, Y]$, then all the tuples in the group should be selected for the aggregation. Otherwise, we have to derive the lower and upper bounds of the group as in the first experiment. For the permuted table, we will still know the exact number of tuples selected in each group. The relative errors for range queries are shown in figure 7. We see that the accuracy of the bounds derived from both the generalized table and the permuted table improve for range queries. In particular, the relative error computed from the permuted table is less than 20% even when the span of the range query is only 5, which does not completely cover any groups in the partition. For the generalized table, the error is still very high when the range of the query is small, because some groups are only partially covered by the query. Since we do not know exactly how many tuples are selected by the range query in a partially covered group, the lower and upper bounds will be quite coarse. When the span of the range query increases, more and more groups are completely covered by the query. Thus, we observe a dramatic drop of relative errors for the generalized table. However, the error for the generalized table is still always higher than that for the permuted table.

The adult database contains 9 attributes as quasi identifiers. When we have less attributes, the hierarchy of generalization may be less coarse and lead to smaller partitions. To study the impact of dimensions on accurate query answering, we have also conducted experiments when assuming "age" is the only quasi identifier. We

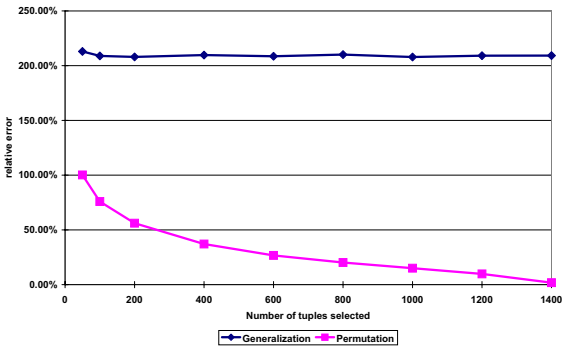

Figure 8: The relative errors of range queries when using the generalized table and the permuted table with only one QI

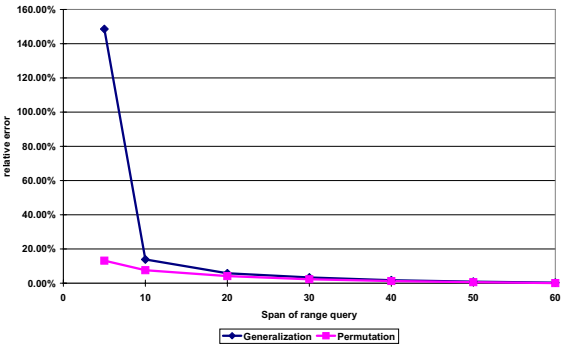

Figure 9: The relative errors of range queries when using the generalized table and the permuted table, with only one QI

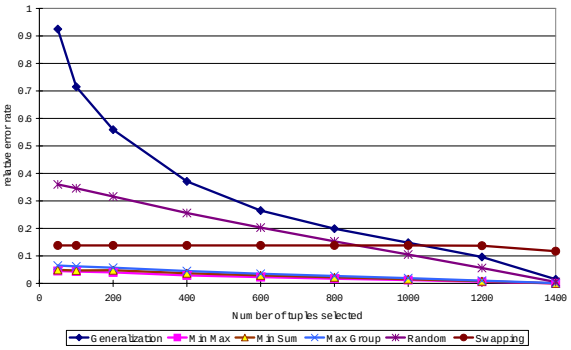

Figure 10: The relative errors of arbitrary queries when using different partitioning algorithms

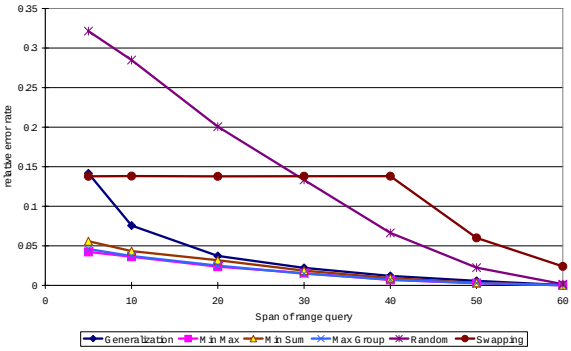

Figure 11: The relative errors of range queries using partitions from different partitioning algorithms

have observed very similar trends to that in Figures 6 and 7 (see figure 8 and 9). Our observation suggests that the poor query answering accuracy from the generalized table is often not caused by high dimension of quasi-identifiers. Instead, it is intrinsic to the domain generalization approach.

The way the microdata table is partitioned has a great impact on the accuracy of the bounds derived from the permuted table. Next, we compare the relative errors of permuted tables when using partitions generated by the following algorithms (see section 6): (1) ℓ-diversity based on domain generalization; (2) Min Max: the approximation algorithm for the minimum max-of-error problem; (3) Min Sum: the optimal algorithm for the minimum sum-of-error problem; (4) Max Group: an algorithm that generates the maximum number of groups in the partition. Similar to the minimum sum-of-error proble, the optimal algorithm can be obtained through dynamic programming; (5) a random algorithm: this algorithm sequentially scans each tuple. As long as the scanned tuples have k distinct sensitive attribute values, and its range is no less than e, they form a group of the partition. The random algorithm serves as a baseline for comparison; and (6) the rank-based data swapping algorithm [13]. To be comparable to (k, e) anonymity, given a tuple t whose rank is $rank(t)$, we select l such that the set of tuples with ranks in $[rank(t) - l, rank(t) + l]$ satisfies (k, e) anonymity. We then swap t with a random tuple whose rank is in the range.

We set $k = 4$ as before, and set $e = 100$ for the two optimization algorithms and the random partitions. To be in favor of the generalization-based approach, the groups in the partition generated by ℓ-diversity algorithm is only required to have no less than 4 distinct values. The parameters of the arbitrary queries and the range queries in this experiment are the same as in the first two experiments.

The relative errors corresponding to the above algorithms are shown in figure 10 (for arbitrary queries) and figure 11 (for range queries). It is clear that the three optimization algorithms introduce significantly less relative errors than the other two algorithms for arbitrary queries. This can be easily explained since the two optimization algorithms are not constraint by pre-defined domain hierarchies. They have more flexibility to partition the table and achieve better accuracy. We also observe that the partition from generalization is even worse than the random partition for arbitrary queries. This shows that the partition derived from pre-defined generalization hierarchies greatly reduces the utility of microdata, even if we use permutation-based anonymization. The rank-based data swapping algorithm, though with better accuracy than that from the partitions obtained through generalization hierarchy, still introduces much large relative errors than the the three optimization algorithm.

Intuitively, if there is a strong correlation between quasi-identifiers and the sensitive attribute, the tuples in the same group tend to have similar values in the partition generated through domain generalization. This may result in more accurate bounds for answering range queries. Our next experiment is to investigate the impacts of correlation on the accuracy of query answering. We compare the partitions obtained by the above four algorithms when varying the correlation between quasi-identifiers and the sensitive attribute. We run range queries that select tuples whose age attributes are in the range $[X, Y]$ where $Y - X = 30$. In the synthetic data set, we introduce a correlation between "age" and "cap-

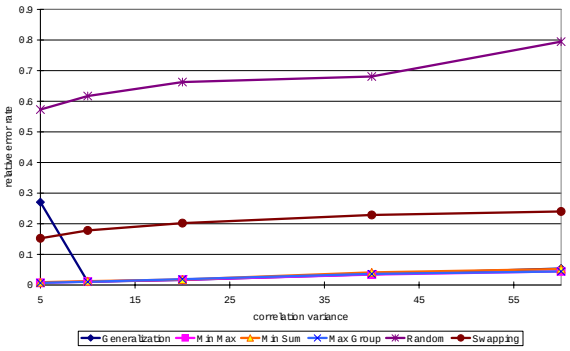

Figure 12: The relative errors of range queries using partitions from different partitioning algorithms, when the strength of the correlation between quasi-identifiers and the sensitive attribute varies.

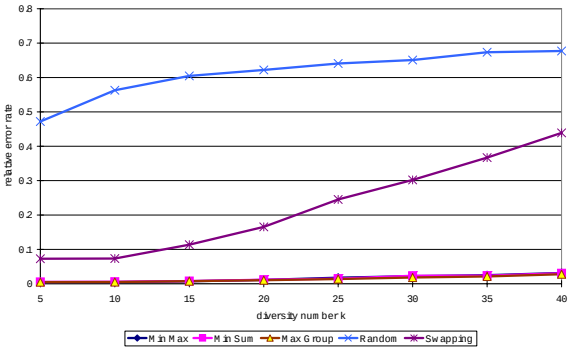

Figure 13: The relative errors of range queries using partitions from different partitioning algorithms

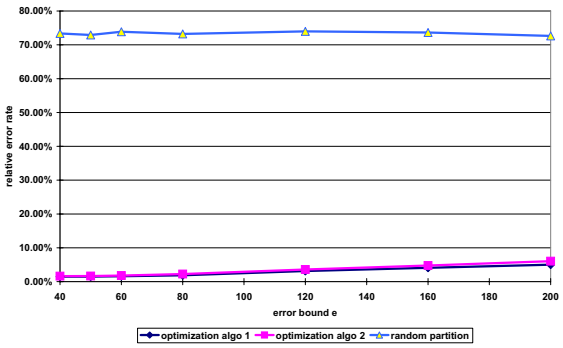

Figure 14: The relative errors of range queries using partitions from different partitioning algorithms

ital loss". The larger a tuple's age attribute, the larger its capital-loss, with a certain variance, which controls the strength of the correlation.

Figure 12 shows the relative errors of the six algorithms when the strength of correlation varies. We observe that as the correlation is strong(variance=5), tuples with the same age often have the same capital loss. Thus a higher generalization is needed, which causes the partition from generalization to yield a large error. As the variance goes up to 10, a lower generalization is sufficient since it is more likely for tuples in the same domain to have different capital loss values. That explains the quick drop of the error when variance=10. As the variance keeps increasing, tuples in the same group tends to have quite different sensitive attribute values, which will cause the error to increase. Since "age" is the only quasi identifier in the synthetic data set, a range query may completely cover many groups in the partition obtained through domain generalization. Therefore, it yields comparable accuracy with the partitions generated by the two optimization algorithms. The randomly algorithm does not take advantage of the correlation between "age" and "capital" loss, and thus performs poorly as expected. The rank-based data swapping algorithm does not perform very well because it is hard to deterministically reason the bounds of a query after data swapping. Only when a long sequence of consecutive sensitive attribute values are covered by a query, can we say for sure that some tuples are definitely included in the original query answers. This only happens frequently when a large portion of tuples are touched by a query.

Finally, we study the tradeoff between privacy and query answering accuracy. Intuitively, the larger k and e are, the more tuples each group in a (k, e)-anonymous partition tends to include, which will in turn introduces more errors when answering aggregate queries. To see the tradeoff more clearly, we run experiments over a synthetic random data set, whose quasi-identifier and sensitive attribute are "age" and "capital-loss" attributes respectively. The ranges of them are the same as in the adult database. We issue range queries over "age" attribute. The span of range queries is set to 30 as before. To examine the impact of the privacy parameter k, we fix $e = 50$, and vary k from 4 to 40. We measure the relative errors of the partitions obtained by the two optimization algorithms and the random partitioning for each k [1]. Figure 13 shows the experiment's results. We see that the accuracy for the partitions obtained by the three optimization algorithms are essentially the same, and do not deteriorate much as we increase k. This suggests that the optimization algorithms are capable to generate partitions that preserve high privacy while still supporting accurate aggregate query answering. Meanwhile, we see that the accuracy of the range-based data swapping approach decreases significantly, since it does not provide a mechanism to minimize the error introduced by data swapping.

We next evaluate the impact of the other privacy parameter e. In this experiment, k is fixed to be 10, and e varies from 40 to 200. Figure 14 shows the same trend as in figure 13: a much larger privacy requirement e does not impact the accuracy of the partitions obtained by the two optimization algorithms. As for the random algorithm, the increase of e does not affect the generated partition much, since random distinct values in the same group usually result in a range much larger than the privacy parameter e set by the experiment. The range-based data swapping approach does not perform well due to the same reason as explained in the previous experiment.

7.2 Query Execution Overhead

Given an aggregate query Q, we compare its execution time over the original un-permuted microdata table with that of the rewritten query Q' which performs selection and joins over the permuted table, the mapping table and the help table. To make the comparison more clearly, we run the experiments on the synthetic data set with 10000 tuples. The partition is $(20, 0)$-anonymous, and is obtained

[1]Partitions from generalization is not studied because the original ℓ-diversity does not have the parameter e. Moreover, we observe that when we require 6-diversity, in the real database "age" is already generalized to "*", and it cannot tell any information about a query on "age".

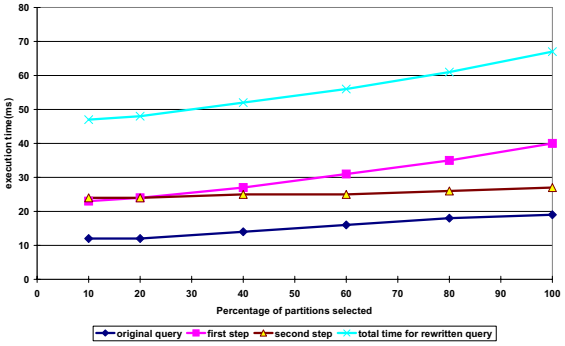

Figure 15: The execution time of the constrained query and the rewritten controlled query when the percentage of selected groups in the partition varies

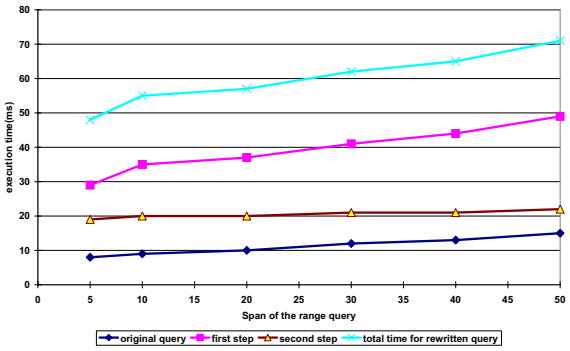

Figure 16: The execution time of the original range query and the rewritten range query when the span of the range varies

by the first optimization algorithm. There are totally 38 groups in the partition. Both arbitrary queries and range queries are tested.

In order to study the impact of the number of tuples and partitions involved in a query on query execution overhead, we use constrained arbitrary queries in this experiment: we vary the number of partitions involved in the query while having the percentage of tuples selected in each group fixed to be 30%. In figure 15, we show the time of the two steps when executing a rewritten query. The first step is to query the permuted table and join it with the mapping table so that the number of tuples selected in each partition is obtained. In the second step, the result from the first step is joined with the help table and the lower and upper bounds are computed. We also show the total running time of the rewritten query and compare it with the case if we run the original query directly on the un-anonymized microdata table. We see that, when the number of involved groups increases, so does the running time of both steps. For the first step, it is because the number of tuples selected in the first steps grows. For the second step, the more groups involved, the longer it takes to finish the join with the help table. The overall execution time of the rewritten query is about four times that of the original query.

The range query takes the same form as in the previous experiments. We increase the span of its range from 10 to 50. Intuitively, the larger the span of the range, the more groups of the partition and tuples will be selected by the query. Figure 16 presents the experiment's result, which is consistent with the experiment using constrained queries.

7.3 Scalability

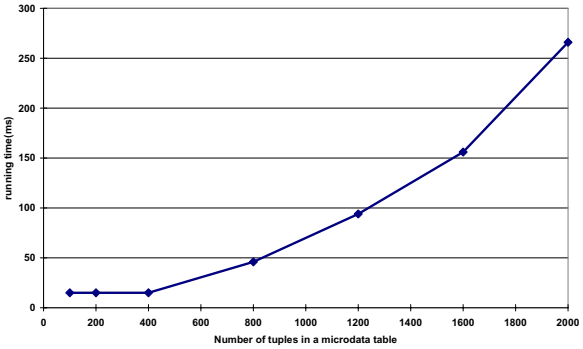

Figure 17: The running time of an optimal partitioning algorithm when the microdata table size scales up

We study the scalability of the optimization algorithms presented in section 6. All of them have computational complexity of $O(n^2)$, and have similar performance. Figure 17 shows the running time of the first partitioning optimization algorithm while the database size is varied from 100 tuples to 2000 tuples.

8. RELATED WORK

The privacy vulnerability of the release of de-identified microdata was first discussed by Sweeney [16, 18]. It has been shown that, after linking a de-identified medical database with voter registration records, some individual's medical record can be uniquely identified. Sweeney further proposed k-anonymity as a model for protecting privacy of microdata. Domain generalization and record suppression have been introduced as two techniques to achieve k-anonymity [17].

In [15], Samarati presented a framework for generalization and suppression based k-anonymity, where the concept of generalization hierarchies was formally proposed. Given a pre-defined domain hierarchy, the problem of k-anonymity is thus to find the minimal domain generalization so that, for each tuple t in the released microdata table, there exist at least k-1 other tuples which have the same quasi-identifiers as t. Samarati also designed a binary search algorithm to identify minimal domain generalizations. The concept of ℓ-diversity is introduced by Machanavajjhala et al. in [11] to prevent attackers with background knowledge.

It has been shown that the problem of general k-anonymity with suppression and arbitrary domain generalizations (instead of predefined generalization hierarchies) is NP-complete [12, 3, 9]. Several approximation algorithms have been proposed [2, 12]. In [20], Yao et al. show that, when several microdata tables are disclosed, even if each of them satisfies k-anonymity, by pooling them together, k-anonymity may be violated. They further design algorithms to detect such violations.

Recently, many works have been done to efficiently compute minimal and optimal generalizations [8, 10]. In [8], Bayardo and Agrawal presented a general model of the problem of finding optimal generalization and suppressions to achieve k-anonymity. The model can accommodate a variety of cost metrics. Pruning techniques have been proposed to reduce the search space of optimal generalization and optimization. In the Incognito approach of [10], generalization hierarchies are explored in a vertical way. It first computes the minimal solution to k-anonymity in the generalization hierarchy for each quasi-identifier. These solutions are then combined to form the candidate generalizations for the domain hi-

erarchies of quasi-identifier pairs. This process continues until a set of minimal domain generalizations are obtained for the full domains of quasi-identifiers. All the above works focus on introducing less imprecise information to microdata. But their impacts on the accuracy of aggregate queries are not discussed.

9. CONCLUSION

Privacy is a serious concern when sensitive information is released together with quasi-identifiers in microdata databases. A majority of previously proposed works focus on anonymizing microdata through domain generalization. Though privacy can be effectively protected by previous works, the impact of anonymization on ad hoc microdata analyses is rarely studied. We observe that in many situations, after domain generalization, the microdata becomes so general that it often has difficulty to answer aggregate queries with reasonable accuracy.

In this paper, we propose an extended privacy objective to better capture the protection of numeric-valued attributes in microdata. We also propose permutation based anonymization techniques. We show that we can achieve the same privacy guarantee as existing work when we partition a microdata table and perform random permutation between quasi-identifiers and sensitive attributes inside the groups of the partition. Further, since the quasi-identifiers of tuples remains in the anonymized table, aggregate queries can be answered much more accurately. We also design auxiliary relations and query rewriting algorithms to facilitate efficient ad hoc analyses over anonymized tables.

There are many interesting issues to be explored in the future. In particular, we are interested in investigating the use of permutation to achieve privacy under other privacy objectives. For example, instead of constraining the ranges of groups in a partition, another attractive objective is to require the difference between any two elements in a group to exceed a certain threshold. It is interesting to investigate efficient optimal or approximation algorithms under such privacy objectives, and study their impact on the accuracy of aggregate query answering.

10. REFERENCES

[1] N.R. Adam and J.C. Wortman. Security-Control Methods for Statistical Databases: A Comparat ive Study. *ACM Computing Surveys*, 21(4), 1989.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing Tables. In *International Conference on Database Theory*, January 2005.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panig raby, D. Thomas, and A. Zhu. k-Anonymity: Algorithms and Hardness. Technical report, Stanford University, 2004.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-Preserving Data Mining. In *Proceedings of the ACM SIGMOD International Conference on M anagement of Data*, Dallas, Texas, May 2000.

[5] T. Dalenius and S.P. Reiss. Data-swapping: A Technique for Disclosure Control. *Journal of Statistical Planning and Inference*, 6, 1978.

[6] Alexandre V. Evfimievski, Ramakrishnan Srikant, Rakesh A grawal, and Johannes Gehrke. Privacy Preserving Mining of Association Rules. In *Proceedings of the Eighth ACM SIGKDD International Conferen ce on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

[7] S.E. Fienberg and J. McIntyre. Data Swapping: Variations on a Theme by Dalenius and Reiss. In *Privacy in Statistical Databases*, Barcelona, Catalonia, June 2004.

[8] Roberto J. Bayardo Jr. and Rakesh Agrawal. Data Privacy through Optimal k-Anonymization. In *Proceedings of 21st IEEE International Conference on Data Engineering*, Tokyo, Japan, April 2005.

[9] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Multidimensional k-Anonymity. Technical Report 1521, University of Winsconsin, 2005.

[10] Kristen LeFevre, David DeWitt, and Raghu Ramakrishnan. Incognito - Efficient Full-Domain K-Anonymity. In *In Proceedings of ACM SIGMOD Conference*, Baltimore, MD, June 2005.

[11] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-Diversity: Privacy Beyond k-Anonymity. In *IEEE International Conference on Data Engineering*, Atlanta, GA, April 2006.

[12] Adam Meyerson and Ryan Williams. On the Complexity of optimal k-Anonymity. In *Proceedings of the 23rd ACM Symposium on the Principles of Database Systems*, 2004.

[13] R.A.Jr. Moore. Controlled Data-Swapping Techniques for Masking Public Use Microdata Sets. Technical Report Statistical Research Division Report Series, RR 96-04, US Bureau of Census, 1996.

[14] K. Muralidhar and R. Sarathy. Security of Random Data Perturbation Methods. *ACM Transactions on Database Systems*, 24(2), 1999.

[15] Periangela Samarati. Protecting Respondent's Privacy in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[16] Latanya Sweeney. Guaranteeing Anonymity When Sharing Medical Data, the Datafly System. *Journal of the American Medical Informatics Association*, pages 51–55, 1997.

[17] Latanya Sweeney. Achieving K-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[18] Latanya Sweeney. K-Anonymity, A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[19] U.C.Irvin Machine Learning Repository. http://www.ics.uci.edu/m̃learn/mlrepository.html.

[20] Chao Yao, Sean Wang, and Sushil Jajodia. Checking for k-Anonymity Violiation by Views. In *International Conference on Very Large Data Bases*, Trondheim, Norway, August 2005.