Dynamic Anonymization: Accurate Statistical Analysis with Privacy Preservation

Xiaokui Xiao Yufei Tao
Department of Computer Science and Engineering
Chinese University of Hong Kong
Sha Tin, New Territories, Hong Kong
{xkxiao, taoyf}@cse.cuhk.edu.hk

ABSTRACT

A statistical database (StatDB) retrieves only aggregate results, as opposed to individual tuples. This paper investigates the construction of a privacy preserving StatDB that can (i) accurately answer an infinite number of counting queries, and (ii) effectively protect privacy against an adversary that may have acquired all the previous query results. The core of our solutions is a novel technique called *dynamic anonymization*. Specifically, given a query, we on the fly compute a tailor-made anonymized version of the microdata, which maximizes the precision of the query result. Privacy preservation is achieved by ensuring that the combination of all the versions deployed to process the past queries does not allow accurate inference of sensitive information. Extensive experiments with real data confirm that our technique enables highly effective data analysis, while offering strong privacy guarantees.

ACM Categories and Subject Descriptors: H3.3 [Information Search and Retrieval]: Retrieval Models.

General Terms: Algorithms, Theory

Keywords: Privacy, Statistical Database, Dynamic Anonymization, m-invariance

1. INTRODUCTION

Privacy preservation has become a major issue in enabling public access to datasets that contain personal information. A bulk of research (see Section 2) in this area has been devoted to *central publication*. In that scenario, a publisher aims at releasing an anonymized version $T^\diamond$ of a *microdata* table T, so that researchers can use $T^\diamond$ to derive statistical results about T, whereas no malicious user, called an *adversary*, can infer the sensitive data of any individual from $T^\diamond$.

As an example, consider the microdata T in Table 1a, which contains three attributes *Age*, *Zipcode*, and *Disease*. Each tuple corresponds to an individual (the column *Name* is not part of T; we use it to facilitate tuple referencing). Attribute *Disease* is sensitive; namely, no adversary should be able to figure out the disease of any individual with high confidence. For this purpose, the publisher releases T_1^* in Table 1b. Here, anonymization is performed with *generalization*, which makes it impossible to uniquely pinpoint the tuple owned by an individual from her/his non-sensitive values. For instance, assume that an adversary knows the age 20 and Zipcode 12000 of Alice. Given T_1^*, the adversary can only find out that Alice is described by the first or second tuple. Hence, with a random guess, the adversary can infer her real disease *flu* only with 50% chance. Since the non-sensitive columns can be used to identify an individual, they are also called the *quasi-identifier* (QI) attributes.

The above process carried out by the adversary is known as a *linking attack* [32, 33]. Such attacks are a major threat of privacy that the database community tackles nowadays. Besides generalization, this threat can also be prevented by another anonymity framework named *anatomy* [38]. We will explain the characteristics of both frameworks in Section 3.1.

1.1 Statistical Databases

This paper advocates a technique different from central publication: *statistical databases* (StatDB), which retrieve only aggregate results, as opposed to individual tuples. Privacy is protected by controlling the precision of the returned answers. The existing approaches (see [1] for an excellent survey) can be classified into three categories: query restriction, output perturbation, and data modification.

Query restriction [17, 26] simply refuses to answer certain queries whose results leak sensitive information. Since the answers of multiple queries may be leveraged (sometimes, in intricate ways) to infer privacy, the system must remember all the past queries, and examine them in deciding the answerability of a new query. Therefore, as time evolves, the space consumption (for the query-log) monotonically escalates, and the cost of judging answerability becomes increasingly expensive. *Output perturbation* [7, 10, 9] works as follows. Given a query, the system finds its exact answer, adds certain noise to (i.e., perturbs) it, and then returns only the perturbed value. Ideally, the noise should be small to ensure the answer's utility, and its generation should make it extremely hard for an adversary to de-noise. This, however, turns out to be difficult. Dinur and Nissim [9] prove that, to safeguard privacy, the system is allowed to answer only a finite number of queries, which is a function of the privacy requirement and the target precision of query results.

Given a microdata table T, *Data modification* [4, 25] prepares an adequately anonymized version $T^\diamond$ of T, such that

SIGMOD'08, June 9–12, 2008, Vancouver, BC, Canada.

Name	Age	Zip.	Disease
Alice	20	12000	flu
Bob	23	58000	gastritis
David	38	41000	flu
Helen	42	23000	gastritis
Jack	46	25000	flu
Ken	48	13000	gastritis
Linda	49	51000	flu
Mary	52	52000	insomnia
Paul	53	49000	gastritis
Ray	59	61000	flu
Tom	61	39000	gastritis

(a) Microdata T

Tuple ID	Age	Zip.	Disease
1 (Alice)	[20, 23]	[12k, 58k]	flu
2 (Bob)	[20, 23]	[12k, 58k]	gastritis
3 (David)	[38, 42]	[23k, 41k]	flu
4 (Helen)	[38, 42]	[23k, 41k]	gastritis
5 (Jack)	[46, 48]	[13k, 25k]	flu
6 (Ken)	[46, 48]	[13k, 25k]	gastritis
7 (Linda)	[49, 53]	[49k, 52k]	flu
8 (Mary)	[49, 53]	[49k, 52k]	insomnia
9 (Paul)	[49, 53]	[49k, 52k]	gastritis
10 (Ray)	[59, 61]	[39k, 61k]	flu
11 (Tom)	[59, 61]	[39k, 61k]	gastritis

(b) Generalization T_1^*

Table 1: Microdata and its generalization

no privacy breach can occur, even if an adversary has the entire $T^\diamond$. For each query, the system directly returns its exact result on $T^\diamond$ (note: not on T). Data modification avoids the drawbacks of the previous two StatDB methodologies. First, (unlike query restriction), it can answer any query, and eliminates the need of auditing and retaining the historical queries. Second, (unlike output perturbation), it is able to process an infinite number of queries.

1.2 StatDB vs. Central Publication

A StatDB has at least two major advantages over central publication. First, a StatDB allows us to build an online server with a user-friendly interface, so that any user (including those with limited scientific backgrounds) around the world can easily query about the microdata. Specifically, the server receives the query parameters through the interface, runs the underlying StatDB, and displays the result in the user's web browser [12]. On the other hand, an anonymized dataset released by central publication is more difficult to analyze. For example, given Table 1b, a user still needs to write a program to obtain a query result, which would not be possible for "naive users" without knowledge of programming. Second, given a query, a StatDB may produce a more accurate answer than can be derived from a directly-published dataset, as will be clarified in the next subsection.

Note that we do not imply that StatDB is superior to central publication in all aspects. In fact, the latter clearly has its own advantages too. An important one, for example, is that, possessing an anonymized dataset, a real expert can perform a large variety of studies, instead of being confined to the functionalities provided by a server. Our intention, instead, is to establish StatDB as another feasible approach for enabling public access to private data, which has its unique merits.

Tuple ID	Age	Zip.	Disease
1 (Alice)	[20, 48]	[12k, 13k]	flu
2 (Ken)	[20, 48]	[12k, 13k]	gastritis
3 (Jack)	[42, 46]	[23k, 25k]	flu
4 (Helen)	[42, 46]	[23k, 25k]	gastritis
5 (David)	[38, 61]	[39k, 41k]	flu
6 (Tom)	[38, 61]	[39k, 41k]	gastritis
7 (Linda)	[49, 53]	[49k, 52k]	flu
8 (Mary)	[49, 53]	[49k, 52k]	insomnia
9 (Paul)	[49, 53]	[49k, 52k]	gastritis
10 (Ray)	[23, 59]	[58k, 61k]	flu
11 (Bob)	[23, 59]	[58k, 61k]	gastritis

Table 2: Another generalized version T_2^* of T

1.3 Dynamic Anonymization

Our objective is to design a StatDB that achieves a standard of privacy preservation acceptable by the central publication literature (i.e., m-invariance [39], which is a stricter version of l-diversity[1] [22]). There exist alternative privacy standards in other fields (e.g., security, surveying, etc.), and they are not our focus. In any case, each standard has its pros and cons. However, as long as a standard (such as l-diversity) has been justified to be useful in many applications, a technique enforcing it has practical importance.

It is easy to come up with a straightforward solution. Given a microdata table T, we can compute an l-diverse table $T^\diamond$, and use it to process queries. This paper presents a new technique, *dynamic anonymization*, which offers better query answers, while ensuring strong privacy protection. Specifically, given a query, we (on the fly) produce, among all the possible anonymized versions of T, a good version, which provides a highly accurate query result. Let Q_i be the i-th query, and $T_i^\diamond$ the anonymized relation deployed to answer Q_i. We guarantee that, even if an adversary acquires the entire set $\{T_1^\diamond, T_2^\diamond, ...\}$, s/he still cannot derive any sensitive data with high confidence. Privacy protection is thus ensured, noting that the most an adversary can get from the result of Q_i is $T_i^\diamond$. Furthermore, our technique has an interesting feature: no information of $T_i^\diamond$ needs to be remembered after processing Q_i.

Let us illustrate the benefits of dynamic anonymization using the microdata T in Table 1a, again employing generalization as the anonymity approach. The first query received by our StatDB is:

Q_1: `SELECT COUNT(*) FROM` StatDB
`WHERE` $Age \in [30, 50]$ `AND` $Disease =$ flu

Suppose that the system chooses the generalized relation T_1^* in Table 1b to process Q_1. The returned result is an interval $[2, 3]$, which encloses the real answer 3 of Q_1. In particular, the result follows the fact that Tuples 3 and 5 in Table 1b *definitely* satisfy Q_1, Tuple 7 *possibly* satisfies Q_1, and the other tuples *cannot* satisfy Q_1, judging from their (generalized) ages and diseases. For instance, Tuple 7 may

[1] l-diversity requires that, in each QI-group, at most $1/l$ of its tuples can have the same sensitive value. We will introduce m-invariance later.

satisfy Q_1 because, from a user's perspective, the original *Age*-value of this tuple may be anything between 49 and 53.

Consider another query:

Q_2: `SELECT COUNT(*) FROM` StatDB
`WHERE` *Zipcode* $\in$ [20k, 40k] `AND` *Disease* = flu

If T_1^* (i.e., Table 1b) is used to process Q_2, by the earlier reasoning, the result is a wide interval $[0, 4]$, since no tuple definitely satisfies Q_2, and Tuples 1, 3, 5, 10 possibly satisfy. Now we compute an alternative generalized version T_2^* of T, as shown in Table 2. Note that the tuple sequence in Table 2 differs from that in Table 1a (the correspondence between the two sequences is indicated by people's names). Answering Q_2 with T_2^* produces a shorter interval $[1, 2]$ (Tuples 3 and 5 definitely and possibly satisfy, respectively). This result is more useful than that from T_1^*.

Even if an adversary has somehow deduced the entire T_1^* and T_2^* (from the query results), equipped with a victim individual's QI-values, s/he can correctly infer the victim's disease only with probability 50%. The secret is that the set $\{T_1^*, T_2^*\}$ obeys *2-invariance* [39]. To explain this concept, let us define (informally) a *QI-group* in a generalized table as a maximal set of tuples with equivalent QI-values. 2-invariance states that, given any tuple t in the microdata T, the two QI-groups in T_1^* and T_2^* containing the generalized versions of t must (i) have an identical size, (ii) contain the same set of *Disease*-values, and (iii) in each QI-group, no *Disease*-value appears more than once. For example, if t is the tuple of Alice in Table 1a, then the QI-group in T_1^* (T_2^*), which t is generalized to, includes the first two rows of Table 1b (Table 2). It can be easily verified that Conditions (i)-(iii) are indeed fulfilled. In general, m-invariance ensures that a linking attack can discover the sensitive value of an individual with probability at most $1/m$ [39].

The previous examples also explain why StatDB promises better data analysis than central publication. As mentioned earlier, Table 1b leads to a good result for query Q_1 but a poor one for Q_2. Interestingly, the reverse is true for Table 2: although as shown earlier it yields a tight interval for Q_2, it produces a long interval $[1, 5]$ for Q_1 (Tuple 3 definitely satisfies Q_2 and Tuples 1, 5, 7, 9 may satisfy). Hence, the two queries cannot be accurately processed simultaneously, no matter which of Tables 1b and 2 is published.

1.4 Contributions and Paper Organization

This paper carries out a systematic study on building a privacy preserving StatDB. First, we formalize a new type of StatDB, based on the concepts of dynamic anonymization and m-invariance. Our formalization applies to two anonymization frameworks: generalization and anatomy. In either case, our StatDB provides rigorous privacy guarantees, against even the most persistent adversary that has audited the results of all past queries.

As a second step, we analyze the algorithmic and theoretic issues arising from each anonymization framework. Interestingly, when anatomy is deployed, for any query Q, the optimal anonymized version of the microdata T (providing the most accurate answer to Q) can be computed by scanning a fraction of T. Finding the optimal generalized version, unfortunately, is NP-hard. Therefore, we develop a fast heuristic algorithm that returns accurate answers.

Finally, we perform extensive experiments to verify the effectiveness of the proposed techniques. Specifically, our StatDB enables highly effective data analysis, while providing strong privacy guarantees. Furthermore, the StatDB is both space- and time-economical. Specifically, it requires a single hash structure with the same size as T (as mentioned in Section 1.3, *none* of the anonymized versions employed in history needs to be retained), and settles every query in less than 0.05 seconds (when T is a real dataset with 600k tuples).

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 explains the concepts of generalization, anatomy, and m-invariance. Section 4 formally defines the problem, and overviews the proposed solutions. Section 5 presents a technique to concisely capture all the possible anonymized versions of the microdata that can be used to answer a query without privacy breaches. Section 6 clarifies the details of a StatDB adopting anatomy, whereas Section 7 extends the discussion to generalization. Section 8 explores alternative approaches to implement a privacy preserving StatDB. Section 9 experimentally evaluates the effectiveness of our techniques. Finally, Section 10 concludes the paper with directions for future work.

2. RELEVANCE TO PREVIOUS WORK

In the following, we focus on four topics in data anonymization that are most relevant to our work, and explain the differences between the existing solutions and the proposed technique.

Statistical Databases. A brief survey of StatDB has been provided in Section 1.1, explaining the three main categories of previous search. We focus on *data modification* because (i) it encompasses our solutions, and (ii) the other categories either cannot support arbitrary queries or can answer only a limited number of them, as explained in Section 1.1.

There are three major methodologies of data modification: *random perturbation* [4, 11], *random noise* [15, 25], and *data swapping* [30]. We will elaborate on random perturbation in Section 8 and adapt it to our scenario. Random-noise offsets every value in the microdata by a certain extent, so that the resulting dataset is different from and yet retains the patterns in the original data. Data swapping, on the other hand, distorts the dataset by exchanging the values among tuples. For example, exchanges can be performed in a way that preserves the "marginal statistics" [30] of the microdata. Methods of both random-noise and data-swapping are often accompanied by theoretical privacy guarantees, which, however, are valid in scenarios that do not involve linking attacks. Hence, until specialized guarantees against such attacks have been established, those methods remain inapplicable to prevention of this type of privacy inferences.

Central Publication. This area primarily aims at preventing linking attacks, as discussed earlier with Tables 1a and 1b. Most solutions adopt generalization [3, 8, 16, 18, 21, 22, 23, 24, 28, 32, 33, 37, 39] or anatomy [13, 40, 38] as their anonymity frameworks. Implementation of either framework must be integrated with a *privacy principle*, for deciding if anonymization fulfills the privacy needs. Several principles have been proposed, and they offer various anonymity guarantees targeting different background knowledge of adversaries. Examples include k-anonymity [32, 33], l-diversity [22], t-closeness [21], m-invariance [39], δ-presence [27], etc. Perturbation has also been applied to central publication [29].

LeFevre et al. [20] develop an interesting *workload-aware* approach. This method assumes the availability of a "representative query workload", and publishes a generalized table that maximizes the accuracy of the workload. Apparently, the applicability of the method is limited when the query distribution is not known in advance. Our dynamic anonymization technique remedies this problem, since it finds a tailor-made anonymized relation for every query on the fly, without having to acquire the query in advance.

Cryptographic Computing. An important topic in cryptography is to develop a protocol of data exchange, such that the amount of information exposed exceeds the amount needed for performing an objective *task* by a provably small threshold. Various protocols exist for tasks such as clustering [34], top-k search [35], etc. Cryptographic computing is inherently different from StatDB (the focus of our paper) in two ways. First, a cryptographic protocol deals with multiple parties, and minimizes the information revealed by each party. The issue does not exist in a StatDB, where there is only one party holding the sensitive data (i.e., the server). Alternatively, from the perspective of cryptography, the "protocol" in a StatDB is trivial: given a query (a.k.a., a task), the server just returns the answer directly, which is obviously the least information for settling the query. Note that there exist cryptographic solutions [31] for secure *delivery* of secret contents from a single party to a recipient. These solutions are also different from multi-party cryptographic protocols, and are complement to StatDB for result transmission.

Second, their privacy goals are different. Disseminating the query result is *intended* in a cryptographic protocol (recall that the protocol tries to reduce the released information *outside* the result), while this breaches privacy in a StatDB. For example, consider the cryptographic protocol of [35] that reports the top-k tuples in the union of the data in several parties. These tuples are revealed *directly*, and it is *not* an issue even if they contain sensitive information. In other words, the protocol cares about exchanging no unnecessary information among parties, rather than whether the data in those objects is private . A StatDB prevents accurate inference of any tuple, and prohibits direct publication of any data. In particular, not only that a StatDB cannot release the exact result of any query, but also it must ensure no confident privacy inference through the correlation in the results of all past queries. Hence, the security model of a cryptographic protocol is no longer suitable in StatDB.

It is important to understand the differences between privacy principles (e.g., l-diversity) and cryptographic protocols in privacy protection abilities. Unlike those applying cryptographic protocols, an application (e.g., central publication and, in our paper, StatDB) enforcing a privacy principle typically does not have a concrete query in mind, but instead, targets generic ad-hoc analysis. A privacy principle assumes less powerful adversaries compared to a cryptographic protocol. This is inevitable because anonymity requirements need to be lowered to improve the utility of anonymized data for broader purposes. However, this does not affect the usefulness of privacy principles, as long as their assumptions are acceptable in the underlying application. Note that "acceptable" does not mean that these assumptions hold against all adversaries (after all, no principle, or even cryptographic protocol, can achieve this), but rather, a significant number of them.

Access Control. In practice, various users have different access permissions to different portions of a database. When the number of users is exceedingly large, enforcing those permissions effectively and efficiently becomes a serious challenge. There have been several solid works for tackling this problem in different environments, e.g., relational databases [36], XML documents [6], temporal databases [5], and so on. Anonymity in StatDB, however, is different from access control. In particular, each query has access to the entire database, except that a fuzzy result (e.g., a range) is returned for privacy preservation.

3. PRELIMINARIES

Let T be a microdata table that contains a sensitive attribute A^s, and d quasi-identifier (QI) attributes A_1^q, ..., A_d^q. Following a common assumption in the literature [22, 23, 38], A^s is categorical, while every A_i^q $(1 \leq i \leq d)$ can be either numerical or categorical. For each tuple $t \in T$, we denote its value on an attribute A as $t[A]$. In the sequel, we will first discuss generalization and anatomy in Section 3.1, and then the m-invariance principle in Section 3.2.

3.1 Anonymization Frameworks

We target adversaries with the following background knowledge.

Definition 1 (QI-conscious Adversary). A *QI-conscious adversary* knows the identity of every individual in T, and her/his QI values. □

Both generalization and anatomy are based on QI-groups:

Definition 2 (QI-Group / Partition). A *QI-group* of T is a subset of the tuples in T. A *partition* of T is a set of disjoint QI-groups whose union equals T. □

Next we define generalization via a *bijection*:

Definition 3 (Generalization [32, 33]). A *generalization* T^* has the same attributes as T, and is defined by a partition P of T. There exists a bijection f from T to T^*, such that

1. For each $t \in T$ and $t^* = f(t)$, $t^*[A_i^q]$ is an interval[2] covering $t[A_i^q]$ $(1 \leq i \leq d)$, and $t^*[A^s] = t[A^s]$.
2. For any tuples t_1 and t_2 in a QI-group of P, $f(t_1)$ and $f(t_2)$ have the same value on all A_i^q $(1 \leq i \leq d)$. □

For any QI-group $G \in P$, we define $G^* = \{t^* | t^* = f(t), t \in G\}$ as a QI-group in T^*, and use $G^*[A_i^q]$ to denote the generalized A_i^q value of the tuples in G^*. For example, Table 1b stems from the following partition of Table 1a:

$$P_1 = \{\{\text{Alice, Bob}\}, \{\text{David, Helen}\}, \{\text{Jack, Ken}\}, \{\text{Linda, Mary, Paul}\}, \{\text{Ray, Tom}\}\}.$$

The bijection between the two tables is reflected by people's names. Similarly, Table 2 is created from another partition:

[2]If A_i^q is categorical, we impose on its values a total ordering, which lists the leaves of its generalization hierarchy [18] from left to right.

Tuple ID	Age	Zip.	G. ID
1 (Alice)	20	12000	1
2 (Bob)	23	58000	1
3 (David)	38	41000	2
4 (Helen)	42	23000	2
5 (Jack)	46	25000	3
6 (Ken)	48	13000	3
7 (Linda)	49	51000	4
8 (Mary)	52	52000	4
9 (Paul)	53	49000	4
10 (Ray)	59	61000	5
11 (Tom)	61	39000	5

(a) The quasi-identifier table

G. ID	Disease	Count
1	flu	1
1	gastritis	1
2	flu	1
2	gastritis	1
3	flu	1
3	gastritis	1
4	flu	1
4	insomnia	1
4	gastritis	1
5	flu	1
5	gastritis	1

(b) The sensitive table

Table 3: The QIT and ST generated from the partition underlying Table 1b

$$P_2 = \{\{\text{Alice, Ken}\}, \{\text{Jack, Helen}\}, \{\text{David, Tom}\}, \{\text{Linda, Mary, Paul}\}, \{\text{Ray, Bob}\}\}.$$

Anatomy can also be formalized using a bijection:

DEFINITION 4 (ANATOMY [40, 38]). Given a partition P of T with n_g QI-groups, an *anatomy* of T includes a *quasi-identifier table* (QIT) and a *sensitive table* (ST). The QIT contains all the QI attributes in T and an attribute "Group-ID". There is a bijection f from T to the QIT, such that,

1. For any tuple $t \in T$ and $t' = f(t)$, $t[A_i^q] = t'[A_i^q]$ for all $i \in [1, d]$.
2. If t is in the j-th ($1 \leq j \leq n_g$) QI-group of P, then t'[Group-ID] equals j.

The ST has three attributes: "Group-ID", A^s, and "Count". A tuple t in the ST indicates that the t[Group-ID]-th QI-group of P has t[Count] tuples with sensitive value $t[A^s]$. □

For instance, given partition P_1 of Table 1a, the anatomy contains the QIT and ST in Table 3. Again, the bijection between Table 1a and the QIT is indicated by names.

As proved in [38], the two anonymization frameworks provide the same protection against QI-conscious adversaries (Definition 1). However, anatomy allows more accurate data analysis, since it enables users to obtain precisely the QI values within each QI-group (see a detailed discussion in [38]). Nevertheless, sometimes disclosing QI values directly may not be acceptable, if the presence of an individual in the microdata is also considered confidential [27]. In this case, generalization should be employed. Since generalization and anatomy are useful in different applications, we consider both frameworks in constructing a StatDB. For convenience, the term *anonymized version* $T^\diamond$ of T will be used, no matter whether $T^\diamond$ is obtained using generalization or anatomy.

3.2 m-Invariance

Recently, a new anonymization principle called *m-invariance* is proposed in [39] to facilitate publication of multiple anonymized versions of the same microdata. The principle is based on two concepts: signature and m-uniqueness.

DEFINITION 5 (SIGNATURE). Let P be a partition of T, and t be a tuple in a QI-group $G \in P$. The *signature* of t in P is the set of distinct sensitive values in G. □

For example, in Table 1b, Alice's tuple belongs to a QI-group with two distinct sensitive values: flu and gastritis. Therefore, the signature of the tuple is {flu, gastritis}.

DEFINITION 6 (m-UNIQUENESS). An anonymized version $T^\diamond$ is *m-unique*, if $T^\diamond$ is generated from a partition, where each QI-group contains at least m tuples, each with a different sensitive value. Such a partition is also said to be m-unique □

Both P_1 and P_2 in Section 3.1 are 2-unique. Now we are ready to clarify m-invariance:

DEFINITION 7 (m-INVARIANCE). A set S of partitions is *m-invariant* if:

1. Each partition in S is m-unique.
2. For any partitions $P_1, P_2 \in S$, and any tuple $t \in T$, t has the same signature in P_1 and P_2.

A set of anonymized versions of T is m-invariant, if the partitions underlying them constitute an m-invariant set. □

Since $\{P_1, P_2\}$ is 2-invariant, so is {Table 1b, Table 2}. The privacy guarantee of m-invariance is established by:

LEMMA 1 ([39]). *Given an m-invariant set of anonymized versions of T, a QI-conscious adversary has at most $1/m$ confidence in inferring the sensitive value of any individual in T.*

4. PRIVACY PRESERVING STAT. DB.

Our objective is to support counting queries of the form

```
SELECT COUNT(*) FROM StatDB
WHERE p(A_1^q) AND ... AND p(A_d^q) AND p(A^s)
```

where $p(A^s)$ is any predicate on A^s, and $p(A_i^q)$ has the format

$$\text{“}A_i^q \in (-\infty, \infty)\text{” or “}A_i^q \in [x_i, y_i]\text{”},$$

where x_i and y_i are two values in the domain of A_i^q. $p(A_1^q)$, ..., $p(A_d^q)$ are the *QI predicates*, and $p(A^s)$ the *sensitive predicate*. Consider, for example, the Q_1 in Section 1.3, and the microdata in Table 1a. Let A_1^q be *Age* and A_2^q be *Zipcode*. Then, $p(A_1^q)$ is $Age \in [30, 50]$. Since Q_1 contains no condition on *Zipcode*, we set $p(A_2^q)$ to $Zipcode \in (-\infty, \infty)$. Finally, $p(A^s)$ is $Disease = \text{flu}$.

Given a counting query Q, the StatDB identifies an anonymized version $T^\diamond$ of T, and uses it to answer Q. The following concept is needed to define the answer.

DEFINITION 8 (POSSIBLE MICRODATA INSTANCE). Given an anonymized version $T^\diamond$ of T, a table T' is a *possible microdata instance*, if T' can be anonymized into $T^\diamond$.

For example, imagine that the first two rows in Table 1a are replaced with $\langle 20, 12000, \text{gastritis}\rangle$ and $\langle 23, 58000, \text{flu}\rangle$ respectively, the resulting table is a possible microdata instance of Table 1b. From the perspective of a user, who can access only (part of) $T^\diamond$ through queries, any possible microdata instance may be the actual microdata. Hence, the answer of a query on $T^\diamond$ should be defined by taking into account all the instances:

DEFINITION 9 (RESULT INTERVAL). Given a counting query Q, and an anonymized version $T^\diamond$ of T, the *result interval of* Q on $T^\diamond$ is $[r_\vdash, r_\dashv]$, where $r_\vdash$ ($r_\dashv$) is the smallest (largest) answer for Q on any possible microdata instance of $T^\diamond$. □

Interval $[r_\vdash, r_\dashv]$ always contains the actual result of Q on T, since T itself is a possible microdata instance. Apparently, a shorter interval is more useful. In Section 1.3, we exemplified how to obtain result intervals, while the next lemma formalizes the computation for any generalized relation.

LEMMA 2. *Let $[r_\vdash, r_\dashv]$ be the result interval of a query Q on a generalization T^* of T. Given a QI-group G^* in T^*, use $s(G^*)$ to denote the number of tuples in G^* satisfying $p(A^s)$. Then,*

$$r_\vdash = \sum_{G^* \in F_\vdash(T^*)} s(G^*), \text{ and } r_\dashv = \sum_{G^* \in F_\dashv(T^*)} s(G^*),$$

where $F_\vdash(T^*)$ and $F_\dashv(T^*)$ are sets of QI-groups in T^*, such that

1. *For any $G^* \in F_\vdash(T^*)$, every value in the interval $G^*[A_i^q]$ satisfies $p(A_i^q)$ for all $i \in [1, d]$.*
2. *For any $G^* \in F_\dashv(T^*)$, some values in $G^*[A_i^q]$ satisfy $p(A_i^q)$ for all $i \in [1, d]$, and not all the values in $G^*[A_i^q]$ satisfy $p(A_i^q)$ for at least one $i \in [1, d]$.*

We omit the proofs from this paper due to the space constraint. To explain functions $F_\vdash(.)$, $F_\dashv(.)$, and $s(.)$, assume that T^* is the T_1^* in Table 1b, and Q is the query Q_1 in Section 1.3. Let G_1^* be the QI-group involving Tuples 3 and 4 in T_1^*. $G_1^* \in F_\vdash(T_1^*)$, because all values in $G_1^*[Age] = [38, 42]$ and $G_1^*[Zipcode] = [12k, 58k]$ satisfy $p(Age)$ and $p(Zipcode)$, respectively. $s(G_1^*)$ equals 1, as the *Disease*-value of only one tuple in G_1^* fulfills $p(A^s)$. As another example, let G_2^* be the QI-group involving Tuples 7, 8 and 9. $G_2^* \in F_\dashv(T_1^*)$, because although both $G_2^*[Age]$ and $G_2^*[Zipcode]$ satisfy $p(Age)$ and $p(Zipcode)$ respectively, some values in $G_2^*[Age]$ do not satisfy $p(Age)$.

Next, we provide a similar result for anatomy.

LEMMA 3. *Let $[r_\vdash, r_\dashv]$ be the result interval of Q on a pair of QIT and ST generated from a partition P of T. For any QI-group G, use $q(G)$ and $s(G)$ for the numbers of tuples in G satisfying the QI and sensitive predicates in Q, respectively. We have*

$$r_\vdash = \sum_{G \in P} h_\vdash(G), \text{ and } r_\dashv = \sum_{G \in P} h_\dashv(G),$$

where $h_\vdash(.)$ and $h_\dashv(.)$ are two functions defined as follows:

$$h_\vdash(G) = \max\{0,\ q(G) + s(G) - |G|\}, \quad (1)$$

$$h_\dashv(G) = \min\{q(G),\ s(G)\}. \quad (2)$$

To clarify $h_\vdash(.)$, $h_\dashv(.)$, $q(.)$, and $s(.)$, imagine that Q is the Q_1 in Section 1.3, and P is the partition of the microdata Table 1a defining the QIT and ST in Table 3. Let G be the QI-group with Group-ID 4 in the QIT. Then, $q(G)$ equals 1, because only the tuple of Linda satisfies the QI predicates, whereas $s(G)$ is also 1, since, as shown in the ST, G has one tuple whose *Disease*-value satisfies the sensitive predicate. Therefore, $h_\vdash(G) = \max\{0, 1 + 1 - 3\} = 0$ and $h_\dashv(G) = \min\{1, 1\} = 1$.

DEFINITION 10 (STATDB). Given a microdata table T and a parameter m, a *privacy preserving statistical database* $\mathcal{D}$ satisfies the following conditions:

1. Given the i-th query Q_i, $\mathcal{D}$ returns a result interval derived from an anonymized version $T_i^\diamond$ of T.
2. $\{T_1^\diamond, T_2^\diamond, T_3^\diamond, ...\}$ is m-invariant. □

Since our StatDB enforces m-invariance, by Lemma 1, we know:

COROLLARY 1. *A QI-conscious adversary has at most $1/m$ confidence in inferring the sensitive value of any individual in T, even if s/he is allowed to issue an infinite number of counting queries to a privacy preserving statistical database.*

Interestingly, to fulfill Condition 2 of Definition 10, it is not necessary to check the m-invariance of a long sequence of anonymized versions. Instead, we can check only two versions at a time:

LEMMA 4. *$\{T_1^\diamond, T_2^\diamond, T_3^\diamond, ...\}$ is m-invariant if and only if $\{T_1^\diamond, T_i^\diamond\}$ is m-invariant for all $i \geq 1$.*

At a high level, the StatDB $\mathcal{D}$ works as follows. Given the first query Q_1, $\mathcal{D}$ invokes a conventional algorithm (provided in [39]) to obtain an m-unique (see Definition 6) anonymized version $T_1^\diamond$ of T, and answers Q_1 with $T_1^\diamond$. The appearance of $T_1^\diamond$ immediately determines a *candidate pool* $\mathcal{S}^\diamond$, which includes all anonymized versions $T^\diamond$ of T such that $\{T_1^\diamond, T^\diamond\}$ is m-invariant. To process any subsequent query Q_i ($i > 1$), $\mathcal{D}$ computes an anonymized version $T^\diamond$ from $\mathcal{S}^\diamond$ leading to a short result interval for Q_i. According to Lemma 4, all the anonymized versions employed in history must form an m-invariant sequence. It remains to solve two problems:

1. How to store the candidate pool $\mathcal{S}^\diamond$ with the minimum space?
2. How to efficiently retrieve the best anonymized version in $\mathcal{S}^\diamond$ to answer a query Q_i?

In Section 5, we will address the first question with *bucketization*. Sections 6 and 7 provide two solutions to the second problem, by adopting anatomy and generalization, respectively.

5. THE BUCKETIZATION TECHNIQUE

Given T and $T_1^\diamond$, $\mathcal{S}^\diamond$ typically contains an enormous number of anonymized versions, rendering the materialization of $\mathcal{S}^\diamond$ impractical. To overcome this problem, we propose a *bucketization* technique. This technique does not capture $\mathcal{S}^\diamond$ directly; instead, it allows fast derivation of any anonymized version in $\mathcal{S}^\diamond$ from a space-efficient structure.

DEFINITION 11 (BUCKETIZATION). Given T and $T_1^\diamond$, a *bucket* is a set of tuples in T whose signatures in $T_1^\diamond$ are identical. The *signature of a bucket* B is the set of sensitive values that appear in B. A *bucketization* U is a set of disjoint buckets, such that (i) no two buckets have the same signature, and (ii) the union of all buckets equals T. □

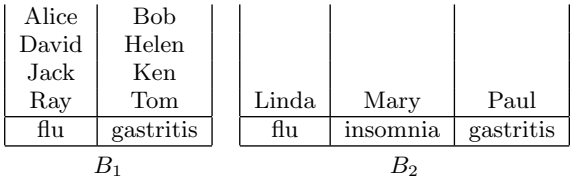


Figure 1: A bucketization of the microdata Table 1a

The bucketization U is determined, once $T_1^\diamond$ is available. In particular, U can be obtained by simply hashing the tuples in T to buckets, using their signatures in $T_1^\diamond$ as the hash values. For example, Figure 1 illustrates the bucketization, assuming that the microdata T is Table 1a, and $T_1^\diamond$ is the generalized Table 1b. The first (second) bucket contains eight (three) tuples, indicated with their owners' names, and has a signature $\{\text{flu}, \text{gastritis}\}$ ($\{\text{flu}, \text{insomnia}, \text{gastritis}\}$). Lemma 5 clarifies an important property of buckets:

LEMMA 5. *Given a bucket $B \in U$ with a signature K, for any sensitive value $v \in K$, there exist $|B|/|K|$ tuples $t \in B$ with $t[A^s] = v$, where $|B|$ is the number of tuples in B, and $|K|$ the number of sensitive values in K (by Definition 11, $|B|/|K|$ is always an integer).*

For example, B_1 in Figure 1 has a signature K with two values flu and gastritis, and each value is possessed by $|B_1|/|K| = 8/2 = 4$ tuples in B_1. A crucial implication of Lemma 5 is that a bucket B can be split into $|B|/|K|$ groups, each containing $|K|$ tuples with distinct sensitive values. This is formally defined as follows.

DEFINITION 12 (DECOMPOSITION). Let B be a bucket with signature K. A *decomposition of B* contains $|B|/|K|$ disjoint QI-groups whose union is B, and all of them have signature K. Given a bucketization U, a *decomposition of U* is the set of QI-groups obtained by decomposing each bucket in U. □

When $|B|/|K| \geq 2$, B can be decomposed in multiple ways. For example, we can divide the bucket B_1 in Figure 1 into four QI-groups, $\{\text{Alice}, \text{Bob}\}$, $\{\text{David}, \text{Helen}\}$, $\{\text{Jack}, \text{Ken}\}$, and $\{\text{Ray}, \text{Tom}\}$, all of which have the same signature $\{\text{flu, gastritis}\}$ as B_1. These QI-groups constitute part of the underlying partition (P_1 in Section 3.1) of Table 1b. Alternatively, B_1 can be split into another four QI-groups, $\{\text{David}, \text{Ken}\}$, $\{\text{Alice}, \text{Helen}\}$, $\{\text{Jack}, \text{Tom}\}$, and $\{\text{Ray}, \text{Bob}\}$, which also have the signature $\{\text{flu, gastritis}\}$, and are in the partition (P_2) of Table 2.

Each decomposition of a bucketization U is essentially a partition of the microdata T. Interestingly, there exists a bijection between the decompositions of U and the anonymized tables in $\mathcal{S}^\diamond$:

LEMMA 6. *Let U be the bucketization of T decided by $T_1^\diamond$. Any decomposition of U is a partition of T underlying an anonymized version in $\mathcal{S}^\diamond$. Conversely, for any anonymized version $T^\diamond \in \mathcal{S}^\diamond$, the partition of T that defines $T^\diamond$ is a decomposition of U.*

Recall that, for each query Q_i to the statistical database $\mathcal{D}$, we aim at discovering the anonymized version $T_i^\diamond \in \mathcal{S}^\diamond$ that minimizes the result interval for Q_i. By Lemma 6, it is equivalent to identify the decomposition P_i of U that generates $T_i^\diamond$. In Section 6 (Section 7), we will develop efficient algorithms for finding P_i, when anatomy (generalization) is the adopted anonymization framework. It is worth mentioning that U is the only information stored in our StatDB. In particular, no information about P_i needs to be remembered. Since each tuple is stored in U exactly once, the space consumption is the same as the microdata.

6. ANATOMY-BASED STAT. DB.

This section studies the following problem. We have a privacy preserving StatDB $\mathcal{D}$ that anonymizes the microdata T with anatomy. Given a query Q and a bucketization U of T, find the optimal decomposition P of U (P is also a partition of T, as shown in Lemma 6) such that P produces, by Lemma 3, the shortest result interval of Q. In Section 6.1, we first explain the characteristics of P. Then, Section 6.2 presents algorithms for computing P.

6.1 Optimal Result Intervals

We will need the notion of "a-interval" frequently:

DEFINITION 13 (A-INTERVAL). Given a query Q and a decomposition L of a bucket $B \in U$, the *a-interval* of L is $[a_\vdash(L), a_\dashv(L)]$, such that

$$a_\vdash(L) = \sum_{G \in L} h_\vdash(G), \text{ and } \; a_\dashv(L) = \sum_{G \in L} h_\dashv(G),$$

where functions $h_\vdash(.)$ and $h_\dashv(.)$ are defined in Lemma 3. □

Intuitively, the a-interval of L is the contribution of B to the overall result interval of Q. To understand this, assume that U has $|U|$ buckets B_1, B_2, ..., $B_{|U|}$, and P is the decomposition of U used to answer Q. As P is the union of the decompositions L_1, ..., $L_{|U|}$ of all buckets B_1, B_2, ..., $B_{|U|}$, each QI-group in P appears in exactly one L_i, for some $i \in [1, |U|]$. Therefore, the result interval $[r_\vdash, r_\dashv]$ of Q from Lemma 3 can be re-written as:

$$r_\vdash = \sum_{i=1}^{|U|} \left(\sum_{G \in L_i} h_\vdash(G) \right) = \sum_{i=1}^{|U|} a_\vdash(L_i) \quad (3)$$

$$r_\dashv = \sum_{i=1}^{|U|} \left(\sum_{G \in L_i} h_\dashv(G) \right) = \sum_{i=1}^{|U|} a_\dashv(L_i) \quad (4)$$

Hence, the length $r_\dashv - r_\vdash$ of the interval equals $\sum_{i=1}^{|U|}(a_\dashv(L_i) - a_\vdash(L_i))$, i.e., the total length of the a-intervals of all L_i.

We say that a decomposition L of a bucket $B \in U$ is *a-optimal* for Q, if L achieves the shortest a-interval among all possible decompositions of B. Therefore, a decomposition P of U minimizes the result interval of Q, if and only if the decomposition of each bucket $B \in U$ is a-optimal. Note that the optimality is on condition of U, i.e., the optimal result may vary for different U.

6.2 Algorithms

In the sequel, we first provide a solution that finds the optimal result interval, by obtaining the a-optimal decomposition of each bucket in U. The solution motivates a faster method, which derives the optimal result without extracting any decomposition of U.

Algorithm A-Decompose (B, Q)
1. K = the signature of B; $L = \emptyset$
2. S_1 = the set of tuples in B that satisfy all $p(A_1^q)$, ..., $p(A_d^q)$ in Q
3. $S_2 = B - S_1$
4. for $i = 1$ to $|B|/|K|$
5. $G = \emptyset$
6. for $j = 1$ to $|K|$
7. v = the j-th sensitive value in K
8. if there exists a tuple $t_1 \in S_1$ with $t_1[A^s] = v$
9. remove t_1 from S_1 and insert it into G
10. otherwise, find a tuple $t_2 \in S_2$ with $t_2[A^s] = v$, remove t_2 from S_2 and insert it into G
11. insert G into L
12. return L

Figure 2: The *A-Decompose* algorithm

A Materialization Approach. Figure 2 presents the *A-Decompose* algorithm that, given a query Q and a bucket $B \in U$, returns an a-optimal decomposition L of B. We illustrate the algorithm, by setting Q to the query Q_2 in Section 1.3, and B to the bucket B_1 in Figure 1 (B_1 concerns the microdata in Table 1a).

EXAMPLE 1. *A-Decompose* begins by obtaining the signature K = {flu, gastritis} of B_1, and initializing an empty set L of QI-groups. Then, it creates (i) a set S_1 that contains the tuples in B_1 satisfying all the QI predicates in Q_2, and (ii) a set S_2 that includes the other tuples in B_1. Here, S_1 = {Helen, Jack, Tom}, and S_2 = {Alice, Bob, David, Ken, Ray}.

Next, *A-Decompose* populates L with $|B_1|/|K| = 8 / 2 = 4$ QI-groups. Each group contains $|K| = 2$ tuples, each carrying a different sensitive value in K. Whenever possible, these tuples are selected from S_1. If S_1 runs out of tuples having a required sensitive value, a tuple from S_2 is chosen instead.

Continuing our example, let us assume that the algorithm picks {Jack, Tom} from S_1 to spawn the first QI-group G_1. Then, these tuples are removed from S_1, which becomes {Helen}. In creating the second group G_2, however, *A-Decompose* sees that S_1 contains no tuple with flu. Hence, it selects a tuple, e.g., Alice, from S_2 with that disease, and adds Alice to G_2. G_2 still needs another tuple with gastritis. Since S_1 has such a tuple, i.e., Helen, it is chosen, completing G_2 as {Alice, Helen}. After this, Alice and Helen are removed from S_1 and S_2, respectively.

Now S_1 becomes empty. Therefore, the last two QI-groups G_3 and G_4 are formed entirely using the tuples in S_2. A possible result, for instance, is G_3 = {David, Ken} and G_4 = {Ray, Bob}. □

Next, we prove that *A-Decompose* always produces an a-optimal decomposition of any bucket B. Let S_1 be the set of tuples in B satisfying all the QI predicates of Q, and α the number of sensitive values in K satisfying the sensitive predicate. For the i-th value $v_i \in K$, use β_i to denote the number of tuples in S_1 with sensitive value v_i. Assuming, without loss of generality, $\beta_1 \leq \beta_2 \leq ... \leq \beta_{|K|}$, we have:

Algorithm A-Count (B, Q)
1. K = the signature of B
2. α = the number of values in K satisfying $p(A^s)$ in Q
3. create an integer array β with $|K|$ elements
4. for $i = 1$ to $|K|$
5. β_i = the number of tuples $t \in B$, such that t satisfies all the QI predicates in Q, and $t[A^s]$ equals the i-th value in K
6. sort the values in array β in ascending order
7. return $[\sum_{i=1}^{\alpha} \beta_i, \sum_{i=|K|-\alpha+1}^{|K|} \beta_i]$

Figure 3: The *A-Count* algorithm

LEMMA 7. *Given a query Q and a bucket B,* A-Decompose *returns an a-optimal decomposition L of B for Q, whose a-interval $[a_{\vdash}(L), a_{\dashv}(L)]$ is given by*

$$a_{\vdash}(L) = \sum_{i=1}^{\alpha} \beta_i, \text{ and } a_{\dashv}(L) = \sum_{i=|K|-\alpha+1}^{|K|} \beta_i.$$

Equipped with *A-Decompose*, our StatDB $\mathcal{D}$ calculates the optimal result interval $[r_{\vdash}, r_{\dashv}]$ of Q as follows. For each bucket B_i ($1 \leq i \leq |U|$) in U, $\mathcal{D}$ obtains an a-interval $[a_{\vdash}(L_i), a_{\dashv}(L_i)]$, where L_i is the a-optimal decomposition of B_i returned by *A-Decompose*. Then, the values of $r_{\vdash}$ and $r_{\dashv}$ are derived with Equations 3 and 4, respectively.

A Non-Materialization Approach. The above analysis points to an important observation. In Lemma 7, $a_{\vdash}(L)$ and $a_{\dashv}(L)$ depend on only α and β_i ($1 \leq i \leq |K|$), which can be easily acquired without constructing the QI-groups in L! Motivated by this, we propose an algorithm *A-Count* in Figure 3 that, given a bucket B and a query Q, returns directly the a-interval $[a_{\vdash}(L), a_{\dashv}(L)]$ of the a-optimal decomposition L of B. Again, let us illustrate the algorithm by setting Q to the query Q_2 in Section 1.3 and B to the bucket B_1 in Figure 1.

EXAMPLE 2. *A-Count* starts by obtaining the signature K = {flu, gastritis} of B_1. Then, it obtains the number $\alpha = 1$ of sensitive values in K satisfying the sensitive predicate of Q_2 (only flu in K satisfies). Next, the algorithm creates an integer array β with $|K| = 2$ elements, where β_1 (β_2) equals the number 1 (2) of tuples in B_1 that satisfy the QI-predicates of Q_2, and their *Disease*-values are flu (gastritis). For instance, $\beta_1 = 1$ because Jack is the only tuple in B_1 that has a Zipcode in [20k, 40k], and carries the *Disease*-value flu. Finally, α and array β are employed to compute the a-interval with Lemma 7. □

Given a query Q, our StatDB answers it in the same way as the materialization approach, except that *A-Decompose* is replaced with *A-Count*.

Employing an Inverted Index. *A-Count* examines all buckets in U to answer a query Q, but this can be easily avoided by indexing the buckets' signatures. Observe that, a bucket does not need to be inspected, if no value in its signature satisfies the sensitive predicate $p(A^s)$ of Q. Therefore, we may create an inverted index, which has an entry for every sensitive value v, and the entry stores the addresses of all the buckets whose signatures include v. To process Q, we first collect the set S of all sensitive values satisfying

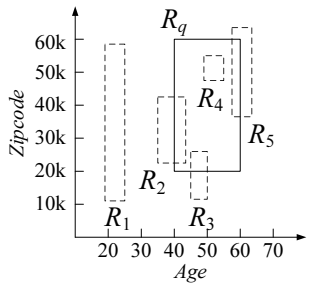


Figure 4: Representation of Q_3 and the QI-groups in Table 2 in the query space defined by Q_3

$p(A^s)$. Then, the buckets containing some elements of S can be efficiently retrieved using the inverted index.

7. GENERALIZATION-BASED STAT. DB.

This section tackles the same problem as Section 6, except that anonymization is performed with generalization. Specifically, given a query Q and a bucketization U of the microdata T, find a decomposition P of U (P is also a partition of T) which produces a generalized version of T that gives, by Lemma 2, the shortest result interval for Q. In Section 7.1, we first clarify a transformation that provides an intuitive interpretation of query answers. Then, Section 7.2 presents a hardness result about finding the optimal answer. Finally, Section 7.3 elaborates a heuristic algorithm.

7.1 A Multidimensional Transformation

Given a query Q, we use λ to denote the number of its QI predicates that have the form "$A_i^q \in [x_i, y_i]$", where x_i (y_i) is not ∞ ($-\infty$). Without loss of generality, assume that those λ predicates are the first λ QI attributes in T, i.e., A_1^q, A_2^q, ..., A_λ^q. Let $\mathbb{Q}$ be a λ-dimensional space, where the i-th ($1 \leq i \leq \lambda$) dimension is A_i^q. We refer to $\mathbb{Q}$ as the *query space* decided by Q. We also define the *query rectangle* of Q, as the rectangle R_q in $\mathbb{Q}$ whose projection on the i-th ($1 \leq i \leq \lambda$) dimension equals $[x_i, y_i]$.

For instance, assume that the microdata is Table 1a, and the system receives a query:

Q_3: `SELECT` `COUNT`(*) `FROM` StatDB
`WHERE` *Age* $\in [40, 60]$ `AND` *Zipcode* $\in [20000, 60000]$
`AND` *Disease* = flu

Its query space involves two dimensions *Age* and *Zipcode*. As illustrated in Figure 4, its query rectangle R_q has a projection $[40, 60]$ ($[20000, 60000]$) on the *Age* (*Zipcode*) dimension. Any point in R_q satisfies the QI predicates p(*Age*) and p(*Zipcode*) of Q_3.

Following the way we formulate R_q, any QI-group G^* in a generalized table T^* can also be represented as a *group rectangle* in $\mathbb{Q}$, whose projection on the i-th ($1 \leq i \leq \lambda$) dimension is $G^*[A_i^q]$. As an example, Figure 4a shows five group rectangles R_1, ..., R_5, where R_i ($1 \leq i \leq 5$) corresponds to the i-th (counting from the top) QI-group in Table 1b.

By modeling queries and QI-groups as rectangles in $\mathbb{Q}$, we obtain a geometric interpretation of the sets $F_\vdash(T^*)$ and $F_\dashv(T^*)$ defined in Lemma 2. Specifically, $F_\dashv(T^*)$ ($F_\vdash(T^*)$) contains the QI-groups whose group rectangles intersect (are entirely covered in) the query rectangle R_q. Obviously, $F_\vdash(T^*)$ is a subset of $F_\dashv(T^*)$. By Lemma 2, the result interval of Q on T^* has a length

$$r_\dashv - r_\vdash = \sum_{G^* \in F_\dashv(T^*) - F_\vdash(T^*)} s(G^*). \quad (5)$$

The length is determined by the set $F_\dashv(T^*) - F_\vdash(T^*)$, which essentially includes the QI-groups in T^* whose group rectangles *partially intersect* R_q.

7.2 Hardness of Computing Optimal Results

Given a QI-group G of the microdata T, there are multiple ways to decide the generalized values in G. We consider *MBR-generalization* [19], namely, $G[A_i^q]$ ($1 \leq i \leq \lambda$) is the tightest interval enclosing the $t[A_i^q]$ of all tuples $t \in G$. Alternatively, if each t is viewed as a point in the λ-dimensional space $\mathbb{Q}$ (through the transformation in Section 7.1), then the group rectangle of G is the minimum bounding rectangle (MBR) of all the points converted from the tuples in G. Hence, we will use the terms "group rectangle of G" and "MBR of G" interchangeably.

Recall that the analysis of anatomy relies on the notion of "a-interval". Likewise, the study of generalization needs a corresponding concept "g-interval":

Definition 14 (G-Interval). For a query Q and a decomposition L of a bucket $B \in U$, the *g-interval* of L equals $[g_\vdash(L), g_\dashv(L)]$ such that

$$g_\vdash(L) = \sum_{G \in F_\vdash(L)} s(G), \text{ and } g_\dashv(L) = \sum_{G \in F_\dashv(L)} s(G),$$

where $s(G)$ is the number of tuples in G satisfying the $p(A^s)$ of Q, and $F_\dashv(L)$ ($F_\vdash(L)$) is the set of QI-groups in L whose MBRs intersect (are contained in) the query rectangle of Q. □

As with an a-interval, the g-interval of L is the contribution of B to the overall result interval of Q. Formally, assume that U has buckets B_1, ..., $B_{|U|}$, and P is the decomposition of U deployed to answer Q. The $[r_\vdash, r_\dashv]$ in Lemma 2 can be calculated as:

$$r_\vdash = \sum_{i=1}^{|U|} \left(\sum_{G \in F_\vdash(L_i)} s(G) \right) = \sum_{i=1}^{|U|} g_\vdash(L_i) \quad (6)$$

$$r_\dashv = \sum_{i=1}^{|U|} \left(\sum_{G \in F_\dashv(L_i)} s(G) \right) = \sum_{i=1}^{|U|} g_\dashv(L_i) \quad (7)$$

Hence, the length of the result interval equals $r_\dashv - r_\vdash = \sum_{i=1}^{|U|} (g_\dashv(L_i) - g_\vdash(L_i))$. Let us say that a decomposition L of B is *g-optimal*, if among all the possible decompositions of B, L has the shortest g-interval. Thus, a decomposition P of U achieves the shortest g-interval of Q, if and only if the decomposition of each bucket $B \in U$ is g-optimal.

Now, we focus on a single bucket $B \in U$, and analyze its g-optimal decomposition. Let K be the signature of B; any decomposition L of B must contain only QI-groups whose signatures are K. Therefore, all QI-groups $G \in L$ have the same $s(G)$ (function $s(.)$ is formulated in Definition 14), which is the number α of sensitive values in K satisfying the sensitive predicate $p(A^s)$ of Q. It follows that:

$$g_\vdash(L) = |F_\vdash(L)| \cdot \alpha \quad (8)$$

$$g_\dashv(L) = |F_\dashv(L)| \cdot \alpha. \quad (9)$$

Hence, the length of the g-interval of L equals $g_{\dashv}(L) - g_{\vdash}(L) = |F_{\dashv}(L) - F_{\vdash}(L)| \cdot \alpha$. Therefore, *$L$ is g-optimal for Q, if and only if it minimizes the cardinality of $F_{\dashv}(L) - F_{\vdash}(L)$, i.e., the number of QI-groups in L whose MBRs partially intersect the query rectangle of Q.* Unfortunately, when $|K| > 2$, this is NP-hard:

LEMMA 8. *Given a query Q, and a bucket B with a signature K, identifying the g-optimal decomposition of B is NP-hard when $|K| > 2$.*

The above result is obtained through a reduction from the *k-dimensional matching* problem, which is NP-hard when $k > 2$ [14]. In the next section, we resort to heuristic approaches for choosing a good decomposition of B.

7.3 Algorithm

In Section 6.2, for anatomy, we developed two approaches, which differ in whether the decomposition of a bucket is materialized. Although the materialization approach was used to motivate the non-materialized version, the latter is always faster. Hence, for generalization, we discuss only the non-materialization method.

Objective. The problem addressed in this subsection is as follows. We have a query Q, and a bucket B with signature K. The goal is to produce a short interval, which equals the g-interval $[g_{\vdash}(L_{\triangle}), g_{\dashv}(L_{\triangle})]$ of a decomposition $L_{\triangle}$ of B, without actually obtaining the concrete QI-groups in $L_{\triangle}$. We give $L_{\triangle}$ the name *phantom decomposition*.

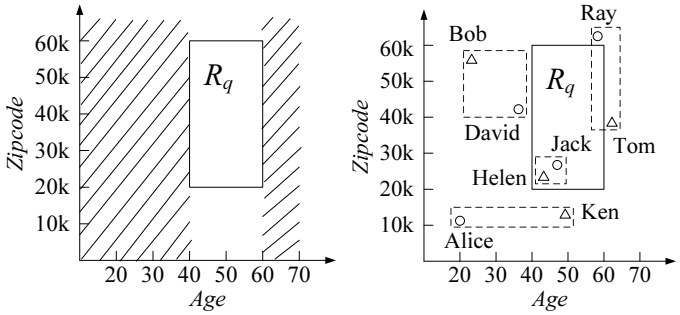
By the analysis of Section 7.2, $L_{\triangle}$ should lead to a small $|F_{\dashv}(L_{\triangle}) - F_{\vdash}(L_{\triangle})|$. Let us divide the QI-groups in $L_{\triangle}$ into three disjoint subsets:

1. $F_{\vdash}(L_{\triangle})$: the set of QI-groups whose MBRs are fully contained in the query rectangle R_q of Q;
2. $F_{\dashv}(L_{\triangle}) - F_{\vdash}(L_{\triangle})$: the set of QI-groups whose MBRs partially intersect R_q;
3. $L_{\triangle} - F_{\dashv}(L_{\triangle})$: the set of QI-groups whose MBRs are disjoint with R_q.

We refer to a QI-group in the i-th ($1 \leq i \leq 3$) subset as a *type-i* QI-group. Since the number of all QI-groups equals the the cardinality $|B|/|K|$ of $L_{\triangle}$, minimization of $|F_{\dashv}(L_{\triangle}) - F_{\vdash}(L_{\triangle})|$ is equivalent to maximizing the numbers of type-1 and -3 groups.

Yield and Margin. We first introduce two concepts imperative to our solution. Given a subset S of B, we define the *yield* of S, as the largest number of QI-groups with signature K (and sharing no tuple) that can be created from S. The yield can be computed easily: it equals $\min\{\beta_1, \beta_2, ..., \beta_{|K|}\}$, where β_i ($1 \leq i \leq |K|$) is the number of tuples in S that carry the i-th sensitive value in K. For example, assume that B is the bucket B_1 in Figure 1, and $S = \{$Ray, Tom, Ken$\}$. The signature K of B is {flu, gastritis}. There is $\beta_1 = 1$ tuple in S taking the sensitive value flu, whereas $\beta_2 = 2$ tuples have gastritis. Hence, the yield of S equals 1.

The second crucial concept is *margin*, which is a special half-plane in $\mathbb{Q}$. Recall that (as mentioned in Section 7.1) query Q has a predicate $A_i^q \in [x_i, y_i]$ for every $i \in [1, \lambda]$, where λ is the dimensionality of the query space $\mathbb{Q}$ defined by Q. The *lower* (*upper*) *margin* along the i-th dimension of $\mathbb{Q}$ is the half-plane satisfying the condition $A_i^q < x_i$ ($A_i^q > y_i$). In this way, altogether 2λ margins are defined. As an example, the left (right) shaded area in Figure 5a demonstrates the lower (upper) margin of Q_3 (in Section 7.1) along the *Age* dimension.



(a) Margins of Q_3 on *Age*

(b) A decomposition of bucket B_1 for query Q_3

Figure 5: Illustration of algorithm *G-Count*

G-Count. Figure 6 formally presents *G-Count* that, given a query Q and a bucket B, returns the g-interval $[g_{\vdash}(L_{\triangle}), g_{\dashv}(L_{\triangle})]$ of a phantom decomposition $L_{\triangle}$ of B. We explain the algorithm by setting Q to the Q_3 in Section 7.1 and B to the B_1 in Figure 1. Let R_q be the query rectangle of Q_3 in its query space $\mathbb{Q}$. Figure 5b demonstrates R_q and the points converted from the tuples in B_1. A dot (triangle) represents a tuple with *Disease*-value flu (gastritis).

EXAMPLE 3. *G-Count* sets off by acquiring the dimensionality $\lambda = 2$ of $\mathbb{Q}$, the signature $K = \{$flu, gastritis$\}$ of B_1, and the number $\alpha = 1$ of values in K satisfying the sensitive predicate of Q_3.

The algorithm proceeds to obtain the largest number n_1 of type-1 QI-groups that can be inserted in $L_{\triangle}$. A QI-group is type-1, if and only if all (the points converted from the) tuples in the group are covered by R_q. Hence, *G-Count* collects the set S_c of tuples in B_1 that are contained in R_q, i.e., $S_c = \{$Helen, Jack$\}$. The value of n_1 equals the yield 1 of S_c, implying that one type-1 QI-group (including Helen and Jack) can be extracted from S_c.

Now, for every sensitive value $v \in K$, *G-Count* removes n_1 tuples from B carrying v (i.e., totally $n_1 \cdot |K|$ tuples are discarded). In particular, all these tuples are chosen from S_c. The removal reflects the fact that, once a tuple is added to a type-1 QI-group in $L_{\triangle}$, it cannot appear in any other QI-group. Continuing our example, Helen and Jack are eliminated from B, which becomes {Alice, Bob, David, Ken, Ray, Tom}.

Then, *G-Count* attempts to maximize the number n_3 of type-3 QI-groups that may be included in $L_{\triangle}$. A QI-group is type-3, if and only if all tuples in the group fall in the same margin of R_q. Following this observation, *G-Count* initializes n_3 to 0, and inspects each of the 2λ margins in turn, trying to increase n_3 after every inspection. Assume that it first examines the lower margin of R_q on the *Age* dimension, and collects the set S_m of tuples {Bob, David, Alice} there. Since the yield y of S_m is 1, the algorithm increases n_3 by 1, indicating that one type-3 QI-group (containing Bob and David) can be spawned from S_m. Furthermore, for each sensitive value $v \in K$, y tuples with value v are deleted from B. Suppose that Bob and David are eliminated from B, which thus changes to {Alice, Ken, Ray, Tom}.

Algorithm G-Count (B, Q)
1. λ = dimensionality of the query space $\mathbb{Q}$ defined by Q
2. K = the signature of B
3. α = the number of values in K satisfying the $p(A^s)$ of Q

   /* Lines 4-7 obtain the number of type-1 QI-groups in the phantom decomposition $L_\triangle$ of B */
4. S_c = the set of tuples in B that satisfy all the QI predicates in Q
5. n_1 = the yield of S_c
6. for $j = 1$ to $|K|$
7. remove n_1 tuples from B that have the j-th sensitive value in K

   /* Lines 8-14 obtain the number of type-3 QI-groups in $L_\triangle$ */
8. $n_3 = 0$
9. for $i = 1$ to λ
10. S_m = tuples of B in the lower margin on the i-th dimension of $\mathbb{Q}$
11. y = the yield of S_m; $n_3 = n_3 + y$
12. for $j = 1$ to $|K|$
13. remove y tuples from B that have the j-th sensitive value in K
14. Repeat Lines 10-14 by using the upper margin on the i-th dimension
15. return $[n_1 \cdot \alpha, (|B|/|K| - n_3) \cdot \alpha]$

Figure 6: The *G-Count* algorithm

Next, *G-Count* performs the same steps with respect to the upper margin on the same dimension. This time, S_m contains a single tuple {Tom}, since the other 3 tuples in B do not fall in that margin. The yield of S_m equals 0; hence, n_3 remains the same. The algorithm now turns to the lower margin of *Zipcode*, and retrieves S_m = {Alice, Ken}, whose yield is 1. Accordingly, n_3 is again increased by 1 (it equals 2 currently), after which both Alice and Ken are evicted from B. leaving only two elements in B = {Ray, Tom}. Finally, examination of the upper margin of *Zipcode* does not alter n_3 and B.

Finally, *G-Count* derives the g-interval $[g_\vdash(L_\triangle), g_\dashv(L_\triangle)]$, by Equations 8 and 9. As set $F_\vdash(L_\triangle)$ includes only type-1 QI-groups, $|F_\vdash(L_\triangle)| = n_1 = 1$. On the other hand, $F_\dashv(L_\triangle)$ involves all the type-1 and -2 QI-groups; thus, $|F_\dashv(L_\triangle)| = |L_\triangle| - n_3 = |B|/|K| - n_3 = 8/2 - 2 = 2$. Therefore, the interval returned is $[|F_\vdash(L_\triangle)| \cdot \alpha, |F_\dashv(L_\triangle)| \cdot \alpha] = [1 \cdot 1, 2 \cdot 1] = [1, 2]$. □

We close this section by elaborating how the StatDB $\mathcal{D}$ computes the result interval $[r_\vdash, r_\dashv]$ of a query Q using a bucketization U. For the i-th ($1 \leq i \leq |U|$) bucket B_i in U, $\mathcal{D}$ applies *G-Count* to produce an interval $[g_\vdash(L_{\triangle i}), g_\dashv(L_{\triangle i})]$. Then, $r_\vdash$ ($r_\dashv$) is calculated with Equation 6 (Equation 7), replacing $g_\vdash(L_i)$ ($g_\dashv(L_i)$) with $g_\vdash(L_{\triangle i})$ ($g_\dashv(L_{\triangle i})$). Apparently, *G-Count* can also be accelerated by deploying an inverted index, created in the same manner as explained in Section 6.2.

8. PERTURBATION-BASED STAT. DB.

Recall that our objective is to build a StatDB that (i) can answer any number of counting queries, and (ii) effectively shield privacy from linking attacks. Although statistical databases have received considerable research attention [1], most of the existing methods are developed in applications where the objective is *not* the prevention of linking attacks. In particular, although some methods (e.g., [25, 30]) have interesting privacy guarantees, those guarantees are specific to their own settings, and do not apply to linking attacks. *Random perturbation* [4, 11], however, is an exception.

Random perturbation falls in the "data modification" category reviewed in Section 1.1. Namely, it transforms a microdata table T to an anonymized version $T^\diamond$. The transformation takes a parameter p, called *retention probability*. Initially, $T^\diamond$ is empty. For every tuple $t \in T$, a coin with head probability p is tossed. If the coin heads, t is added to $T^\diamond$ directly. Otherwise, $t[A^s]$ is replaced with a random value generated in the domain of A^s; the modified t is then inserted into $T^\diamond$. Notice that, regardless of the tossing result, the QI values of t remain unchanged.

Given a counting query formalized in Section 4, $T^\diamond$ can be used to provide an estimated result, whose derivation is nicely explained in [4]. Agrawal et al. [4] prove several privacy guarantees offered by random perturbation. Their analysis is not directly applicable to linking attacks, due to the difference in the modeling of an adversary's background knowledge. In the next lemma, we present the guarantee of perturbation in our settings.

LEMMA 9. *Given a perturbed version $T^\diamond$ of T produced with retention probability p, a QI-conscious adversary has at most $p + (1-p)/|A^s|$ confidence in inferring the sensitive value of any individual in T, where $|A^s|$ is the domain size of A^s.*

The lemma allows us to obtain the maximum p ensuring the same privacy protection as m-invariance. Specifically, by equating $p + (1-p)/|A^s|$ to $1/m$ (see Corollary 1), we solve the maximum as

$$(|A^s| - m)/(m \cdot (|A^s| - 1)). \tag{10}$$

If a lower retention probability is used, the query answer computed from $T^\diamond$ becomes more unreliable, since it has a larger variance, or equivalently, a longer confidence interval. In the experiments, we will show that, offering the same amount of privacy protection, our technique gives shorter result intervals.

It is worth mentioning that, as with anatomy, random perturbation allows precise QI-values to appear in an anonymized version T directly. Therefore, it cannot be used in the applications supported by generalization, where QI-values must be distorted to hide individuals' presence in the microdata [23].

9. EXPERIMENTS

This section experimentally evaluates the effectiveness of the proposed solutions. We deploy a real dataset SAL downloadable at *http://ipums.org*. SAL includes 600k tuples, each of which describes the personal information of an American adult. The dataset has a schema with 8 attributes: { *Age*, *Gender*, *Marital-status*, *Birth-place*, *Education*, *Occupation*, *Race*, *Salary*}, all of which have integer domains, and their domain sizes are 79, 2, 6, 57, 17, 25, 8, and 50, respectively.

From SAL, we generate microdata tables with various cardinalities n and numbers d of QI-attributes. Specifically, a

Parameter	Tested Values
d	3, 4, 5, 6, **7**
ql	2%, 4%, **6%**, 8%, 10%
λ	1, **2**, 3
n	100k, 200k, 300k, 400k, 500k, **600k**

Table 4: Parameters examined and their values

dataset named n-SAL-d (i) takes the first d attributes in the schema of SAL as its QI attributes, and *Salary* as its sensitive attribute, and (ii) contains n tuples randomly sampled from the projection of SAL on the $d+1$ columns selected earlier. For instance, 100k-SAL-3 has QI attributes *Age*, *Gender*, *Marital-status*, and a sensitive attribute *Salary*; it contains 100k tuples of SAL projected on those columns.

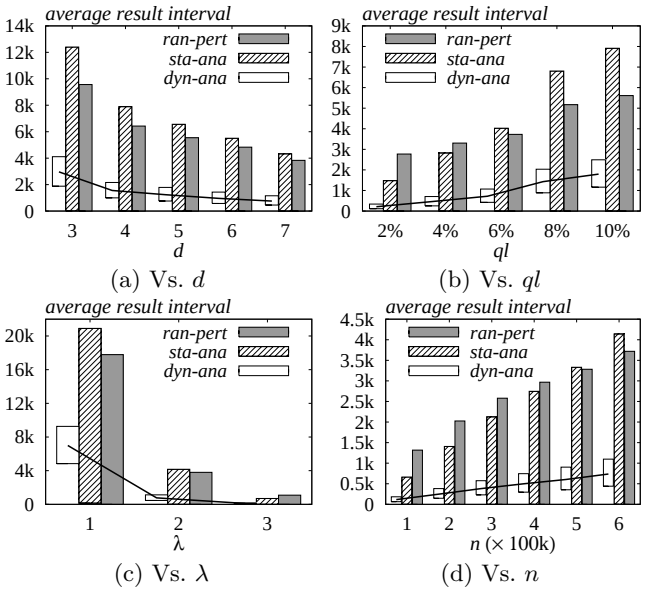
Each query conforms to the generic form in Section 4, and has two parameters: an integer λ and a real value $ql \in (0, 1]$. On λ random QI attributes and the sensitive attribute of the underlying microdata T, the query has a "non-trivial" predicate. Specifically, let A be one of those $\lambda+1$ attributes; the predicate on A has the form $A \in [x, y]$, where the range $[x, y]$ is a random interval in the domain of A, and its length $y - x + 1$ equals $\lceil ql \cdot |A| \rceil$, with $|A|$ being the domain size of A. On the other $d - \lambda$ QI attributes A, the query has a trivial predicate $A \in (-\infty, \infty)$. A *workload* includes 2k queries with the same parameters.

Given a microdata table T, we construct a StatDB using the proposed dynamic anonymization technique, referred to as *dyn-ana* or *dyn-gen*, depending on whether anatomy or generalization is the anonymization framework. Both *dyn-ana* and *dyn-gen* enforce 10-invariance, i.e., an adversary can correctly infer the sensitive value of an individual only with probability 1/10. We also implement the random perturbation approach, denoted as *ran-pert*, discussed in Section 8. The retention probability equals 4/49, as is given by Formula 10 (setting m to 10 and $|A^s|$ to 50), to ensure the same privacy protection as 10-invariance. As an additional competitor, we include *sta-ana* (*sta-gen*), which is a StatDB that uses only a *single* anatomized (generalized) version of T to process all queries, which is obtained using the algorithm in [38] ([19]). Here, the prefix "sta" stands for "static".

We compare alternative methods by their effectiveness in answering counting queries. Given a query, *dyn-ana* (*dyn-gen*) invokes the algorithm *A-Count* (*G-Count*) in Figure 3 (Figure 6) to obtain a result interval. For *sta-gen* (*sta-ana*), the result interval is acquired with Lemma 2 (Lemma 3). For *ran-pert*, on the other hand, we calculate an 80% confidence interval as the result interval, following the derivation in [4]. A method is more accurate, if its result interval is shorter. Note that the performance of *sta-ana* and *sta-gen* essentially represents the accuracy that would be obtained in central publication.

The above five methods are classified into two groups. The first group, called *QI-revealing*, includes *dyn-ana*, *sta-ana*, and *ran-pert*, since they reveal the QI values directly in an anonymized version. The second *QI-concealing* group involves *dyn-gen* and *sta-gen*. Obviously, methods of different groups are incomparable. QI-revealing approaches permit derivation of more accurate results, at the risk of allowing an adversary to infer the presence of an individual (see Section 3.1).

Table 4 summarizes the parameters to be examined, as well as their values tested. Unless otherwise stated, each



(a) Vs. d (b) Vs. ql (c) Vs. λ (d) Vs. n

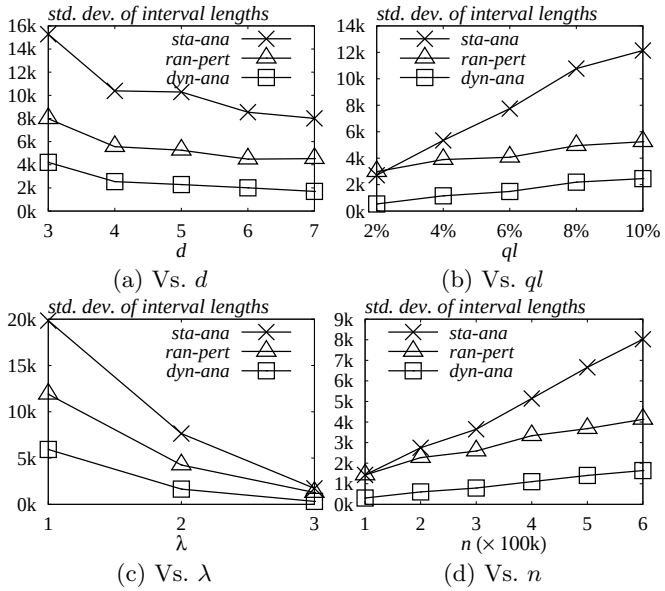
Figure 7: Average result interval comparison (QI-revealing)

parameter is set to its default value, as shown in bold in the table. All the experiments are performed on a machine running a Pentium IV CPU at 3 GHz.

Query Accuracy of QI-Revealing Methods. We first use a workload with default parameters to compare the accuracy of *dyn-ana*, *sta-ana*, and *ran-pert* on datasets 600k-SAL-d, when d varies from 3 to 7. For every d, we measure the *average actual result* (AAR) of all the queries in the workload, the *average result interval* (ARI), and *length standard deviation* (LSD) of each method. Specifically, the ARI is a range $[r_{\vdash}, r_{\dashv}]$, where $r_{\vdash}$ ($r_{\dashv}$) equals the average of the lower (upper) bounds of the result intervals returned by the corresponding method for all the queries in the workload. LSD is the standard deviation of the lengths of those queries' result intervals.

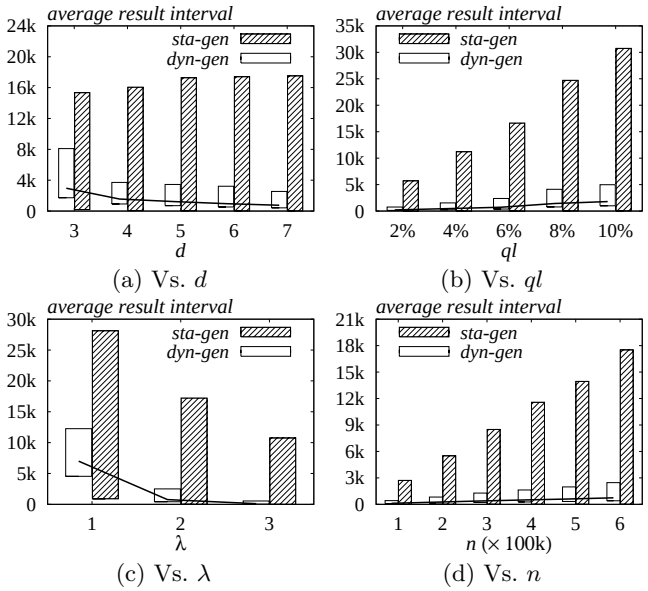
The curve in Figure 7a plots the AAR as a function of d. At each d, there are three columns, indicating the ARIs of the three methods, respectively. Particularly, the bottom (top) of a column captures the lower (upper) bound of the corresponding ARI. The ARIs of *dyn-ana* are significantly shorter than those of the other methods, indicating that *dyn-ana* provides the most accurate query answers. In particular, the lower bound of each ARI of *ran-pert* (*sta-ana*) is useless, since it equals (is close to) zero, which is a trivial lower bound of any result interval. In general, the ARIs of all methods become shorter, when the AAR decreases. For *ran-pert*, this phenomenon is decided by the formula (in [4]) for calculating its 80% confidence intervals. For *dyn-ana* and *sta-ana*, the phenomenon is also expected, because a smaller AAR implies that fewer QI-groups contribute to a query's result, and hence, to the error as well.

Focusing on the dataset 600k-SAL-7, Figure 7b (7c) illustrates the AARs and ARIs as a function of the query parameter ql (λ), as this parameter changes from 2% to 10% (1 to 3). Figure 7d inspects the influence of the dataset cardinality n, as it grows from 100k to 600k. Figure 8 demonstrates the LSDs in the experiments of Figure 7, and smaller deviations imply more stable quality of the query results. The



(a) Vs. d (b) Vs. ql (c) Vs. λ (d) Vs. n

Figure 8: Standard deviation comparison (QI-revealing)



(a) Vs. d (b) Vs. ql (c) Vs. λ (d) Vs. n

Figure 9: Average result interval comparison (QI-concealing)

proposed *dyn-ana* is the clear winner in all cases. In particular, it significantly outperforms *sta-ana*, confirming our motivation in Section 1.3 that a StatDB promises more effective data analysis than central publication.

Query Accuracy of QI-Concealing Methods. Having demonstrated the superiority of *dyn-ana* among the QI-revealing solutions, we proceed to evaluate the QI-concealing approaches. For this purpose, we repeat the same set of experiments in Figures 7 and 8, with respect to *dyn-gen* and *sta-gen*. The results are presented in Figures 9 and 10. As expected, the proposed *dyn-gen* outperforms its competitor considerably in query accuracy. Furthermore, the behavior of *dyn-gen* (*sta-gen*) is analogous to that of *dyn-ana* (*sta-ana*). The only exception is that, in Figure 9a, the ARI of *sta-gen* grows longer as d increases, whereas the ARI of *sta-ana* shrinks in this scenario (Figure 7a). This is because the quality of generalization degrades severely as the number d of QI attributes becomes larger [2], while, as shown in [38], anatomy does not have such degradation.

10. CONCLUSIONS AND FUTURE WORK

This paper studies the construction of a privacy preserving statistical database (StatDB) using anatomy and generalization. We develop a new technique called *dynamic anonymization* that, given a query, efficiently computes a tailor-made anonymized version of the microdata. In this way, better accuracy of the query result is achieved, compared to conventional approaches that use a single version to answer all queries. Privacy protection is guaranteed by ensuring that all the anonymized versions employed in history satisfy the m-invariance principle. As verified by extensive experiments, our StatDB provides highly accurate results to counting queries, and effectively protects data from privacy attacks, even if an adversary has acquired the results of all the past queries.

This work also motivates several directions for future research. First, while in this paper we focus on counting queries, it is interesting to investigate the extension of the proposed solutions to other aggregation operators such as MIN, MAX, and SUM, when the sensitive attribute has a numeric domain. Second, our discussion has concentrated on static microdata, whereas it remains unclear how to tackle data updates (i.e., existing/new tuples are removed/inserted from/into the underlying dataset). Finally, it is a challenging yet exciting topic to explore other forms of statistical databases in non-relational environments (e.g., spatial databases), to address the needs of data analysis and privacy protection there.

Acknowledgements

This work was supported by grants CUHK 1202/06 and 4161/07 from the research grant council of HKSAR. The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

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

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

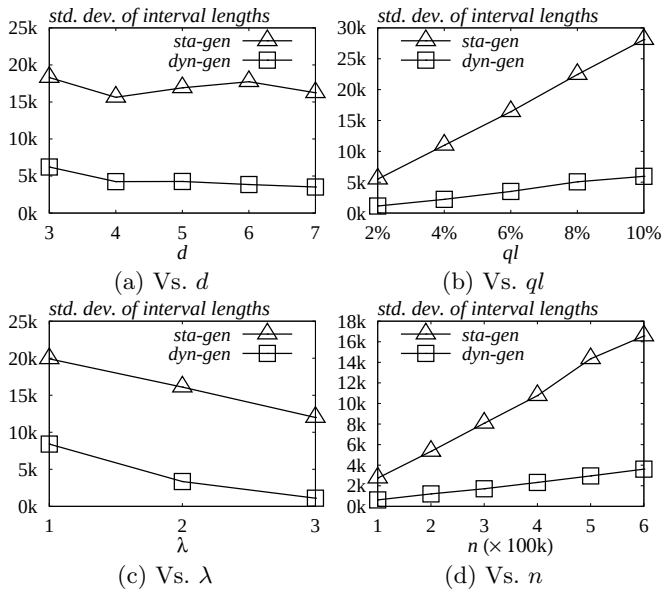
[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *ICDT*, pages 246–258, 2005.

[4] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *SIGMOD*, pages 251–262, 2005.

[5] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *TODS*, 23(3):231–285, 1998.

[6] E. Bertino and E. Ferrari. Secure and selective dissemination of xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.

[7] A. Blum, C. Dwork, F. McSherry, and K. Nissim.



(a) Vs. d (b) Vs. ql

(c) Vs. λ (d) Vs. n

Figure 10: Standard deviation comparison (QI-concealing)

Practical privacy: the sulq framework. In *PODS*, pages 128–138, 2005.

[8] B.-C. Chen, R. Ramakrishnan, and K. LeFevre. Privacy skyline: Privacy with multidimensional adversarial knowledge. In *VLDB*, pages 770–781, 2007.

[9] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

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

[12] E. Ferrari and B. M. Thuraisingham. Security and privacy for web databases and services. In *EDBT*, pages 17–28, 2004.

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

[14] E. Hazan, S. Safra, and O. Schwartz. On the hardness of approximating k-dimensional matching. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(20), 2003.

[15] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *SIGMOD*, pages 37–48, 2005.

[16] T. Iwuchukwu and J. F. Naughton. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *VLDB*, pages 746–757, 2007.

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

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

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

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

[21] N. Li and T. Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.

[22] A. Machanavajjhala, J. Gehrke, and D. Kifer. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[23] D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worst-case background knowledge for privacy preserving data publishing. In *ICDE*, 2007.

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

[25] K. Muralidhar and R. Sarathy. Security of random data perturbation methods. *TODS*, 24(4):487–493, 1999.

[26] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards robustness in query auditing. In *VLDB*, pages 151–162, 2006.

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

[28] H. Park and K. Shim. Approximate algorithms for k-anonymity. In *SIGMOD*, pages 67–78, 2007.

[29] V. Rastogi, S. Hong, and D. Suciu. The boundary between privacy and utility in data publishing. In *VLDB*, pages 531–542, 2007.

[30] S. P. Reiss. Practical data-swapping: the first steps. *TODS*, 9(1):20–37, 1984.

[31] J. Rothe. Some facets of complexity theory and cryptography: A five-lecture tutorial. *ACM Computing Surveys*, 34(4):504–549, 2002.

[32] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, 13(6):1010–1027, 2001.

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

[34] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *SIGKDD*, pages 206–215, 2003.

[35] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *ICDE*, pages 545–546, 2005.

[36] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun. On the correctness criteria of fine-grained access control in relational databases. In *VLDB*, pages 555–566, 2007.

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

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

[39] X. Xiao and Y. Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *SIGMOD*, 2007.

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