INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

# Statistical Database Auditing Without Query Denial Threat

Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, Yingjiu Li

To cite this article:
Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, Yingjiu Li (2014) Statistical Database Auditing Without Query Denial Threat. INFORMS Journal on Computing

Published online in Articles in Advance 22 Sep 2014

. http://dx.doi.org/10.1287/ijoc.2014.0607

*INFORMS Journal on Computing*
*Articles in Advance*, pp. 1–15
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2014.0607


# Statistical Database Auditing Without Query Denial Threat

Haibing Lu
Leavey School of Business, Santa Clara University, Santa Clara, California 95053, hlu@scu.edu

Jaideep Vaidya, Vijayalakshmi Atluri
Rutgers University, Newark, New Jersey 07102
{jsvaidya@rbs.rutgers.edu, atluri@rutgers.edu}

Yingjiu Li
School of Information Systems, Singapore Management University, 178902 Singapore, yjli@scu.edu.sg

Statistical database auditing is the process of checking aggregate queries that are submitted in a continuous manner, to prevent inference disclosure. Compared to other data protection mechanisms, auditing has the features of flexibility and maximum information. Auditing is typically accomplished by examining responses to past queries to determine whether a new query can be answered. It has been recognized that query denials release information and can cause data disclosure. This paper proposes an auditing mechanism that is free of query denial threat and applicable to mixed types of aggregate queries, including sum, max, min, deviation, etc. The core ideas are (i) deriving the complete information leakage from each query denial and (ii) carrying the complete leaked information derived from past answered and denied queries to audit each new query. The information leakage deriving problem can be formulated as a set of parametric optimization programs, and the whole auditing process can be modeled as a series of convex optimization problems.

*Keywords*: statistical database; privacy; auditing; query denial; optimization
*History*: Accepted by Alexander Tuzhilin, (former) Area Editor for Knowledge and Data Management; received October 2012; revised August 2013, February 2014; accepted April 2014. Published online in *Articles in Advance*.

## 1. Introduction

Information technologies have been extensively used to collect and share personal data in areas such as healthcare research, crime analysis, customer relationship management, credit analysis, and demographics. Although it has provided much convenience to our work and daily lives, it has raised strong public concerns about individual privacy. In the healthcare industry, we have recently observed rapid transition toward electronic medical records and data sharing. It has been reported that more than 70 million Americans have some portion of their medical records in electronic format (Kaelber et al. 2008). Healthcare researchers can even access individual Medicare and Medicaid claims data at the website of The Center of Medicare and Medicaid Service, a federal agency. Indeed, many medical identity theft cases have been reported due to electronic medical records, e.g., Agrawal and Budetti (2012). In demographics, it was found that 87% of the U.S. population is uniquely identified by {date of birth, gender, postal code} from the 1990 U.S. Census summary data (Sweeney 2002). Privacy scandals can seriously damage a company's reputation and credibility. In August 2006, AOL released a file containing 20 million search queries for more than 640,000 users, not including the identities of the users, with the intention to provide data for research into online browsing behavior. It was soon found that many users could be easily reidentified by analyzing those seemingly innocuous queries. This caused several lawsuits and legal complaints against AOL.

Various protection mechanisms have been proposes to address the data privacy concern. A conventional approach focuses on designing statistical databases (SDBs) and forming restrictions for accessing confidential data. A SDB (Adam and Wortmann 1989) typically refers to a database used for statistical analysis purposes. An important example is the database maintained by the U.S. Census Bureau. While a SDB contains data at the individual record level, users are typically only allowed to ask queries over aggregates. This is to protect the privacy of data that may be sensitive at the individual record level. For example, if the record-level data include private information, such as salary, product cost, and patient health information, the database users should only be allowed to access innocuous statistics over groups. With knowledge of enough aggregate statistics, sophisticated adversaries can infer confidential data.

Securing SDBs has been the focus of much research since the late 1970s. To control inference from aggregate

statistics, many mechanisms have been proposes. They include auditing queries, e.g., Chin and Özsoyoglu (1982), Chowdhury et al. (1999), query restrictions, e.g., Friedman and Hoffman (1980), Nunez et al. (2007), Dobkin et al. (1979), perturbation, e.g., Matloff (1986), Muralidhar et al. (1999), Lee et al. (2010), Muralidhar et al. (1995), Li and Sarkar (2006), Sarathy et al. (2002), Li and Sarkar (2013), cell suppression, e.g., Castro (2007), Fischetti and Salazar (2001), providing approximate answers, e.g., Kadane et al. (2006), Garfinkel et al. (2002), anonymous data collection, e.g., Kumar et al. (2010), and data shuffling or swapping, e.g., Muralidhar and Sarathy (2006), Li and Sarkar (2011). A good survey of classic inference control techniques on SDBs can be found in Adam and Wortmann (1989). A survey of current advancements on privacy in data publishing, e.g., $k$-anonymity (Samarati and Sweeney 1998), $l$-diversity (Machanavajjhala et al. 2006), $t$-closeness (Li et al. 2007), and differential privacy (Dwork 2008), can be found in Fung et al. (2010).

Clearly, not one proposes protection mechanism is suitable for all SDBs. But among various protection mechanisms, auditing has attracted substantial research interests over the past three decades with its earliest discussion dated back to the 1970s (Chin 1978, Schlorer 1975). Auditing is the continuous monitoring of the user's knowledge that is derived from responses to past queries and used to determine how to respond to a new query. As one of the better protection mechanisms, the features of auditing are well described by Chin and Özsoyoglu (1982) as: (i) Absolute security: By checking the query history, auditing allows us to answer a query only when it is secure to do so. (ii) Maximum information: Given the query history, auditing can provide the maximum information to users, which includes accurate answers and as many query answers to the user as the security permits. (iii) Flexibility: It is flexible to use because protection can be tailored to different sets of queries of users' choice.

Chin and Özsoyoglu (1981) proposes the first formal scheme for auditing. It is to deny a query when the answer combined with past query answers can compromise the database. There exist efficient implementation algorithms. For example, to audit sum-only queries to prevent full disclosure, by representing answered queries and the new query with a matrix and performing Gauss transform, one can quickly determine whether answering the new query would comprise the database. The scheme has been used for decades as a de facto scheme for auditing, which we call *conventional auditing*. Much of the following research focus on various aspects of auditing, like improving algorithm performance (Lu et al. 2009), auditing multidimensional SDBs (Wang et al. 2003, Lu and Li 2008, Li and Lu 2008, Wang et al. 2004), preventing interval-based inference (Li et al. 2003), etc.

Recently, Kenthapadi et al. (2005) discovered a fundamental security flaw in conventional auditing. Query denials release information too. Conventional auditing fails to take the fact into consideration and causes privacy disclosure in some cases. To illustrate it, we borrow the example used in Kenthapadi et al. (2005). A database has three variables $\{x_1, x_2, x_3\}$ of the same value 5 and the auditing goal is to prevent full disclosure. The first query is the sum of the three variables and answered. The second query is the maximum of the three variables and denied because the answer implies all three variables are 5. However, when the query is denied, given the fact that a query is denied only if the answer would cause full disclosure, a sophisticated adversary can figure out that the denied answer must be 5.

Kenthapadi et al. (2005) proposes a new scheme called simulatable auditing. It examines a new query solely based on past query answers without consulting the database. A new query is denied if there exists a database solution, which satisfies all past query answers, and the answer to the new query would comprise that database. Indeed, this scheme effectively prevents the query denial threat, whereas the data utility is significantly hurt. Suppose a database contains all nonnegative elements. Then any sum query cannot be answered, because if all elements are 0s, which is a feasible database solution, then the query answer comprises the database. Being aware of the issue, Kenthapadi et al. (2005) further proposes a relaxed scheme. At each auditing time, it samples a large number of feasible database solutions, which are consistent to the past query answers. If answering a new query does not cause privacy disclosure for the majority of the sampled solutions, then answer the query. The scheme has two limitations: (i) Computationally expense. It is difficult to sample a feasible database solution that satisfies all past query answers, while a large number of feasible database solutions need to be generated at each auditing time. (ii) No guarantee of security. It is because the sampled large number of database solutions may not include the real database. However, their work renewed research interest in auditing.

Malvestuto and Moscarini (2006) proposes another auditing scheme for sum-only queries that we call *modified conventional auditing*. The scheme adds one more step to the conventional auditing. At each auditing time, it firstly computes the bounds of the answer to the new query by inspecting past answered queries. Then it computes the bounds of each database variable by inspecting past answered queries and the derived bounds of the answer to the new query. As the bounds of the answer to the new query are derived from the past answered queries, it does not improve on the estimation of the bounds of database variables. Therefore the modified conventional auditing scheme

would always reach the same decisions as the conventional auditing scheme, and hence is insecure. What the two schemes ignore is that an adversary can obtain additional knowledge from past queries.

In this paper, we study SDBs auditing on various types of queries, including sum, max, min, and deviation. We propose an auditing framework, free of the query denial threat. We strictly comply with the original premise of auditing: continuously monitor the user's knowledge that is derived from responses to past queries and use it to determine how to respond to a new query. But we are aware that responses include both query answers and denials. We will propose the first solution to derive the complete information leakage from a query denial. We are also aware of different natures of various query types, as sum and deviation are of continuous nature, and max and min possess discrete nature. The discrete nature of queries would cause full disclosure of element values when a query denial occurs. To eliminate the query denial threat caused by the inherent discrete nature of max and min queries, we employ the simulatable auditing scheme as an auxiliary step to our general auditing framework.

Our contributions can be summarized as: (i) We present an auditing framework, free of the query denial threat, for various types of queries, including sum, max, min, and deviation. Note that deviation queries have never been studied in the auditing setting. (ii) We provide the first solution to derive the complete information leaked from a query denial. (iii) We design implementation algorithms built on existing parametric optimization results.

## 2. Sum

Sum is the most common aggregate query type and supported in all database systems. We start with sum queries, which are also the focus of the auditing research. We first introduce the problem setting. We denote a SDB with $n$ elements by $\{x_1, \ldots, x_n\}$, and the prior-known bounds on elements by $L \le X \le U$. If none, then $L$ and $U$ are $-\infty$ and $\infty$, respectively. The prior-known bounds are to reflect reality. For instance, a person's salary cannot be negative. We are aware that in some cases adversaries may have more prior information than the bounds of data values. We are also aware that approaches like differential privacy (Dwork 2008) can be used against arbitrary prior knowledge. The consequence of guarding against arbitrary prior knowledge is the significant degradation of the utility of data, which we will discuss explicitly in §6. A user is allowed to continuously submit a sum query of $\sum_{i \in S} x_i$ over any data group $S$. The task of auditing is to prevent an adversary from breaching the database privacy. Full disclosure is commonly studied in the auditing literature. However, one may argue that a variable of 100 is nearly disclosed if an adversary ascertains that the variable is between 99 and 101. In this paper, we adopt the interval-based privacy notion (Li et al. 2003) defined as follows.

Definition 1 (Interval-Based Privacy). A variable $x_i$ is considered safe if one cannot ascertain that $x_i$ resides in an interval with length less than $\delta_i$, the safe threshold for $x_i$.

Suppose a variable $x_i$ with threshold value 5. If one ascertains that $x_i$ is within $[0, 5]$, $x_i$ is considered safe. But if the interval is improved to $[0, 4]$, then $x_i$ is compromised. We are aware that there are other privacy notions, such as $k$-anonymity, $l$-diversity, and distribution-based privacy notions. Because we consider a SDB with all numerical values, this paper uses the interval-based privacy notion. The auditing problem can then be described as the following.

Definition 2 (Auditing Problem). Devise an efficient and effective query response strategy such that all variables are safe regarding the interval-based privacy notion.

### 2.1. Existing Auditing Schemes

Before we present our auditing scheme, we first examine the limitations of existing auditing schemes. To illustrate them, throughout this section, we will use one example as follows. A nonnegative numeric database consists of variables $\{x_1, x_2, x_3, x_4, x_5\}$. They are $\{10, 10, 2, 2, 10\}$ with safe thresholds $\{5, 5, 1, 1, 6\}$, respectively. Queries $\{Q_1, Q_2, Q_3, Q_4\}$ with their accurate answers are listed in the following order:

$$Q_1\colon x_1 + x_2 = 20,$$
$$Q_2\colon x_1 + x_3 = 12,$$
$$Q_3\colon x_2 + x_4 = 12,$$
$$Q_4\colon x_1 + x_5 = 20.$$

*Conventional Auditing.* Conventional auditing is the first proposes auditing scheme in the literature. It can be formally stated as the following.

Definition 3 (Conventional Auditing). Whenever a new query is posed, if the answer to it, when combined with past query answers, can infer that for one variable the difference of its lower and upper bound is *less than or equal to* its safe threshold, deny the query, otherwise answer it.

To implement conventional auditing, the auditor only needs to continuously solve LPs as formulated in (1), where $AX = b$ represents the past answered queries,

$\sum_{i\in Q_m} x_i = b_m$ denotes the new query, and $L \le X \le U$ are public information on $X$.

$$\min(\max)\ \ x_1(x_2, \ldots, x_n) \quad \text{s.t.} \begin{cases} AX = b, \\ \sum\limits_{i\in Q_m} x_i = b_m, \\ L \le X \le U. \end{cases} \tag{1}$$

According to conventional auditing, $Q_1$ is first answered, because returning the answer of 20 only helps refine the bounds for $\{x_1, x_2\}$ to $[0, 20]$; $Q_2$ is answered as well, although the bounds of $x_1$ and $x_2$ are refined to $[0, 12]$ and $[8, 20]$, respectively. However, if $Q_3$ is answered, the bounds for both $x_1$ and $x_2$ would be refined to $[8, 12]$. The interval length is 4, which is less than their safe threshold 5. Therefore $Q_3$ should be denied. The system will proceed to examine $Q_4$. To do so, the system needs to solve the following LPs:

$$\min(\max)\ \ x_1(x_2 \ldots x_5) \quad \text{s.t.} \begin{cases} Q_1\colon\ x_1 + x_2 = 20, \\ Q_2\colon\ x_1 + x_3 = 12, \\ Q_4\colon\ x_1 + x_5 = 20, \\ x_1, \ldots, x_5 \ge 0. \end{cases} \tag{2}$$

The result is $x_1 \in [0, 12]$, $x_2 \in [8, 20]$, $x_3 \in [0, 12]$, $x_4 \ge 0$, and $x_5 \in [8, 20]$. Every variable is considered to be safe regarding their safe thresholds. Therefore, $Q_4$ is answered.

However, this is wrong. If $Q_4$ is answered, $x_5$ will be disclosed, as it can be deduced to fall in $[8, 13]$, and the length is less than its safe threshold 6. The reason is that the denial of $Q_3$ releases some information.

To explain the reason, let us first denote the real answer to $Q_3$ by $A_3$. Given $Q_1\colon x_1 + x_2 = 20$, $Q_2\colon x_1 + x_3 = 12$, and $Q_3\colon x_2 + x_4 = A_3$, it is not difficult to infer that $x_1 \in [20 - A_3, 12]$, $x_2 \in [8, A_3]$, $x_3 \in [0, \min\{12, A_3 - 8\}]$, and $x_4 \in [0, A_3 - 8]$. Denying $Q_3$ implies that if $A_3$ is released, at least for one variable, the difference of its lower and upper bounds becomes less than or equal to its safe threshold. The following are four possibilities:

$$\begin{cases} x_1\colon\ 12 - (20 - A_3) \le 5, \\ x_2\colon\ A_3 - 8 \le 5, \\ x_3\colon\ \min\{12, A_3 - 8\} - 0 \le 1, \\ x_4\colon\ A_3 - 8 \le 1. \end{cases} \tag{3}$$

For cases $x_1$ and $x_2$, $A_3 \in (-\infty, 13]$. For cases $x_3$ and $x_4$, $A_3 \in (-\infty, 9]$. Because the adversary cannot ascertain which variables are to be disclosed, all he can infer is $A_3 \in (-\infty, 13] \cup (-\infty, 9] = (-\infty, 13]$.

Given the answered $Q_2$ of $x_1 + x_3 = 12$, we have $x_1 \le 12$. By combining it with the answered $Q_1$ of $x_1 + x_2 = 20$, we further have $x_2 \ge 8$. Therefore, $A_3 = x_2 + x_4 \ge 8$. Finally, from the denial of $Q_3$ in addition to the past two query answers, one can infer $8 \le A_3 \le 13$.

Given $8 \le A_3 \le 13$ derived from the denial of $Q_3$, the adversary can deduce $x_1 \in [7, 12]$ and $x_2 \in [8, 13]$, which make the privacy of both $x_1$ and $x_2$ at the edge of being breached with safe thresholds of 5. However, they are still considered to be safe according to the data disclosure definition.

The real threat of the query denial of $Q_3$ comes when auditing $Q_4$. By solving LPs (4), where $Q_3\colon 8 \le x_2 + x_4 \le 13$ is the complete information leakage from the denial of $Q_3$, $x_5$ is deduced to fall in $[8, 13]$. Hence the privacy of $x_5$ is breached, because its safe threshold is 6. However, conventional auditing fails to detect the breach:

$$\min(\max)\ \ x_1(x_2 \ldots x_5) \quad \text{s.t.} \begin{cases} Q_1\colon\ x_1 + x_2 = 20, \\ Q_2\colon\ x_1 + x_3 = 12, \\ Q_3\colon\ 8 \le x_2 + x_4 \le 13, \\ Q_4\colon\ x_1 + x_5 = 20, \\ x_1, \ldots, x_5 \ge 0. \end{cases} \tag{4}$$

*Simulatable Auditing.* Simulatable auditing was proposed to prevent the attack of query denials. Its basic idea is essentially to protect data privacy by denying any suspicious query that may cause trouble. Its original definition as stated in Kenthapadi et al. (2005) is given as follows.

Definition 4 (Simulatable Auditing (Kenthapadi et al. 2005)). An auditor is simulatable if the decision to deny or give an answer to the query $q_t$ is made based exclusively on $q_1, \ldots, q_t$ and $a_1, \ldots, a_{t-1}$ (and not $a_t$ and not the data set $X = \{x_1, \ldots, x_n\}$), and possibly also the underlying probability distribution $\mathcal{D}$ from which the data was drawn.

To achieve such a simulatable auditing, Kenthapadi et al. (2005) proposes to do the following. Given previously posed queries $\{q_1, \ldots, q_{m-1}\}$ and their answers $\{a_1, \ldots, a_{m-1}\}$, a newly posed query $q_m$ will be denied if (i) there exists a feasible answer $a'_m$ to the new query $q_m$, which is consistent with all past query answers and (ii) releasing $a'_m$ would breach some variable's privacy. The essential idea of simulatable auditing is to deny more queries, including innocent queries to achieve data security. However, it suffers from the serious data utility issue.

Look at $Q_1\colon x_1 + x_2 = 20$ in the previous example. Obviously $x_1 + x_2 = 0$ is a feasible answer to the query. If the answer is 0, given all variables are nonnegative, both $x_1$ and $x_2$ are 0, and hence uniquely identified. According to simulatable auditing, $Q_1$ should be denied, as well as all subsequent queries.

*Modified Conventional Auditing.* Malvestuto and Moscarini (2006) propose an auditing scheme for sum queries, which attempts to solve the query denial issue. Its basic idea is summarized as the following.

Definition 5 (Modified Conventional Auditing). Whenever a new query is posed, if the answer to it when combined with past query answers does not threaten data privacy, answer the query; otherwise, return an approximate answer, the bounds of the real answer, which are derived from past query answers.

The scheme adds one more step to conventional auditing: If the answer to a new query is determined to be dangerous, derive the bounds of the real answer from past query answers and release such bounds instead of the real answer. However, the query submitter alone can derive such released bounds, as he knows all past query answers. In fact, an intelligent adversary can infer more information. Releasing bounds is the same as telling the query submitter that the real answer would make some variable to be in danger. Thus the query submitter can narrow the denied real answer and would eventually use the narrowed results to threaten data privacy. Therefore, modified conventional auditing still suffers from the attack of query denials as conventional auditing, because it inaccurately calculates the complete information leakage from query denials.

To illustrate, look at the previous example again. $Q_1$ and $Q_2$ are answered, and $Q_3$ obviously should be denied. Instead of denying $Q_3$, modified conventional auditing derives the lower and upper bounds of $Q_3$ based on past query answers and returns such an approximate answer to the query submitter. By solving LPs (5), the constraints of which are answers to $Q_1$ and $Q_2$, $x_2 + x_4$ is limited to $[8, +\infty)$, which is issued to the user and will be carried over to audit subsequent queries.

$$\min(\max)\ \ x_2 + x_4 \quad \text{s.t.} \begin{cases} Q_1\colon x_1 + x_2 = 20, \\ Q_2\colon x_1 + x_3 = 12, \\ x_1, \ldots, x_5 \geq 0. \end{cases} \tag{5}$$

The system proceeds to check $Q_4$ by solving LPs (6), where $Q_3\colon x_2 + x_4 \geq 8$ is the information derived from the previous step.

$$\min(\max)\ \ x_1(x_2 \ldots x_5) \quad \text{s.t.} \begin{cases} Q_1\colon x_1 + x_2 = 20, \\ Q_2\colon x_1 + x_3 = 12, \\ Q_3\colon x_2 + x_4 \geq 8, \\ Q_4\colon x_1 + x_5 = 20, \\ x_1, \ldots, x_5 \geq 0. \end{cases} \tag{6}$$

The results of the above LP suggest that $x_1 \in [0, 12]$, $x_2 \in [8, 20]$, $x_3 \in [0, 12]$, $x_4 \geq 0$, and $x_5 \in [8, 20]$, which is exactly the same as the results of conventional auditing. Hence, according to modified conventional auditing, $Q_4$ is answered. However, as explained before, releasing $Q_4$ would infer $x_5 \in [8, 13]$, and hence breach the privacy of $x_5$. The reason modified conventional auditing fails is because it inaccurately calculates the complete information leakage from a query denial. The denial of $Q_3$, in fact, can narrow down the answer of $Q_3$ to $[8, 13]$ instead of $[8, +\infty)$.

### 2.2. New Auditing Scheme

From the previous example, we observe that there is no privacy threat if every query is inspected by incorporating complete information released from both past query answers and denials. The observation naturally leads to the prototype of our new auditing scheme as follows.

At each auditing time, the lower and upper bounds of every variable are derived by inspecting *complete* information released from both past query answers and denials along with the current query.

If we strictly comply with the above auditing scheme, it is unlikely to find a practical implementation algorithm. It is because a denied answer can be narrowed down to a feasible solution region composed of discrete intervals. Because discrete intervals cannot be formulated as linear equalities or inequalities employed in a standard LP form, it poses great difficulty for inspecting the subsequent queries. We are still able to solve the problem by formulating it as a mixed-integer programming (MIP) problem by introducing slack integer variables, and MIP is generally NP-hard (Garey and Johnson 1979).

To further elaborate, consider the following example of nonnegative variables $\{x_1, x_2, x_3, x_4\}$, all with safe thresholds of 1. Suppose the following two queries are posed, where $\lambda$ denotes the answer of $x_1 + x_2$:

$$\begin{aligned} &Q_1\colon x_1 + x_2 + x_3 + x_4 = 5, \\ &Q_2\colon x_1 + x_2 = \lambda. \end{aligned} \tag{7}$$

$Q_1$ is answered and then $Q_2$ is denied. Because $Q_2$ is denied, an adversary would know that the answer of $Q_2$ can infer an interval of some variable's value, with length equal to or less than the variable's safe threshold. Furthermore, the adversary can infer that $\lambda$ must fall in either $[0, 1]$ or $[4, 5]$. When $\lambda \in [0, 1]$, as $0 \leq x_1, x_2 \leq \lambda$, $x_1$, and $x_2$ suffer from the disclosure threat, and hence $Q_2$ has to be denied. When $\lambda \in [4, 5]$, as $0 \leq x_3, x_4 \leq 5 - \lambda$, $x_3$, and $x_4$ suffer from the disclosure threat, and hence $Q_2$ needs to be denied. Denying $Q_2$ is equivalent to releasing the information of $\lambda \in [0, 1] \cup [4, 5]$. If we need to carry this information over to inspect subsequent queries, when auditing the following query, the formulated optimization problems for deriving bounds of involved variables are no longer convex optimization problems.

To address the issue, we propose a refined version of the previous auditing scheme.

Definition 6 (Auditing Sum Queries). At each auditing time, for each variable, derive its lower and upper bounds by inspecting both exact and approximate query answers in the past along with the new query:

- If the difference of the lower and upper bounds for every variable is greater than its safe threshold, return the exact answer;
- Else, return an approximate answer, which is obtained by deriving the region of possible denied query answers (which could include multiple separate intervals), and *returning the interval in the region that contains the real answer*.

Note that if the region of possible denied answers include multiple nonoverlapping intervals, we return the interval containing the real answer, which corresponds to a lower and upper bound of the real answer.

To illustrate it, we still consider example (7). Suppose $x_1 = 0.4$, $x_2 = 0.4$, $x_3 = 0.2$, $x_4 = 4$, and $Q_2$: $x_1 + x_2 = 0.8$. Even though denying a query only helps infer $\lambda \in [0, 1] \cup [4, 5]$, we release $\lambda \in [0, 1]$. The information we return is more than what a query denial implies (since we in effect reveal which variables' privacy is being threatened). The reason we return such an interval is because the returned information still makes the whole feasible solution space to be convex. Then to audit subsequent queries, we are still able to formulate it as a series of LPs.

### 2.3. Security Analysis

Is the auditing scheme for sum queries secure? This section will answer this question. First, let us examine how information is released from a query denial. From an adversary's perspective, if a query is denied, there must exist some variable such that the difference of its lower and upper bounds is less than or equal to its safe threshold, given the denied query's answer. If $x_i$ causes the query denial, one can deduce a feasible solution region $feasible_i(\lambda)$ of the denied answer $\lambda$, such that $\lambda$ being any value in $feasible_i(\lambda)$ would make the difference of $x_i$'s lower and upper bounds less than or equal to its safe threshold. But the adversary has no knowledge of which variable (variables) causes the query denial. Therefore the complete information leakage from a query denial is $\bigcup_i feasible_i(\lambda)$. The following theorem is used to prove that each $feasible_i(\lambda)$ must be one continuous interval.

Theorem 1. *If given* $\{AX = b, aX = \lambda, L \le X \le U\}$, *the values, which* $\lambda$ *can have so that it is possible to deduce that the difference of the lower and upper bounds of* $x_i$ *is less than or equal to* $\delta_i$, *must be one continuous interval.*

Denote $\mathcal{S}$ to be the value set of $\lambda$ such that $\forall \lambda \in \mathcal{S}$, $\{AX = b, aX = \lambda, L \le X \le U\}$ deduces $x_i$'s upper bound of $\max_\lambda(x_i)$, and $x_i$'s lower bound of $\min_\lambda(x_i)$, such that $\max_\lambda(x_i) - \min_\lambda(x_i) \le \delta_i$. Theorem 1 proves that if $\lambda_1, \lambda_2 \in \mathcal{S}$, and $\lambda_2 > \lambda_1$, any $\lambda_3 \in [\lambda_1, \lambda_2]$ must belong to $\mathcal{S}$, in other words, $\max_{\lambda_3}(x_i) - \min_{\lambda_3}(x_i) \le \delta_i$. Because $\max_{\lambda_3}(x_i) - \min_{\lambda_3}(x_i) \le \delta_i$ is equivalent to that for any pair of solutions $X^1(\lambda_3)$ and $X^2(\lambda_3)$, both of which satisfy constraints of $\{AX = b, aX = \lambda_3, L \le X \le U\}$, $|X_i^1(\lambda_3) - X_i^2(\lambda_3)| \le \delta_i$ holds, where $X_i^1(\lambda_3)$ and $X_i^2(\lambda_3)$ denote the values of $x_i$.

For any $\lambda_3 \in [\lambda_1, \lambda_2]$, we can represent $\lambda_3$ as $\lambda_3 = \varepsilon\lambda_1 + (1 - \varepsilon)\lambda_2$, where $\varepsilon \in [0, 1]$. Any feasible solution $X(\lambda_3)$ satisfying $\{AX = b; aX = \lambda_3; L \le X \le U\}$ can be represented as $X(\lambda_3) = \varepsilon X(\lambda_1) + (1 - \varepsilon)X(\lambda_2)$ as well. This can be seen as follows:

- $A(\varepsilon X(\lambda_1) + (1 - \varepsilon)X(\lambda_2)) = \varepsilon AX(\lambda_1) + (1 - \varepsilon) \cdot AX(\lambda_2) = \varepsilon b + (1 - \varepsilon)b = b$;
- $a(\varepsilon X(\lambda_1) + (1 - \varepsilon)X(\lambda_2)) = \varepsilon aX(\lambda_1) + (1 - \varepsilon)aX(\lambda_2) = \varepsilon\lambda_1 + (1 - \varepsilon)\lambda_2 = \lambda_3$;
- $X(\lambda_3) = \varepsilon X(\lambda_1) + (1 - \varepsilon)X(\lambda_2) \ge (\varepsilon + (1 - \varepsilon))L \ge L$;
- $X(\lambda_3) = \varepsilon X(\lambda_1) + (1 - \varepsilon)X(\lambda_2) \le (\varepsilon + (1 - \varepsilon))U \le U$.

Next, we will prove that for any $X^1(\lambda_3)$ and $X^2(\lambda_3)$, $|X_i^1(\lambda_3) - X_i^2(\lambda_3)| \le \delta_i$ holds:

$$\begin{aligned}
&\left|X_i^1(\lambda_3) - X_i^2(\lambda_3)\right| \\
&= \left|\varepsilon X_i^1(\lambda_1) + (1 - \varepsilon)X_i^1(\lambda_2) - (\varepsilon X_i^2(\lambda_1) + (1 - \varepsilon)X_i^2(\lambda_2))\right| \\
&= \left|\varepsilon(X_i^1 - X_i^2) + (1 - \varepsilon)(X_i^1 - X_i^2)\right| \le \delta_i.
\end{aligned}$$

Theorem 1 shows that each $feasible_i(x)$ is one continuous interval. When a query denial occurs, the adversary has no knowledge of which variable (variables) causes the query denial. The complete information leakage from a query denial is $\lambda \in \bigcup_i feasible_i(x)$, where $\{x_i\}$ are all involved variables. The complete information leakage $\bigcup_i feasible_i(x)$ could be one continuous interval or multiple discrete intervals. Without loss of generality, $\bigcup_i feasible_i(x)$ can be represented by $\bigcup_j [L_j, U_j]$, where $\{[L_j, U_j]\}$ are nonoverlapping intervals. The auditing scheme is to release the single interval $[L_j, U_j]$, which contains the true answer, and carry it over to inspect subsequent queries. Now, the security question is: would releasing $\lambda \in [L_j, U_j]$ threaten data privacy?

Theorem 2. *Suppose the past exact and approximate query answers do not breach data privacy, a new query is denied according to the auditing scheme for sum queries, and* $\bigcup_j [L_j, U_j]$, *where* $\{[L_j, U_j]\}$ *are nonoverlapping, is the derived possible values of the denied answer* $\lambda$. *Returning the interval* $[L_j, U_j]$, *which contains the real query answer, does not threaten data privacy.*

For convenience, we represent past exact answers and approximate answers as $\{A_1 X = b; LB \le A_2 X \le UB\}$, and the new query as $aX$.

It is known that $\bigcup_j [L_j, U_j] = \bigcup_i feasible_i(\lambda)$, where $feasible_i(\lambda)$ is the feasible value of the denied answer

$\lambda$ given $x_i$ causes the query denial. Theorem 1 states that $feasible_i(\lambda)$ is a continuous interval. Therefore the returned interval $[L_j, U_j]$ must be the union of some $feasible_i(\lambda)$.

As such, all involved variables $\{x_i\}$ can be divided into two groups, one group with $feasible_i(\lambda)$ belonging to $[L_j, U_j]$, and the other group with $feasible_i(\lambda)$ not belonging to $[L_j, U_j]$. If $x_i$ has $feasible_i(\lambda) \subseteq [L_j, U_j]$, releasing $[L_j, U_j]$ cannot breach the privacy of $x_i$. It is because $\{A_1X = b; aX \in [L_j, U_j]; LB \le A_2X \le UB\}$ cannot infer information more than $\{A_1X = b; aX \in feasible_i(\lambda); LB \le A_2X \le UB\}$, which only deduces an interval of $x_i$ with length equal to $\delta_i$, in which case $x_i$ is still considered safe. If $x_i$ has $feasible_i(\lambda)$ not belonging to $[L_j, U_j]$, releasing $[L_j, U_j]$ obviously does not affect the privacy of $x_i$ at all.

### 2.4. Deriving Information Leakage

This section studies how to derive $feasible_i(\lambda)$, the possible values of the denied answer $\lambda$, which would limit the feasible solutions of $x_i$ to an interval with length less than or equal to its safe threshold. Suppose we have answered a set of queries, which can be represented by the equation system $\{AX = b, L \le X \le U\}$, where $L \le X \le U$ are prior known bounds for $X$ and have not denied any query yet. Assume a new query $aX$, whose real answer is $\lambda$, arrives. After solving a series of LPs as (8) for all involved variables, the auditor decides to deny the query.

$$\min(\max)\ \ x_i \quad \text{s.t.} \begin{cases} AX = b, \\ aX = \lambda, \\ L \le X \le U. \end{cases} \tag{8}$$

As stated before, from the adversary's perspective, the reason for this denial must be that for some variable $x_i$, the difference of its upper bound and its lower bound is less than or equal to its safe threshold $\delta_i$. The problem of finding $feasible_i(\lambda)$ can be described as follows.

Problem 1. If given $\{AX = b, aX = \lambda, L \le X \le U\}$ for variable $x_i$, what values can $\lambda$ have so that $\max_\lambda(x_i) - \min_\lambda(x_i) \le \delta_i$, where $\max_\lambda(x_i)$ and $\min_\lambda(x_i)$ denote the maximum and minimum values that $x_i$ can take given $\lambda$?

When $\lambda$ is treated as an unknown parameter, the optimization problem (8) becomes a typical right-hand side (RHS) parametric LP problem. A RHS parametric LP problem is a LP problem with a variable (parameter) on the right-hand side of the linear constraints. The study of RHS parametric LP can be traced back to the beginning of operations research (Dantzig 1963). It has been shown that the optimal objective function value of a RHS parametric LP problem is a piecewise linear function of the parameter. There exists an efficient algorithm to derive such a function (Vanderbei 2008). The basic procedure is as follows: first, choose a feasible value of $\lambda$ and determine its characteristic interval, where the objective function optimality does not change; then study adjacent characteristic intervals till the whole real region is traversed.

We adopt the algorithm to deduce $feasible_i(\lambda)$. First, use the algorithm to determine a piecewise function, say, $f_i^1(\lambda)$, for $\max_\lambda(x_i)$, and a piecewise function, say, $f_i^2(\lambda)$, for $\min_\lambda(x_i)$. Then $feasible_i(\lambda)$ is the feasible values of $\lambda$ that make $f_i^1(\lambda) - f_i^1(\lambda) \le \delta_i$.

To illustrate this process, we reconsider the example employed in §2. As explained before, $Q_3$: $x_2 + x_4 = 12$ has to be denied, because it can breach the privacy of $x_1$ and $x_2$. Denote $\lambda$ to be the answer to $Q_3$. Given $Q_3$ being denied, we demonstrate how to compute $feasible_1(\lambda)$, the feasible values of $\lambda$ that make the difference of lower and upper bounds of $x_1$ less than or equal to its safe threshold 5.

$$\max(\min)\ \ x_1 \quad \text{s.t.} \begin{cases} x_1 + x_2 = 20, \\ x_1 + x_3 = 12, \\ x_2 + x_4 = \lambda, \\ x_1, x_2, x_3, x_4 \ge 0. \end{cases} \tag{9}$$

By solving the RHS parametric LP (9), we find that $\max\ x_1$ is 12 when $\lambda \ge 8$, while it is infeasible to have $\lambda < 8$. When $8 \le \lambda \le 20$, $\min\ x_1$ is $20 - \lambda$, and when $\lambda \ge 20$, $\min\ x_1$ is 0. The piecewise functions of $\max\ x_1$ and $\min\ x_2$ dependent on $\lambda$ are depicted in Figure 1. It is not difficult to see that $feasible_1(\lambda)$ is [8, 13].

### 2.5. Discussion

In this section, we briefly discuss the two concerns that some people may have: (i) the auditing scheme releases more information than necessary and (ii) an auditing process needs to solve a large number of LPs.

With the auditing scheme, feasible solutions are maintained in a convex space. As such, whether auditing a query or deriving information from a query

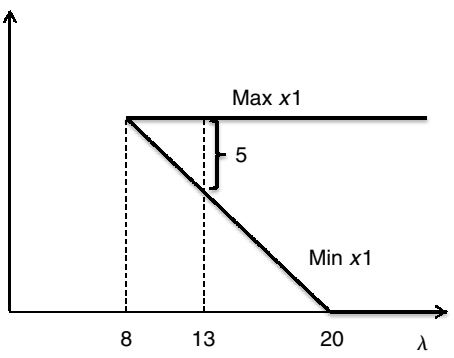


**Figure 1** **Derivation of Information Leakage**

denial, the auditor only needs to solve a number of linear programs. However, the benefit comes at the cost of leaking more information than necessary, as we release the specific interval of the denied answer. If the auditor does not feel comfortable with this scenario, he or she can simply take more computational effort to derive and release all intervals that the denied answer may fall in by employing the same information leakage deriving method and carry them on to audit subsequent queries. In particular, when the first query denial occurs, because the previous queries constitute a polytope, the auditor can use the exact same method to drive the feasible intervals of the denied answer. But for subsequent query denials, the auditor cannot employ the method directly to derive information leakage, because the feasible region of the previously denied answer might be the union of multiple discrete intervals. However, the auditor can derive the feasible intervals for the subsequently denied answer by repetitively employing the information leakage deriving method, because the feasible solution space constituted by the past query answers and denials are the union of multiple polytopes. For each polytope, the auditor can derive intervals that may cause the new query to be denied. The union of all such derived intervals is the complete information leakage.

Whether deriving information leakage from query denials, auditing sum queries needs to solve a large number of LP problems to inspect the difference of the lower and upper bounds for every involved variable. While it is not difficult to solve one LP problem, solving such LP problems is still a huge burden for the system. Worse, the size and number of LP problems keep growing as more queries are issued. Although we cannot avoid those LPs, we can reduce the overall computing time by adopting the strategy employed in Lu et al. (2009). It uses two patterns existing in the LPs formulated in an auditing process. First, at each auditing time, the formulated LPs share the same constraints. Second, the LPs formulated for auditing the next query only have one more constraint than the currently formulated LPs. It is known that a feasible solution of one LP can be quickly constructed at the basis of the solution of another similar LP. Finding a feasible solution is an integral part in simplex methods and typically takes half the computing time of solving a LP problem. By leveraging the similarity of LPs in an auditing process, we can reduce the overall computing time.

## 3. Max and Min

Max and min queries have been less researched. Representative research results include Chin (1986), Kleinberg et al. (2003), Kenthapadi et al. (2005), and Nabar et al. (2006). But none of those schemes can be directly applied to our scenario, i.e., real-valued data regarding interval-based data disclosure policy, free of query denial threat. Chin (1986) proposes the first auditing scheme for max and min queries. Because they assume all queries come together as a batch, their scheme does not consider the query denial issue. The auditing scheme in Kleinberg et al. (2003) is able to eliminate the query denial threat, but at the great loss of data utility. Motivated by Kleinberg et al. (2003), Kenthapadi et al. (2005) propose a simulatable auditing scheme, which effectively counteracts the query denial threat and also improves data utility. But their scheme considers full data disclosure and is applicable to max- or min-only queries. Improved upon Kenthapadi et al. (2005), the auditing scheme in Nabar et al. (2006) can handle mixed max and min queries, but it does not address interval-based disclosure. More importantly, their scheme cannot be applied to mixed query types, including sum, max, min, and deviation. In this section, we modify the auditing scheme in Nabar et al. (2006) in accord with the interval-based data disclosure policy. We will later show how to incorporate the scheme into a general auditing framework for the mixed query types.

A max query $Q_j$ can be represented by $\max(Q_j) = b_j$, where $b_j$ is the query answer. Similarly, a min query $Q_j$ can be represented by $\min(Q_j) = b_j$. Given a set of max and min queries $\{Q_1, \ldots, Q_t\}$ and their answers $\{b_1, \ldots, b_t\}$, the maximum possible value of $x_i$ is $\min(b_j \mid Q_j$ is a *max query*, $x_i \in Q_j)$, denoted by $\mu_i$. Similarly, the minimum possible value of $x_i$ is $\max(b_j \mid Q_j$ is a *min query*, $x_i \in Q_j)$, denoted by $\ell_i$. We will say $x_i$ is a *min extreme* element of $Q_j$, if $Q_j$ is a min query, $x_i \in Q_j$, and $b_j = \ell_i$, and $x_i$ is a *max extreme* element of $Q_j$, if $Q_j$ is a max query, $x_i \in Q_j$, and $b_j = \mu_i$.

The uniqueness of max and min queries comes from their inherent discrete nature. Consider a max query $\max(Q_j) = b_j$; it gives two pieces of information: (i) $\forall x_i \in Q_j$, $x_i \le b_j$ and (ii) $\exists x_i \in Q_j$, $x_i = b_j$. The former is of continuous nature, like sum queries, while the latter is combinatorial oriented. Consider $Q_1$: $\max(x_1, x_2, x_3) = 10$ and $Q_2$: $\max(x_1, x_2) = 5$, where the safe threshold for every variable is 1. $Q_1$ is answered since no privacy disclosure occurs. $Q_2$ is denied, because given $\max(x_1, x_2, x_3) = 10$ and $\max(x_1, x_2) = 5$, $x_3$ becomes the only extreme element of $Q_1$, and thus $x_3$ must be 10. However, if $Q_2$ is denied, an adversary can still infer $x_3$ to be 10, because otherwise $Q_2$ would not be denied. In this case, whether you answer or deny $Q_2$, $x_3$ is disclosed.

We have observed that a max and min query denial can occur in two cases: (i) some variable is fully disclosed as it becomes a max or min extreme element or (ii) some variable is partially disclosed because the difference of its lower and upper bounds is less than or equal to its safe threshold. For the former case, if there exists one element, which could become a max or

min extreme element because of the denied answer, then denying the query would still fully disclose the element. For the latter case, answering a query would not immediately compromise data privacy, because an adversary can only narrow down the denied answer to an interval, in which any possible answer could cause some variable to be partially disclosed. We propose the following auditing scheme.

Definition 7 (Auditing Max and Min Queries). For each new max and min query, we first use the simulatable auditing scheme in Nabar et al. (2006) to check if there exists any possible answer to the query, which is consistent to previous query answers and makes some variable an extreme element.

- If yes, deny the query. (Note that the decision is made without consulting the database.)
- Else, consult the database to see if the real answer would make some element partially disclosed (i.e., the difference of the lower and upper bounds is less than the safe threshold).

—If yes, deny the query and derive the information leakage from the query denial, which will be carried over to audit the subsequent queries.

—Else, answer the query.

The first step of the algorithm is to counteract the query denial effect caused by the discrete nature of max and min queries. Given a set of previously answered max and min queries, $Q_1, \ldots, Q_{t-1}$, the auditing scheme needs to check if there is any possible answer to the new query $Q_t$ that is consistent with past answers and would cause an $x_i$ to be an extreme element. Nabar et al. (2006) showed that it is not necessary to check all possible answers and it suffices to check a finite number of points. In particular, let $Q'_1, \ldots, Q'_l$ be the query sets of previous queries that intersect with $Q_t$, ordered according to their corresponding answers $b'_1 \le \cdots \le b'_l$. We only need to consider each $b_t \in \{b'_{lb}, b'_1, (b'_1 + b'_2)/2, b'_2, \ldots, b'_l, (b'_{l-1} + b'_l)/2, b'_l, b'_{ub}\}$, where $b'_{lb} = b'_1 - 1$ and $b'_{ub} = b'_l + 1$, to check if it is consistent to the previous answers and causes some element to be an extreme element. All of them can be efficiently implemented. The second step is to consult the database to check if the true answer would cause some element to be partially disclosed. If a query is denied, an adversary knows there exists some element, which would be partially disclosed if the answer is released. Without loss of generality, consider the denied query $Q_j$ as a max query. The real answer $b_j$ to the max query $Q_j$ only affects the upper bound of variables in $Q_j$. Therefore, the cause for the query denial is $\exists i \in S_j, b_j - \min(x_i) \le \delta_j$. Because an adversary does not know which one is at risk, the information leakage from the denial of the max query $Q_j$ is $b_j \le \max_{i \in S_j}(\delta_j + \min(x_i))$, where $\min(x_i)$ can be derived from past query responses. Similarly, the denial of the min query $Q_j$ releases $b_j \ge \min_{i \in S_j}(\max(x_i) - \delta_j)$. The released information should be carried along with answered queries to audit the subsequent queries.

## 4. Standard Deviation

Standard deviation has never been studied in the database auditing literature, although it is supported by most SDBs. A standard deviation query $Q_j$ can be represented by $\text{std}(Q_j) = b_j$, computed as

$$\sqrt{\frac{\sum_{i \in Q_j}(x_i - \sum_{t \in Q_j} x_t/|Q_j|)^2}{|Q_j| - 1}} = b_j,$$

where $|Q_j|$ denotes the set size. The standard deviation value $b_j$ can be further expanded as a quadratic function

$$\sum_{i \in Q_j}\left(|Q_j|x_i - \sum_{t \in Q_j} x_t\right)^2 = (|Q_j| - 1)|Q_j|^2 b_j^2.$$

Standard deviation shows how much variation exists from the average. A low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data points are spread out. From the statistical inference perspective, the most risky case would be a standard deviation value being 0, which implies all data points have the same value. However, answers to a set of standard deviation queries without additional information does not improve the lower and upper bounds of involved data points at all, since standard deviation only provides closeness information on data points. If there are prior known bounds of data points, then standard deviation query answers could compromise a database. Consider a database of $\{x_1, x_2, x_3\}$ and it is known that $x_1, x_2 \in [0, 2]$ and $x_3 \in [8, 10]$. If $\text{std}(x_1, x_2, x_3) = 3.4641$, which is the minimum standard deviation value among feasible solutions, then it is disclosed that $x_1$ and $x_2$ are 2 and $x_3$ is 8.

To avoid triviality, we study how to respond to standard deviation queries with elements of prior-known bounds. Look at a batch of standard deviation queries $\{Q_1, \ldots, Q_t\}$; the bounds of variables $X$ are denoted by $[L, U]$. To examine whether the query answers compromise the database, we can formulate and solve a set of nonlinear optimization problems as follows:

$$\min(\max)\ x_i \quad \text{s.t.} \begin{cases} \displaystyle\sum_{i \in Q_j}\left(|Q_j|x_i - \sum_{t \in Q_j} x_t\right)^2 = (|Q_j| - 1)|Q_j|^2 b_j^2, \\ \qquad \text{for } j = 1, \ldots, t, \\ L \le X \le U. \end{cases} \tag{10}$$

Constraints (10) are typical semidefinite programming problems (Vandenberghe and Boyd 1996), because

the value of $\sum_{i\in Q_j}(|Q_j|x_i - \sum_{t\in Q_j} x_t)^2$ is nonnegative for any $X$. It is well known that semidefinite programs can be solved efficiently, both in theory and practice. However, a safe auditing scheme has to consider information leakage from a query denial and carry it over to audit subsequent queries. A good thing is that standard deviation queries are continuously oriented, which is the same as sum queries. In other words, if we deny a standard deviation query when $\exists\, i, \max(x_i) - \min(x_i) \le \delta_i$, the feasible solution space is a convex set or the union of several convex sets. We adopt the same auditing strategy as for sum queries.

Definition 8 (Auditing Standard Deviation Queries). For each new standard deviation query, formulate and solve a series of semidefinite programming problems to compute the lower and upper bounds for each variable. If $\exists\, i, \max(x_i) - \min(x_i) \le \delta_i$, deny the query, find the feasible values of the denied answer that would make the query to be denied, and return the interval of continuous feasible solutions that contain the real answer; else, answer the query.

We release the interval of feasible solutions that contain the real answer so that all following auditing problems are forumulated as convex optimization problems. Although we release more information than necessary, all variable privacy is kept intact. We skip the scheme security analysis, since the proof for the sum query case can be easily extended here, as both query types are convex in nature.

Now we study how to find the feasible values of the denied answer that would deny a standard deviation query. Denote the denied answer as $\lambda$, and the maximum and minimum of $x_i$ dependent on $\lambda$ as $f_i^1(\lambda)$ and $f_i^2(\lambda)$, respectively. Computing $f_i^1(\lambda)$ and $f_i^2(\lambda)$ is a typical RHS parametric semidefinite programming problem. Berkelaar et al. (1996) show that the solution to a RHS quadratic programming is concave and piecewise quadratic. In Goldfarb and Scheinberg (1999), an explicit formula is provided to compute the interval of a RHS parameter value, where the optimal solution (a function of the RHS parameter) is unchanged. In fact, the formula is very similar to the one for RHS parametric LP. Given the formula, we can derive $f_i^1(\lambda)$ and $f_i^2(\lambda)$ and find the feasible solutions $feasible_i(\lambda)$ that make $f_i^1(\lambda) - f_i^2(\lambda) \le \delta_i$. Because an adversary cannot determine which one is at risk, the total information leakage is $\lambda \in \bigcup_i feasible_i(\lambda)$. We then release the interval containing the real answer to enable the subsequent auditing problems to be formulated as convex optimization problems.

## 5. Mixed Query Types

In this section, we provide a consolidated framework for auditing mixed query types, including sum, max, min, and standard deviation. Suppose old queries are $\{Q_1, \ldots, Q_{t_1}, Q'_1, \ldots, Q'_{t_2}, Q'_1, \ldots, Q'_{t_3}\}$, where $\{Q_1, \ldots, Q_{t_1}\}$ are sum queries, $\{Q'_1, \ldots, Q'_{t_2}\}$ max or min queries, and $\{Q''_1, \ldots, Q''_{t_3}\}$ standard deviation queries. The auditing problem is to determine whether to answer or deny a new query $Q_{\text{new}}$. Based on what we have studied in the previous sections, the adversary's knowledge from responses to the past queries can be represented by a set of equality and inequality constraints, which are either in a linear or positive definite quadratic form. Specifically, if a sum query $Q_i$ is answered, the information leakage is $\text{sum}(Q_i) = b_i$. If the query is deemed dangerous, an interval $[l_i, u_i]$ is returned and the information leakage is $l_i \le \text{sum}(Q_i) \le u_i$. Similarly, if a standard deviation query $Q''_i$ is answered, the information leakage is $\text{std}(Q''_i) = b''_i$. Else, an interval is returned and the information leakage is $l''_i \le \text{std}(Q''_i) \le u''_i$. A max or min query $Q'_i$ is either denied or fully answered. If it is denied because there exists a possible answer, which makes some element an extreme element, then there is no information leakage, because the decision is reached without consulting the database. If it is denied because the real answer would make some element partially disclosed, then the information leakage can be represented by $l'_i \le \max/\min(Q'_i) \le b'_i$, such that $\max/\min(Q'_i)$ being any value in $[l'_i, b'_i]$ would make some element partially disclosed. We denote the set of feasible solutions satisfying those constraints, derived from past query responses, by $\mathcal{Q}$. The consolidated auditing scheme for mixed query types is as follows.

Definition 9 (Auditing Mixed Query Types). Given $x \in \mathcal{Q}$, the information leakage from the responses to the past queries, to audit a new query $Q_{\text{new}}$, we do the following:

- If it is a sum query, formulate and solve a set of semidefinite programming problems of $\max/\min(x_i \mid x \in \mathcal{Q},\ \text{sum}(Q_{\text{new}}) = b_{\text{new}})$ for each $x_i$. If there is an $x_i$ being compromised, formulate and solve a set of parametric semidefinite programming problems of $\max/\min(x_i \mid x \in \mathcal{Q},\ \text{sum}(Q_{\text{new}}) = \lambda)$ for each $x_i$ to derive the feasible values of $\lambda$ that would compromise the database, and then return the interval containing the exact answer.
- If it is a standard deviation query, run the same procedure as above, except for replacing the constraint $\text{sum}(Q_{\text{new}}) = b_{\text{new}}$ with $\text{std}(Q_{\text{new}}) = b_{\text{new}}$.
- If it is a max or min query, check whether there exists a possible answer that is consistent to past answers and causes the existence of extreme elements.
  - —If yes, deny the query (note the decision is made without consulting the database).
  - —If no, consult the database to see if the real answer makes some element partially compromised.
    - ∗ If yes, deny the answer and derive information leakage.
    - ∗ If no, answer the query.

The framework treats max and min queries differently, due to their inherent discrete nature. In the first step of auditing a max or min query, there is no need to check all possible answers. It suffices to check a finite number of points, as what we did to audit max and min-only queries. Suppose it is a max query. For consistency checking, we use a method in Vandenberghe and Boyd (1996) to check the feasibility of the semidefinite constraint set $\{x \in \mathcal{Q}; x_i \le b'_i, \forall x_i \in Q_{\text{new}}\}$. To determine the existence of extreme elements, we adopt the same algorithm used in Nabar et al. (2006). To derive the information leakage from a query denial, we formulate and solve parametric semidefinite programming problems to find out the possible values of $b'_i$ that would make some element partially disclosed. We take the interval containing the real answer as the information leakage and carry it over to audit the subsequent queries.

## 6. Experimental Study

An auditing scheme can be evaluated in three dimensions: privacy, utility, and efficiency. Privacy is the most important because it is the reason for the existence of a privacy protection mechanism. Data utility determines the usage of a database, and efficiency affects user experience. Unanimously, privacy can be examined with regard to the defined data privacy policy and efficiency can be measured in running time. There are many ways to measure utility. For example, Nabar et al. (2006) measure utility by the number of answered queries. But an auditing scheme could provide ambiguous and partial answers to unlimited queries without compromising a database and that does not make the auditing scheme preferable. So we measure utility by the amount of released information on database elements. Regarding the interval-based privacy policy, the best auditing scheme is the one that allows a user to infer the feasible database solutions to a polytope with $\max(x_i) - \min(x_i) = \delta_i, \forall x_i$, where $\delta_i$ is the predefined safe threshold of $x_i$.

### 6.1. Sum

The first experiment is to study sum-only queries. We compared our auditing scheme with conventional auditing and simulatable auditing. Note that we do not consider the modified auditing scheme, as it is essentially the same as the conventional auditing scheme. We generated a database of 100 elements with values randomly drawn from $[1, 100]$, and 200 random sum queries uniformly drawn from the pool of possible sum queries. For each element, we set its safe threshold to be 0.1 times its value. We run all auditing schemes against the same database and queries, and report the results in Figures 2–4, where each diamond denotes a query and the dark diamond means the database is compromised. We observe the following: First, in terms of privacy, both our auditing scheme and simulatable auditing protects the privacy throughout the auditing process, whereas conventional auditing is unable to protect privacy after query 90. We also notice that the first abnormal query response occurs at query 90 for both conventional and our auditing scheme. Conventional auditing denied the query, while our auditing scheme partially answered the query. After that, conventional auditing answered query 91, which immediately compromised the database, due to the ignorance of information leakage of the denial of query 90. As the result, the database continued to stay in the compromised state, even though the most subsequent queries were denied. In contrast, our auditing scheme partially answered queries 90–92 and 94–200, and fully answered query 93. It provides data information and maintains data privacy. Second, regarding utility, our auditing scheme provides the maximum data utility, because at query 96, the privacy boundaries had been reached. In contrast, conventional auditing releases more information than the privacy policy allows, which is unacceptable, and simulatable auditing does not answer any query, because all elements are known to fall in a bounded range. Third, in terms of efficiency, our auditing scheme takes the most computing. However, the computing time of our auditing scheme is comparable to that of conventional auditing. Firstly, for queries 1–89, our auditing scheme and conventional auditing took the same amount of time because no query denial occurs yet. For queries 90–95, our auditing scheme took more computing time, as it needs to derive information leakage. But after query 96, our auditing scheme took the same amount of time as conventional auditing, because the privacy boundaries were reached and there was no need to derive information leakage. To provide a partial answer,

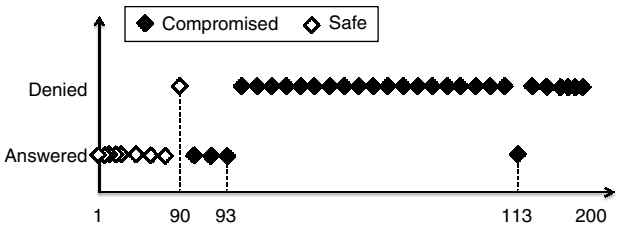


**Figure 2** **Conventional**

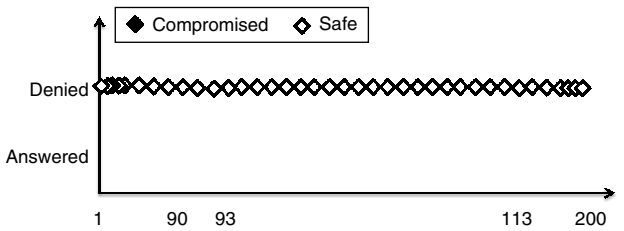


**Figure 3** **Simulatable**

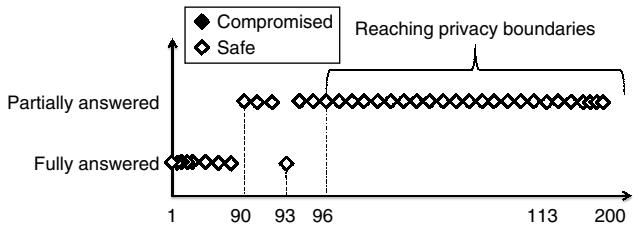


**Figure 4 Our Auditing Scheme**

our auditing scheme only needs to formulate and solve two LPs with the information derived from queries 1–96 as constraints to obtain the lower and upper bounds of the query.

Simulatable auditing took the least computing time, but with no query being answered. The zero data utility is because the simulatable auditing scheme is too strict. Recall that the simulatable auditing scheme denies a query if there is a feasible answer, which can cause privacy disclosure. For a first query $\sum_i x_i$ against a nonnegative database, a feasible and consistent answer is 0, which indicates every variable is zero. If the answer is 0, all variables are uniquely determined to be 0. Therefore the query must be denied, as must all subsequent sum queries. The example shows that the simulatable auditing scheme is not suitable for cases where variables have prior known bounds. However, databases with prior known bounds are very common in practice. For instance, salary is nonnegative, age is a positive number less than 150, etc.

### 6.2. Mixed

The second experiment is to study mixed query types. First, we present a result on a small data set, depicted in Figure 5, to provide some insights. The data set comprises 20 elements $\{x_1, \ldots, x_{20}\}$ with values randomly drawn from $[1, 20]$. The task is to audit 100 random queries generated by the following two-step procedure: (i) determine a random number $k$ in $[1, 20]$ and then select $k$ random elements from $\{x_1, \ldots, x_{20}\}$, and (ii) randomly specify the query type so that sum, max or min, and standard deviation have the same probability. The safe threshold is 0.1 times each element's value. We made the following observations: (i) Eight sum queries are fully answered. The number would be much larger if we were auditing sum-only queries. For a database of $n$ elements, element values would be fully disclosed by $n$ linearly independent sum queries. However, for reasonable safe threshold values, the number of fully answered queries should be close to the number of total elements, which has been verified by many existing studies. One recent evidence is Figure 4, in which 90 queries over a database of 100 elements are answered. Figure 5 shows that as answers to max, min, and standard deviation queries leak information, the number of answered sum queries is significantly reduced. (ii) Four max or min queries are fully answered and the rest are denied, not partially answered. So the simulatable auditing step incorporated in our scheme to counteract the discrete nature threat of max and min queries is the sole cause for them to be denied. (iii) There are 13 standard deviation queries being answered, which is larger than the sum of answered sum, max, and min queries. It shows that an answer to a standard deviation query does not release much information relatively. (iv) After query 40, the privacy boundaries are reached; in other words, the maximum utility is achieved, while the privacy is well kept. It also tells us that for the remaining queries, there is no need to apply our sophisticated auditing scheme, as no additional information can be released. An auditor can simply answer the query based on released information without consulting the database.

To validate the observations made from the previous single case, we conducted more experiments. We generated five databases with the number of elements ranging from 20 to 100, with the same generation procedure as that for the previous example. The results are reported in Figure 6 with the patterns matching the findings from Figure 5. It is observed that (i) the query types in the order of the number of queries being fully answered are standard deviation, sum, and max or min and (ii) the total number of queries being fully answered increase as the database size increases.

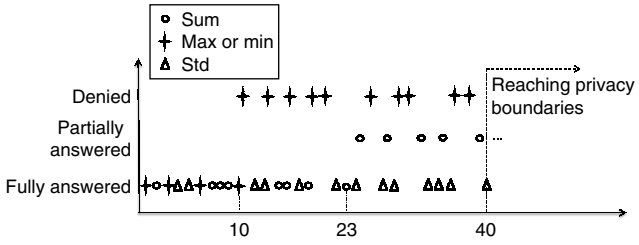


**Figure 5 Auditing Mixed Query Types**

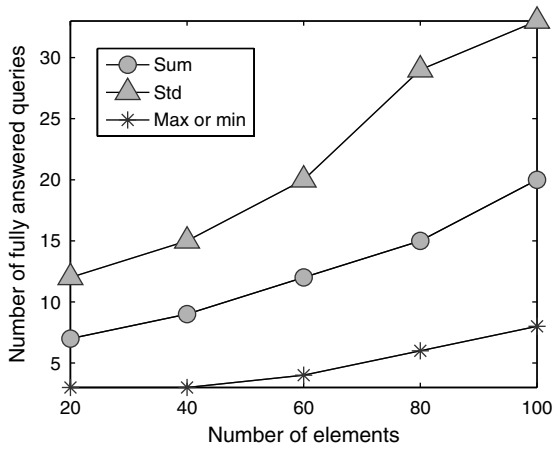


**Figure 6 More Results on Mixed Query Types**

We noticed that almost no max or query was partially answered. Therefore we conducted more experiments by removing the step of using the simulatable auditing strategy to eliminate the discrete nature threat of max and min queries. We found that nearly all max and min queries were fully answered. As element values are uniformly drawn and widely dispersed, answers to max and min queries do not improve the lower and upper bounds of elements by much. We conclude that the discrete nature is the main cause for fewer max and min queries being answered.

### 6.3. Trade-off

In this section, we study the trade-offs along different dimensions. Because privacy is the principle of auditing and should never be compromised, what we can do is exchange utility in return for efficiency. There are many ways to improve efficiency: one way is to utilize the algorithm in Kleinberg et al. (2003) to eliminate occurrence of extreme elements, when auditing a max or min query, instead of the algorithm in Nabar et al. (2006). Kleinberg et al. (2003) show that given a collection $\{Q_1, \ldots, Q_t\}$ of max (or min) queries, for element $i$ to be a max extreme element, there must exist $Q_j \backslash (\bigcup_{r \in c} Q_r) = \{i\}$, where $c$ is a subcollection. To eliminate occurrence of extreme elements, we can deny a max (or min) query without consulting the database if there exists $Q_j \backslash (\bigcup_{r \in c} Q_r) = \{i\}$ for some element $i$, which can be implemented efficiently. The consequence is that more innocuous queries would be denied. If we use the modified scheme to audit max- and min-only queries, the probability of a query denial would be increased. If we audit mixed query types, including sum, max, min, and deviation, and consider reaching privacy boundaries is the maximum utility, the modified scheme does not reduce utility. The only effect is fewer max and min queries would be answered. To find out the detailed trade-off effect, we executed the original auditing scheme and the modified scheme with the algorithm in Kleinberg et al. (2003) being plugged against the five databases created for the previous experiment. The results are reported in Figure 7. The left vertical axis denotes the ratio of fully answered max and min queries by the original scheme to the number of answered max and min queries by the modified scheme and the right vertical axis represents the ratio of average auditing time for a max or min query by the original scheme to the time by the modified scheme. Figure 7 shows that the original scheme takes significantly more computing time, because the original scheme requires solving a large number of optimization problems, while the modified scheme takes almost no time. However, the utility gain at the great cost of computing time is not that significant, as illustrated by the left vertical axis of Figure 7. We conclude that if it is not critical to answer max and min queries, it is advisable to take the modified scheme.

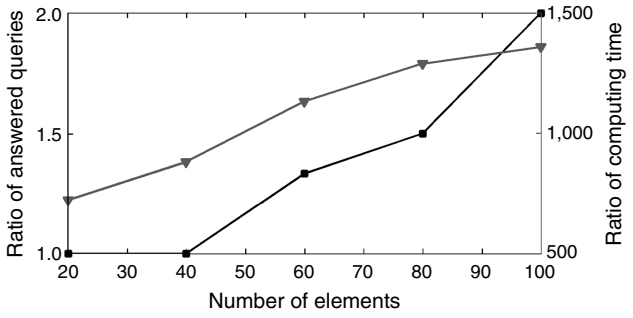


**Figure 7 Answered Queries vs. Computing Time**

Another way to improve efficiency is to deny a sum query right after the first sum query denial occurs. As observed in Figure 5, after the first query denial, the privacy boundaries will be quickly approached. Because deriving information leakage takes a lot of time, if we simply deny all queries after the first query denial, then much time is saved. We conducted an experiment to compare the original scheme with the "lazy" scheme regarding computing time and released information on five synthetic databases, with number $n$ of elements ranging from 20 to 100 and element values drawn from $[0, n]$. All queries to be audited are sum queries and the threshold value is 0.1 times an element value. The maximum knowledge on $x_i$ that a user is allowed to obtain is the bounds being improved from $[0, n]$ to $[l_i, u_i]$, where $u_i - l_i = 0.1 \times x_i$. Comparing the original scheme and the "lazy" scheme regarding the released information of $x_i$, we take the measure of $(n - 0.1 \times x_i)/(n - (u_i' - l_i'))$, where $u_i'$ and $l_i'$ are the lower and upper bounds of $x_i$ at the end of the "lazy" scheme. We compute the average measure over all elements for each database and report the results in the left vertical axis

ratio of the total auditing time of the original scheme to that of the "lazy" scheme. We observed that the "lazy" scheme provides decent information with less computing time. We conclude that if efficiency is a top concern for a system, it is worth trying the "lazy" scheme.

There are many other ways to improve efficiency. For example, we could incorporate other privacy mechanisms into the auditing mechanism, e.g., perturbation might be the easiest way to achieve data privacy. Keep in mind that doing so would lose many properties of auditing, such as flexibility and accuracy. As reported in Dinur and Nissim (2003), for a SDB by an $n$-bit string $d_1, \ldots, d_n$ with a query being a subset $q \subseteq [n]$ to be answered by $\sum_{i \in q} d_i$, to achieve privacy, one has to add perturbation of magnitude $\Omega(\sqrt{n})$. For SDBs of positive integers $x_1, \ldots, x_n$, the perturbation needs to be of magnitude $\Omega(\sqrt{\sum_i x_i})$. To illustrate this, we did an experiment on a synthetic database of 100 elements with five sum queries. Figure 9 reports the

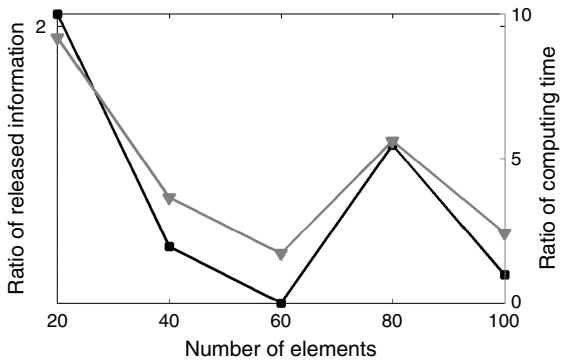


**Figure 8** **Released Information vs. Computing Time**

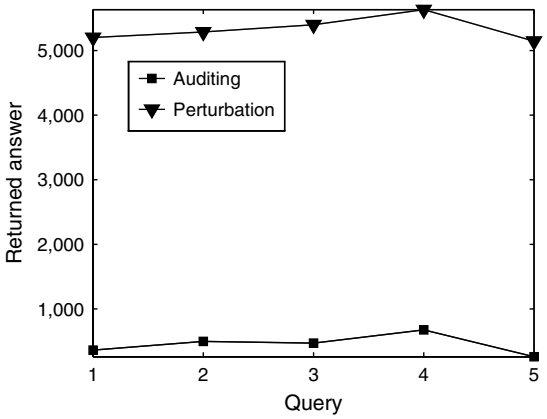


**Figure 9** **Comparison of Released Answers**

answers returned by the auditing scheme and from a perturbed database. We observed that the perturbation mechanism significantly alters real answers. Therefore, perturbation might only be suitable for large data sets where auditing is not feasible.

Note that it has long been recognized that "...auditing may serve as a solution to the SDB security problem for small SDBs" (Chin and Özsoyoglu 1982, p. 575). Auditing provides the complete data privacy, maximum data utility, and query flexibility at a great computational cost. Auditing is typically formulated as a set of linear programs, therefore the scalability and practicability of the auditing approach largely depends on the state of art in optimization technologies. For instance, the current front line large-scale LP/QP solver engine[1] can solve linear and quadratic programming problems with up to 32,000 variables and 32,000 constraints in the standard version. It is reasonable to apply the auditing approach to SDBs with thousands of variables or less. As optimization technologies advance, the practicability of auditing will increase accordingly. Significant advancements on optimization technologies have been observed since the 1970s when auditing was first introduced. In addition, our auditing algorithm is faster than the conventional auditing algorithms. As briefly mentioned in §2.5, our algorithm takes advantage of the two patterns observed in the set of linear programs formulated in an auditing process, which allow us to utilize the sensitivity analysis technologies used in optimization to reduce half the computing time.

[1] http://www.solver.com/large-scale-lpqp-solver-engine.

## 7. Conclusion

In this paper, we present an auditing framework, which is applicable to mixed query types, including sum, max, min, and deviation. The framework provides the maximum data utility and is free of query denial threat. The key idea is acknowledging the fact that query denials leak information. Upon each query denial, we derive information leakage and treat it as a part of the adversary knowledge when auditing subsequent queries. Due to the discrete nature of max and min queries, when auditing a max or min query, we employ the simulatable auditing strategy to eliminate the occurrence of extreme elements. The experimental study shows that our scheme provides the maximum data utility to users, as the privacy boundaries are reached for each case. Experimental results also show that standard deviation has more queries being answered than other types, which is because a standard deviation query does not release much information. We also observed that max and min have the least queries that can be answered, because the simulatable auditing strategy that we added to the whole auditing process denies many innocuous queries. Designing a better auditing algorithm for max and min queries will be our future work.

### References

Adam NR, Wortmann JC (1989) Security-control methods for statistical databases: A comparative study. *ACM Comput. Surveys* 21:515–556.

Agrawal S, Budetti P (2012) Physician medical identity theft. *JAMA* 307:459–460.

Berkelaar AB, Jansen B, Roos K, Terlaky T (1996) Sensitivity analysis in (degenerate) quadratic programming. Technical Report 96-26, (Delft University of Technology, Delft, the Netherlands).

Castro J (2007) A shortest-paths heuristic for statistical data protection in positive tables. *INFORMS J. Comput.* 19:520–533.

Chin FY (1978) Security in statistical databases for queries with small counts. *ACM Trans. Database Systems* 3:92–104.

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

Chin FYL, Özsoyoglu G (1981) Statistical database design. *ACM Trans. Database Systems* 6:113–139.

Chin FYL, Özsoyoglu G (1982) Auditing and inference control in statistical databases. *IEEE Trans. Software Engrg.* 8:574–582.

Chowdhury SD, Duncan GT, Krishnan R, Roehrig SF, Mukherjee S (1999) Disclosure detection in multivariate categorical databases: Auditing confidentiality protection through two new matrix operators. *Management Sci.* 45:1710–1723.

Dantzig GB (1963) *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ).

Dinur I, Nissim K (2003) Revealing information while preserving privacy. *Proc. Twenty-Second ACM Sympos. Principles Database Systems* (ACM, New York), 202–210.

Dobkin D, Jones AK, Lipton RJ (1979) Secure databases: Protection against user influence. *ACM Trans. Database Systems* 4:97–106.

Dwork C (2008) Differential privacy: A survey of results. *TAMC* 4978:1–19.

Fischetti M, Salazar JJ (2001) Solving the cell suppression problem on tabular data with linear constraints. *Management Sci.* 47: 1008–1027. .

Friedman AD, Hoffman LJ (1980) Towards a fail-safe approach to secure databases. *IEEE Sympos. Security and Privacy, Oakland, CA.*

Fung BCM, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surveys* 42:14:1–14:53.

Garey MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York).

Garfinkel R, Gopal R, Goes P (2002) Privacy protection of binary confidential data against deterministic, stochastic, and insider threat. *Management Sci.* 48:749–764.

Goldfarb D, Scheinberg K (1999) On parametric semidefinite programming. *Appl. Numer. Math.* 29:361–377.

Kadane JB, Krishnan R, Shmueli G (2006) A data disclosure policy for count data based on the COM-Poisson distribution. *Management Sci.* 52:1610–1617.

Kaelber DC, Jha AK, Johnston D, Middleton B, Bates DW (2008) A research agenda for personal health records (phrs). *J. Amer. Medical Informatics Assoc.* 15:729–736.

Kenthapadi K, Mishra N, Nissim K (2005) Simulatable auditing. *Proc. Twenty-Fourth ACM Sympos. Principles Database Systems* (ACM, New York), 118–127.

Kleinberg JM, Papadimitriou CH, Raghavan P (2003) Auditing Boolean attributes. *J. Comput. Syst. Sci.* 66:244–253.

Kumar R, Gopal R, Garfinkel R (2010) Freedom of privacy: Anonymous data collection with respondent-defined privacy protection. *INFORMS J. Comput.* 22:471–481.

Lee S, Genton MG, Arellano-Valle RB (2010) Perturbation of numerical confidential data via skew-*t* distributions. *Management Sci.* 56:318–333.

Li N, Li T, Venkatasubramanian S (2007) *t*-Closeness: Privacy beyond *k*-anonymity and *l*-diversity. Chirkova R, Dogac A, Tamerözsu M, Sellis TK, eds. *Proc. 23rd IEEE Internat. Conf. Data Engrg.* (IEEE Computer Society, Los Alamitos, CA), 106–115.

Li X-B, Sarkar S (2006) Privacy protection in data mining: A perturbation approach for categorical data. *Inform. Systems Res.* 17:254–270.

Li X-B, Sarkar S (2011) Protecting privacy against record linkage disclosure: A bounded swapping approach for numeric data. *Inform. Systems Res.* 22:774–789.

Li X-B, Sarkar S (2013) Class-restricted clustering and microperturbation for data privacy. *Management Sci.* 59:796–812.

Li Y, Lu H (2008) Disclosure analysis and control in statistical databases. *ESORICS, Lecture Notes in Computer Science*, Vol. 5283 (Springer, New York), 146–160.

Li Y, Wang L, Jajodia S (2003) Preventing interval-based inference by random data perturbation. *Proc. 2nd Internat. Conf. Privacy Enhancing Tech., San Francisco*, 160–170.

Lu H, Li Y (2008) Practical inference control for data cubes. *IEEE Trans. Dependable Sec. Comput.* 5:87–98.

Lu H, Li Y, Atluri V, Vaidya J (2009) An efficient online auditing approach to limit private data disclosure. *ACM Internat. Conf. Extending Database Tech.* (ACM, New York), 636–647.

Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) *l*-Diversity: Privacy beyond *k*-anonymity. *IEEE Internat. Conf. Data Engrg.* (IEEE Computer Society, Los Alamitos, CA), 24.

Malvestuto FM, Moscarini M (2006) Auditing sum-queries to make a statistical database secure. *ACM Trans. Inform. System Security* 33:451–464.

Matloff NS (1986) Another look at the use of noise addition for database security. *IEEE Sympos. Security Privacy* (IEEE Computer Society, Los Alamitos, CA), 173–181.

Muralidhar K, Sarathy R (2006) Data shuffling—A new masking approach for numerical data. *Management Sci.* 52:658–670.

Muralidhar K, Batra D, Kirs PJ (1995) Accessibility, security, and accuracy in statistical databases: The case for the multiplicative fixed data perturbation approach. *Management Sci.* 41:1549–1564.

Muralidhar K, Parsa R, Sarathy R (1999) A general additive data perturbation method for database security. *Management Sci.* 45:1399–1415.

Nabar SU, Marthi B, Kenthapadi K, Mishra N, Motwani R (2006) Towards robustness in query auditing. *Proc. 32nd Internat. Conf. Very Large Data Bases, Seoul, Korea*.

Nunez MA, Garfinkel RS, Gopal RD (2007) Stochastic protection of confidential information in databases: A hybrid of data perturbation and query restriction. *Oper. Res.* 55:890–908.

Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: *k*-anonymity and its enforcement through generalization and suppression. Technical reportT, SRI International, Menlo Park, CA.

Sarathy R, Muralidhar K, Parsa R (2002) Perturbing nonnormal confidential attributes: The Copula approach. *Management Sci.* 48:1613–1627.

Schlorer J (1975) Confidentiality of statistical records: A threat-monitoring scheme for on line dialgoue. *Methods Inform. Medicine* 14:36–42.

Sweeney L (2002) *k*-anonymity: A model for protecting privacy. *Internat. J. Uncertainty Fuzziness Knowledge-Based Systems* 10: 557–570.

Vandenberghe L, Boyd S (1996) Semidefinite programming. *SIAM Rev.* 38:49–95.

Vanderbei RJ (2008) *Linear Programming: Foundations and Extensions*, 3rd ed. (Springer-Verlag, New York).

Wang L, Jajodia S, Wijesekera D (2004) Securing OLAP data cubes against privacy breaches. *IEEE Sympos. Security Privacy* (IEEE Computer Society, Los Alamitos, CA), 161–175.

Wang L, Li Y, Wijesekera D, Jajodia S (2003) Precisely answering multi-dimensional range queries without privacy breaches. *Eur. Sympos. Res. Comput. Security* (Springer, New York), 100–115.