

An Efficient Online Auditing Approach to Limit Private Data Disclosure

Haibing Lu
MSIS and CIMIC, Rutgers
University
haibing@cimic.rutgers.edu

Vijayalakshmi Atluri
MSIS and CIMIC, Rutgers
University
atluri@cimic.rutgers.edu

Yingjiu Li
Singapore Management
University
yjli@smu.edu.sg

Jaideep Vaidya
MSIS and CIMIC, Rutgers
University
jsvaidya@cimic.rutgers.edu

ABSTRACT

In a database system, disclosure of confidential private data may occur if users can put together the answers of past queries. Traditional access control mechanisms cannot guard against such breaches to private data. Online auditing techniques have been advanced to limit such disclosure of private data. Essentially, before answering any query, these techniques inspect the answers of the past queries to determine whether answering this query would compromise the stated data disclosure policies. While the primary requirement for online auditing is high efficiency, existing auditing approaches are expensive with respect to both computational time and space. Specifically, this cost is excessive in the general case of auditing arbitrary aggregate queries over real-valued confidential attributes with respect to interval-based privacy disclosure.

In this paper, we model this problem as the well-studied linear programming (LP) problem and propose an efficient online auditing solution for constantly monitoring the bounds of protected attributes. The previously proposed approaches in this direction repetitively employ the LP. Consequently, for each new query, they require evaluation of the entire set of answers to past queries. In this paper, we propose a novel approach to employ LP that keeps the prior evaluation state in a concise form and conducts an incremental evaluation. Basically, our approach treats the online auditing problem as a series of updation problems. Each time when a new query is issued by a user, instead of solving a new LP problem with up-to-date log of all queries, we modify the existing bounds obtained in auditing previous queries based on certain rules so as to get the updated bounds with the new query added. Since it significantly reduces the computation time and storage space, our approach offers the first practical solution for the interval-based online auditing problem.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the ACM. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires a fee and/or special permissions from the publisher, ACM. *EDBT 2009*, March 24–26, 2009, Saint Petersburg, Russia. Copyright 2009 ACM 978-1-60558-422-5/09/0003 ...\$5.00

Our experimental results demonstrate that our solution is about 30 times faster than the existing solutions.

1. INTRODUCTION

With the fast development of computer technology and the wide spread use of networks, people can now have unprecedented access to huge amount of information much easier. While this enhances availability, it puts private data at risk. Though access control can be used to protect private information of individuals (e.g., salary) from being accessed by unauthorized users, it does not prevent this information from being disclosed via seemingly innocuous queries (e.g., SUM, GROUP BY, AVERAGE, COUNT, etc.). Consider the following toy example. A user may submit the following query to a database payroll with attributes including title, sex, department, and salary, where the attribute salary is protected.

```
SELECT SUM(salary)
FROM payroll
WHERE title=professor, department=Biology;
```

Assume that there are three professors in the Biology department and the sum of their salaries is \$450,000. Since this query does not enable the user to gain access to the salary directly, it is allowed. The query and its answer can be written as $x_1 + x_2 + x_3 = 450,000$. Similarly, the user may issue another query and obtain its answer as $x_1 + x_2 = 300,000$ via the following query

```
SELECT SUM(salary)
FROM payroll
WHERE title=professor, sex=male, department=Biology;
```

However, combining the answers to the above two queries, the user is able to infer that the salary of the only female professor in the Biology department is \$150,000. This simple example illustrates that it is essential to determine whether any SQL query reveals information forbidden by the database system's data disclosure policies by examining past queries. This problem, known as *auditing* [10] of queries by inspecting the answers of the past queries to determine whether these answers could have been put together by a user to infer confidential information. The process of performing such

auditing whenever a new query arrives, is known as *online auditing* [7].

The first question in online auditing is how to measure privacy disclosure. Three types of privacy disclosure measures have been considered in the research literature: exact-value disclosure, interval-based disclosure, and probability-based disclosure. The exact-value disclosure [7, 24, 32] means that an individual's privacy is disclosed iff the exact value of a protected attribute for that individual is disclosed. It is also called full disclosure. The interval-based disclosure [27] generalizes the concept of exact-value disclosure, stating that an individual's privacy is disclosed iff the difference of the lower and upper bounds of a protected attribute for that individual being disclosed to be no greater than a predetermined threshold. In the above example, the female professor's privacy is considered to be disclosed if her salary is known to be between \$145,000 and \$155,000 given the threshold being \$15,000. It is clear that the exact-value disclosure can be considered as a special case of the interval-based disclosure where the threshold is zero. The probability-based disclosure means the privacy disclosure occurs when the posterior distribution of the data after answering queries is significantly different from its prior distribution that the snooper knows at the beginning [23, 37]. Probability-based disclosure is sound. However, the work of calculating the distribution function change for each variable caused by the change of the geometrical polytope constructed by the answered queries is inhibitive. Even if the computing time is not an issue for probability-based disclosure, we need to monitor the change of the bounds for each variable anyway. In [23], to quantify the difference of prior and posterior distributions for each variable, they divided the whole real domain into small intervals and counted the sampled data falling in each interval before and after the new query to see the difference. For each variable, we actually need to check some intervals determined by their bounds, instead of the whole real domain. Considering all above facts, we choose to study interval-based disclosure over the other two measures.

In particular, we focus on auditing arbitrary SUM queries over a real-valued attribute that is protected. A up-to-date log of all previously answered queries is maintained for each user in auditing. A new query is denied if it enables its issuer, when combined with the corresponding query log, to infer that the difference between the bounds of any protected value is no greater than a predetermined threshold; otherwise, it is answered and incorporated in the query log.

Auditing interval-based disclosure is equivalent to monitoring the changes of upper and lower bounds for all appeared variables. When SUM queries arrive in an arbitrary manner, employing linear programming (LP) is the most appropriate way to go because there is no extra information helping us to deduce the bounds of variables. If additional constraints are added, there might be more efficient approaches available, other than linear programming, such as audit expert system for exact-value disclosure [10] and Fre'chet bound for the 2-dimensional OLAP [26]. Compared to these, LP is really a powerful tool for auditing interval-based disclosure, because there are several efficient algorithms available for it, like primal simplex method, dual simplex method and interior point method [44]. However, the existing auditing approach [28]

simply proposes to employ the LP algorithm repetitively to calculate the upper and lower bounds for each variable upon the arrival of every new query. Though a single LP problem can be efficiently solved, the existing auditing approach requires excessive CPU time and storage as it involves solving a large number of LP problems for all protected values that appear in a user's query history. The worse thing is the size of the LP problem becomes larger when more queries come.

[23] pointed out that denying a query could leak extra information to snoopers in some cases. So they proposed simulatable auditing. Simulatable auditing means the decision of answering or denying a query can be deduced by both auditor and snooper. In other words, the auditor makes the decision only based on the past answered queries with the newly posted query, without accessing the internal data values. Thus, if there exists a result to the newly posted query, which is consistent with the past queries, causing privacy disclosure for one variable, the query is denied. The simulatable auditing loses much utility, because it denies many queries which are not harmful indeed, although it keeps full privacy. The other drawback is the computation is intimidating, because for each consistent result, the auditor needs to examine if there is privacy disclosure occurring, let alone the work of finding the consistent result. Considering these facts and that the information leaked by denying a query is rather limited, especially when a number of variables involved, in this paper we do not consider the effect caused by denying a query.

In this paper, we propose a novel approach that performs incremental evaluation to determine whether or not to allow a new query. Essentially, we treat the online auditing problem as a series of updation problems. More specifically, we use horizontal updation and vertical updation to take advantage of the relationship among different LP problems in online auditing. A key observation is that, given a set of queries, the formulated LP problems for auditing different protected values only differ in the objective functions, while the constraints remain exactly the same. The horizontal updation opens the black box of these formulated LP problems and modifies the solution of one LP problem to quickly derive the solutions of the other LP problems without having to solve them repetitively. When a new query is issued by a user, a new LP problem is formulated for auditing each protected value. The formulated LP problem is the same as the existing LP problem for auditing the same value except that one more constraint representing the new query is added. The vertical updation is designed to create a shortcut between the existing LP and the new LP in auditing. Essentially, the vertical updating is similar to the horizontal updating in the way in which it solves a new LP problem by utilizing the result of another approximate LP problem. However, the required techniques are different.

Although the LP approach for interval-based auditing problem on real value has been proposed earlier, it is not viable because online auditing calls for fast response time and small storage space. When compared to the earlier solutions, our proposed approach enjoys many advantages. (1) Both theoretical analysis and numerical experiments show that our solution is significantly more efficient than the existing auditing approach. Our solution is about 30 times faster than

the existing solution for auditing 30 queries in our numerical experiments. We also observe that our solution is increasingly more efficient when auditing more and larger queries. (2) Besides the advantage in computation time, our algorithm requires much less storage space, because through the whole auditing process, we store only one simplex tableau, but not the log of all queries. (3) It allows every user to specify a different level of privacy.

To the best of our knowledge, our auditing solution represents the first significant improvement over the existing LP-based approach in the general case of auditing arbitrary SUM queries over a real-valued attribute in terms of interval-based disclosure. Since our solution fares significantly better with respect to time and space complexity, we believe it is the first practical solution for the online auditing problem.

The rest of this paper is organized as follows. In Section 2, we review the literature work related to disclosure control and auditing. In Section 3, we provide the preliminary knowledge about LP, which will be exploited in the following sections. In Section 4, we formulate the online auditing problem and treat it as a series of updating problems including horizontal updating and vertical updating. In Sections 5 and 6, we present the horizontal updating and vertical updating, respectively. In Section 7, we incorporate horizontal updating and vertical updating to construct our online auditing algorithm and analyze its efficiency. In Section 8, we conduct numerical experiments to evaluate the efficiency of our online auditing algorithm. Finally, in Section 9, we conclude the paper and discuss some future research directions.

2. RELATED WORK

Traditionally, auditing is considered to be one of the security control methods for statistical databases [1, 16, 48]. Other security control methods can be classified into restriction-based and perturbation-based. The restriction-based techniques protect privacy information from being disclosed by imposing restrictions. The restrictions can be placed on queries [46, 47] such as the size of each query [13], the overlap among different queries [14], and the types of queries [46, 47], or they can be placed on source data such as partition [9, 41], microaggregation [15, 48], generalization [21, 45], and suppression [11, 12, 17, 18]. The perturbation-based techniques protect sensitive information by adding random noises. The random noises can be added to query answers [5], data structures [40], or source data [2, 3, 6, 20, 22, 29, 35, 43]. Compared to these methods, auditing has advantages such as allowing database systems to provide users unperturbed query results as long as the results will not lead to any privacy disclosure without imposing additional restrictions.

Our work is different from the recent research on k -anonymity [39, 42], ℓ -diversity [31], and t -closeness [25] for protecting published data about individuals without revealing privacy information about them. The k -anonymity privacy requires that each equivalence class (i.e., a set of records that are indistinguishable from each other) consists of at least k records. The ℓ -diversity further enhances the requirement that each equivalence class contains at least ℓ well-represented values for each sensitive attribute, while the t -closeness privacy requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute

in the whole dataset. These privacy requirements can be used to evaluate certain anonymization process that partitions the protected data into equivalence classes before publication of data. In comparison, we address the problem of protecting a sensitive attribute in a database which may not necessarily be anonymized. In addition, the database is not published; instead, it is protected against privacy disclosure via user queries.

The auditing problem has been investigated in different cases, mostly for auditing SUM queries (except [7]). Chin and Ozsoyoglu [8, 10] investigated the auditing problem over a protected attribute with respect to exact-value disclosure. They designed a system called audit expert, which maintains a binary matrix to efficiently represent a user’s knowledge about the protected attribute. It has been shown that the audit expert takes $O(n^2)$ time for processing a new query given n database records, and $O(mn^2)$ time for auditing a set of m queries. The audit expert remains one of the most efficient auditing algorithms with respect to exact-value disclosure.

Different types of protected attributes have been investigated in auditing. The auditing of boolean attributes have been proven to be NP-hard [24]. The auditing of integer attributes is NP-hard as well [27]. However, they can be relaxed to real problem at the cost of losing accuracy. The auditing of real attributes is polynomial due to the existence of polynomial LP solutions such as the interior-point methods [44]. From practical point of view, the simplex method is unarguably among most efficient methods [44] to solve the problem of auditing real attributes.

Auditing can be performed over either well-structured queries or arbitrary queries. The auditing of well-structured queries such as OLAP queries [46], data cube queries [26], and range queries [47] can be more efficient than the auditing of arbitrary queries since the structural information of queries can be utilized to speed up the auditing process. In this paper, we investigate the auditing problem in the general case of auditing arbitrary SUM queries over a real-valued attribute in terms of interval-based disclosure.

[30, 33, 34, 38] consider the extended auditing problem on select-project-join queries for general databases. [36] provides a good survey of all of the above methods.

The idea of solving a bunch of LPs by taking advantage of their similarity is not new. It was first introduced in operational research community, called sensitivity analysis or postoptimality analysis [19]. It was out of the concern that the parameter values of a LP and constraints are subject to change in reality. To avoid respectively solving many similar LPs, many methods specialized for different similarity situations were proposed. This idea was also successfully applied by computer science community to solve human-computer interface problems, called incremental LP, [4]. We are the first introducing this idea to database area and give the algorithms tailored for the auditing problem.

3. PRELIMINARIES

In this section, we briefly review the primal simplex method, the dual simplex method and the two-phase method, which provide preliminary knowledge for constructing our online

auditing algorithm.

3.1 Primal Simplex Method

A LP problem can be modeled as the following.

$$\begin{aligned} \min Z &= CX \\ \text{s.t.} \quad &\begin{cases} AX = b \\ X \geq 0 \end{cases} \end{aligned} \quad (1)$$

where C, X are vectors of n real values, b is a vector of m real values, and A is an $m \times n$ matrix ($m \leq n$) in the real domain. Without loss of generality (with linear transformation), assuming that A can be written as (B, N) , where B is an invertible $m \times m$ matrix,

$$\begin{aligned} (B, N)X &= b \\ BX_B + NX_N &= b \\ X_B + B^{-1}NX_N &= B^{-1}b \end{aligned}$$

If $B^{-1}b \geq 0$, then $(B^{-1}b, 0)$ is a feasible solution of the LP problem. In such case, we call $(B^{-1}b, 0)$ **feasible basic solution** and call B **feasible basis**. The variables corresponding to X_B are called **feasible basic variables**. After getting the feasible basic solution, we have

$$\begin{aligned} CX &= C_B X_B + C_N X_N \\ &= C_B(B^{-1}b - B^{-1}NX_N) + C_N X_N \\ &= C_B B^{-1}b - (C_B B^{-1}N - C_N)X_N \end{aligned}$$

If $C_B B^{-1}N - C_N \leq 0$, then $(B^{-1}b, 0)$ is the optimal solution, because X_N cannot be less than 0. The value of $C_B B^{-1}N - C_N$ is called **criterion value**.

In this paper, we focus on the simplex method, which is one of the most efficient methods for solving LP problems. Its basic idea is to find a feasible solution first and then keep swapping one basic variable and one non-basic variable such that the objective value is decreased constantly. When the objective value cannot be decreased anymore, the current feasible solution is the optimal solution. All the work is done in a table called **simplex tableau** by the procedure **pivoting**. A simplex tableau has the form shown in Figure 1, where all symbols correspond to the problem in (1). Recall that X_B denotes the basic feasible variables, X_N denotes the non-basic feasible variables, $C_B B^{-1}b$ is the objective function value, $C_B B^{-1}N - C_N$ is the criterion value, and $B^{-1}b, 0$ is the feasible basic solution. For the convenience of explanation, Figure 1 can be recoded as Figure 2. It shows δ denotes $C_B B^{-1}N - C_N$, a denotes $B^{-1}N$ and so on. Suppose we swap $x_r \in X_B$ and $x_k \in X_N$. The cell a_{rk} , corresponding to the row of the variable x_r and the column of the variable x_k in the tableau, is called a pivot. Pivoting is executed by dividing the row of x_r by a_{rk} and adding this row multiplied by some number to the other rows such that the cells at the column of x_k become 0. After that, swap the columns of x_r and x_k . The result will have exactly the same form as Figure 1. Regarding how to choose variables to swap, there are many rules. The most common one is as shown in Algorithm 1 [44].

		X_B	X_N
Z	$C_B B^{-1}b$	0	$C_B B^{-1}N - C_N$
X_B	$B^{-1}b$	I	$B^{-1}N$

Figure 1: Simplex Tableau

		X_B	X_N
Z	$C_B b'$	0	δ
X_B	b'	I	a

Figure 2: Simplex Tableau of Simple Denotation

Algorithm 1 Primal Simplex Algorithm

- 1: Find a feasible basis B , such that $B^{-1}b \geq 0$. Build the corresponding simplex tableau as in Figure 1.
- 2: $\delta_k = \max\{\delta_j | j \in N\}$
- 3: If $\delta_k \leq 0$, stop. The current feasible solution $X = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ is the optimal solution.
- 4: If $\delta_k > 0$ and the corresponding vector $a_k \leq 0$, the optimal solution is infinite.
- 5: Otherwise, choose a_{rk} such that $b'_r/a_{rk} = \min\{b'_i/a_{ik} | a_{ik} > 0\}$, as the pivot. Do pivoting, then go back to step 2.

3.2 Dual Simplex Method

A standard LP like equation (1) has a dual LP form as the following:

$$\begin{aligned} \max Z &= Wb \\ \text{s.t.} \quad &WA \leq C \end{aligned} \quad (2)$$

in which W is the vector of variables and the parameters b and C are same as in problem (1). There is an important property that the optimal objective value of problem (2) is equal to the optimal objective value of problem (1). Thus, by searching through feasible basic solutions of problem (2) such that the objective value is increased, we can finally get the optimal objective value of problem (1) as well. The basic idea of dual simplex method is to implement this searching process in the simplex tableau of problem (1), but for the problem (2). Its supporting theory is that a solution of problem (1), called **primal infeasible basic solution**, satisfying $X = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ and $C_B B^{-1}N - C_N \leq 0$, corresponds to a basic feasible solution of (2). Thus, dual simplex method is searching from one basic infeasible solution of problem (1) to another, such that the objective value is increased. Algorithm 2 gives the rule of how to traverse one primal infeasible basic solution to another.

The reason why we introduce dual simplex method is that, we will use it in Section 4 for updating mixed variables. Primal simplex algorithm and dual simplex algorithm are essentially the same. Normally, if finding a primal infeasible basic solution is much easier than finding a feasible basic solution, then dual simplex method is chosen.

3.3 Two Phase Method

The simplex methods like Algorithms 1 and 2 are very practical and efficient ways dealing with linear programming problems. Now the question is how to find the first basic solution to start these algorithms? Consider the primal simplex algorithm. For equations of n variables and rank k , the possible combinations of basic feasible variables are $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, which is a huge number. So instead of looking for a starting basic feasible solution among them, an extra optimization problem is constructed to get the starting feasible solution as the first phase and based on it, primal

Algorithm 2 Dual Simplex Algorithm

- 1: Find a basis B , such that $C_B B^{-1} N - C_N \leq 0$. Build the corresponding simplex tableau like Figure 2.
 - 2: If $b' = B^{-1}b \geq 0$, get the optimal solution $X = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ and the optimal objective value $Z = C_B B^{-1}b$; otherwise let $b'_r = \min\{b'_i | i = 1, \dots, m\}$.
 - 3: If $a_{rk} \geq 0$, there is no feasible solution; otherwise choose a_{rk} such that $\delta_k/a_{rk} = \min\{\delta_j/a_{rj} | a_{rj} < 0\}$ as the pivot. Do pivoting, then go back to step 1.
-

simplex method on the original problem is run as the second phase. The idea of the first phase is that for the problem (1), add the relaxing variables $X_a = \{x_{n+1}, \dots, x_{n+m}\}$ and put each of them in one constraint like (3). It is obvious that if there exists a feasible solution for the original problem (1), the optimal objective value of problem (3) is 0; in other words X_a are 0. The advantage of the constructed problem is that we have the feasible basic variables X_a right away. Thus by running the simplex method, we get a optimal solution like $(x'_1, x'_2, \dots, x'_n, 0, \dots, 0)$, where $(x'_1, x'_2, \dots, x'_n)$ is a feasible solution for the original problem (1). Subsequently, we can go to the second phase to solve the original problem by applying the primal simplex method. Note that besides this method, there are other similar methods such as big M [44] of finding the first basic feasible solution, which is as follows.

$$\begin{aligned} & \min \sum_{x_i \in X_a} x_i \\ & \text{s.t.} \begin{cases} AX + X_a = b \\ X, X_a \geq 0 \end{cases} \end{aligned} \quad (3)$$

4. MODELING ONLINE AUDITING

The goal of online auditing is to check if answering an arriving query would breach privacy, given that the users have complete knowledge of the previous queries. A sum query can be represented as an equation. For example, the query on the average salary of all faculty in the Biology department can be indicated as

$$x_1 + x_2 \dots + x_n = n * \text{Average}$$

where x_i denotes the salary of faculty i and n is the total number of faculty members. When users are also allowed to query the average salary by attributes such as sex, age and position, each query can improve the precision of the estimate of some faculty salary. Such unconstrained querying, may lead to disclosure of private information.

The basic task of online auditing is to continually check the lower and upper bounds of every variable to make sure that the bounds are not close enough to breach privacy. Privacy is breached if the difference between the inferred lower and upper bounds of a value is smaller than a predetermined threshold. Note that our research on online auditing is orthogonal to the definition of safe threshold. The system manager may assign different forms of safe threshold to different variables in auditing. For example, professor A, who does not care his salary information is released, can set the safe threshold for his or her salary data to be 0. After the safe thresholds are set for all variables, if the new query can reduce the bounds within the safe threshold for any variable, that query is denied. Otherwise, it is allowed.

The task of online auditing can be modeled as a series of growing linear programming problems. When the new query

$\sum_{x_i \in s_t} x_i = S_t$ arrives, where s_t is the variables set associated with t th query, the server should run the following linear programming problem to find the lower bound (indicated by $\min x_j$) and the upper bound (indicated by $\max x_j$). As $\max x_j$ can be equivalently solved by $\min -x_j$, for simplicity, in the following we discuss only $\min x_j$. The first $t-1$ constraints correspond to the previously released $t-1$ queries. $x_i \geq 0$ is the default constraint, to ensure positive values (for example, salary can never be negative). When the attribute can also take on negative values, this constraint can be removed. Other semantics such as $x_i \geq c$, can also be similarly modeled through simple linear transformations. Finally, we can get the linear programming model as equation (4)

$$\begin{aligned} & \min x_j \\ & \text{s.t.} \begin{cases} \sum_{x_i \in s_1} x_i = S_1 \\ \dots \\ \sum_{x_i \in s_{t-1}} x_i = S_{t-1} \\ \sum_{x_i \in s_t} x_i = S_t \\ x_i \geq 0, \forall x_i \in s_1 \cup \dots \cup s_{t-1} \cup s_t \end{cases} \end{aligned} \quad (4)$$

Linear programming problems are not difficult to solve. Simplex methods can efficiently deal with them in practice. However, for online auditing, a lot of linear programming problems have to be solved, essentially one for each query. Worse is that, the size of the problem increases for each subsequent query. Finally, a solution is required very quickly. Thus, a more efficient solution is clearly needed.

The key idea of our solution is to treat each successive linear programming problem simply as an updation of the previous result. This is possible since each successive linear programming problem only differs from the prior problem in two ways. First, the objective function is different – instead of $\min x_1$, we would like to solve $\min x_2$. Secondly, there is one more constraint based on the t th query.

Taking these two facts into consideration, we can divide the series of updation problems into two types, horizontal and vertical. These can be defined as follows:

DEFINITION 1 (THE HORIZONTAL UPDATION PROBLEM).
Given the same set of queries, as well as the bounds of one set of variables, how can the prior result be modified to get the bounds of other variables.

DEFINITION 2 (THE VERTICAL UPDATION PROBLEM).
Given the same set of variables, objective function, and bounds under the previous queries, how can the prior result be modified to get the bounds when a new query arrives.

In the following sections, we look at algorithms to perform both horizontal and vertical updation.

5. HORIZONTAL UPDATION

As discussed above, we have a horizontal updation problem when the objective function is modified, while the constraints are unchanged. Thus, we replace c by c' as in (5).

$$\begin{aligned} & \min z = C'X \\ & \text{s.t.} \begin{cases} AX = b \\ X \geq 0 \end{cases} \end{aligned} \quad (5)$$

Suppose that we have already solved the previous problem with the objective function CX and get the final tableau. According to the structure of tableau, we only need to update the criterion values to be $C'_B B^{-1}N - C'_N$ and Z to

		\bar{X}_B	\bar{X}_N
Z	$C'_B B^{-1}b$	0	$C'_B B^{-1}N - C'_N$
X_B	$B^{-1}b$	I	$B^{-1}N$

Figure 3: Updated Simplex Tableau for (5)

be $C'_B B^{-1}b$. This is sufficient since the optimal solution of the previous problem is a basic feasible solution of the new problem. In other words, we do not need to expend time on finding the first feasible solution for the new problem. Intuitively, we save approximately half of the time by omitting the first phase. If $C'_B B^{-1}N - C'_N \leq 0$, the current solution is in fact the optimal solution. Otherwise, we pivot according to the primal simplex methods. Thus, the horizontal update algorithm can be formally represented as follows: Next, we

Algorithm 3 Algorithm for Horizontal Updating

- 1: Update the criterion values $C_B B^{-1}N - C_N$ to be $C'_B B^{-1}N - C'_N$ and the objective value $C_B B^{-1}b$ to be $C'_B B^{-1}b$ respectively.
- 2: If $C'_B B^{-1}N - C'_N \leq 0$, get the optimal solution $X = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$ and the optimal objective value $C'_B B^{-1}b$. Otherwise, do pivoting as the Algorithm 1 until $C'_B B^{-1}N - C'_N \leq 0$.

illustrate the algorithm through a small example. Suppose that we have the following three queries shown in Figure 4. The figure also shows the simplex tableau result obtained by solving $\min x_1$. From this, we can see that the optimal solution is (1 4 3 0) and $\min x_1 = 1$.

$$\begin{cases} x_1 + x_2 = 5 \\ x_1 + x_3 = 4 \\ x_2 + x_3 + x_4 = 7 \end{cases}$$

		x_1	x_2	x_3	x_4
Z	1	0	0	0	-1/2
x_1	1	1	0	0	-1/2
x_2	4	0	1	0	1/2
x_3	3	0	0	1	1/2

Figure 4: the Simplex Tableau Result for $\min x_1$

Now assume that we need to solve $\min x_2$. Following Step 1 of Algorithm 3, we have

$$C'_B B^{-1}N - C'_N = (0 \ 1 \ 0) \times I \times \begin{pmatrix} -1/2 \\ 1/2 \\ 1/2 \end{pmatrix} - 0 = 1/2.$$

Since this is positive, it shows that the current feasible solution is not the optimal solution. Therefore, we update the simplex tableau as in Figure 5. According to the pivoting rule (Step 4 of Algorithm 1), we choose the cell labeled * as the pivot. After pivoting, the new simplex tableau is shown in Figure 6, where the criterion value is -1. Thus, the current solution (4 1 0 6), is optimal, and $\min x_2 = 1$.

For $\min x_3$, $C'_B B^{-1}N - C'_N = 1/2$. Following the same procedure as above, we will easily get $\min x_3 = 0$.

For $\min X_4$, $C'_B B^{-1}N - C'_N = 0$. Thus, the current feasible solution is optimal. Thus, we can immediately stop, and compute $\min x_4 = 0$.

6. VERTICAL UPDATION

Vertical updation occurs when the constraints are changed while the objective function remains the same. This will

		x_1	x_2	x_3	x_4
Z	4	0	0	0	1/2
x_1	1	1	0	0	-1/2
x_2	4	0	1	0	1/2
x_3	3	0	0	1	1/2*

Figure 5: the Simplex Tableau for $\min x_2$ (I)

		x_1	x_2	x_4	x_3
Z	1	0	0	0	-1
x_1	4	1	0	0	1
x_2	1	0	1	0	-1
x_4	6	0	0	1	2

Figure 6: the Simplex Tableau for $\min x_2$ (II)

happen, whenever a new query, $a'X = b'$ is posed. The updated optimization problem is depicted in (6).

$$\begin{aligned} \min Z = CX \\ \text{s.t.} \begin{cases} AX = b \\ a'X = b' \\ X \geq 0 \end{cases} \end{aligned} \tag{6}$$

This is a more challenging problem. Given the final tableau of 1 and a new query, we need to figure out how to modify the given result to get the new result. We address this problem in three cases, depending on whether the new query consists of all new variables, mixed variables, and all old variables.

6.1 All New Variables

If the new query consists of all new variables, releasing the query result does not affect the lower and upper bounds of the old variables at all. The upper bounds of the new variables are the query result and the lower bounds are 0 (assuming non-negative values). For example, assume the previous queries are

$$\begin{aligned} x_1 + x_2 = 5 \\ x_2 + x_3 = 7 \end{aligned}$$

The bounds for x_1, x_2 and x_3 are $[0,5]$, $[0,5]$ and $[2,7]$ respectively. If the new query is $x_4 + x_5 = 8$, it has no intersection with previous queries. Thus, the bounds of previous variables stay unchanged. For new variables, one can easily infer that x_4 and x_5 are in $[0,8]$. Thus, the updated bounds for all variables are $[0,5]$, $[0,5]$, $[2,7]$, $[0,8]$ and $[0,8]$.

STATEMENT 1. *If the new query consists of all new variables, the bounds of old variables stay unchanged. The lower bound for all new variables is 0 and the upper bound is the result of the query.*

Note that even if the new query contains all new variables, we still need to update the tableau by choosing one new variable as the basic variable and letting the rest of them be non-basic variables. For example, suppose the previous tableau is as in Figure 6 and the new query is $x_5 + x_6 = 10$. Then by choosing x_5 as the basic variable, the updated tableau is shown in Figure 7.

6.2 Mixed Variables

If the new query contains both old and new variables, it is easy to determine if the previous solution is still optimal, simply by updating the simplex tableau. From the previous simplex tableau result, we know that the coefficient matrix of previous queries as linear equations can be represented as in Figure 8. Supposing the new query is

		x_1	x_2	x_5	x_4	x_3	x_6
Z	1	0	0	0	0	-1	0
x_1	4	1	0	0	0	1	0
x_2	1	0	1	0	0	-1	0
x_4	6	0	0	1	0	2	0
x_5	10	0	0	0	1	0	1

Figure 7: Updated Tableau for the Query of All New Variables

$t_1x_1 + \dots + t_nx_n + x_{n+1} + \dots + x_{n+k} = t$, where x_1, \dots, x_n are old variables, t_1, \dots, t_n are either 0 or 1 and x_{n+1}, \dots, x_{n+k} are new variables. With the new query, the new coefficient matrix is shown in Figure 9, where T_B and T_N denote the coefficients corresponding to X_B and X_N respectively. Linear transformations can make it have the form of Figure 10. Now, $\{X_B, x_{n+1}\}$ can be basis variables for all of linear equations. However, do they correspond to a basic feasible solution? Perhaps even the optimal solution? For this, we must look at the previous criterion values $C_B B^{-1}N - C_N \leq 0$. After the new query arrives, the criterion values corresponding to the basic variables $\{X_B, x_{n+1}\}$ become $(C_B, 0) \begin{pmatrix} B^{-1}N \\ T'_N \end{pmatrix} - C_N = C_B B^{-1}N - C_N$. It shows the updated criterion values unchanged, still negative. Let $(X'_B, x_{n+1}, x_{n+2}, \dots, x_{n+k}, X'_N)$ be the solution using $\{X_B, x_{n+1}\}$ as the basic variables. If $t' \geq 0$, then this solution is an optimal basic feasible solution. If not, although this solution is not a basic feasible solution, it is a primal infeasible basic solution, which corresponds to a feasible solution of its dual problem. Thus, we can employ the dual simplex method to find the optimal solution. All procedures can be done from the previous simplex tableau.

X_B	X_N	Result
I	$B^{-1}N$	$B^{-1}b$

Figure 8: Previous Coefficient Matrix

X_B	x_{n+1}	$x_{n+2} \dots x_{n+k}$	X_N	Result
I	0	0	$B^{-1}N$	$B^{-1}b$
T_B	1	1...1	T_N	t

Figure 9: Current Coefficient Matrix (I)

Thus, we have the algorithm for vertical updates of mixed variables as Algorithm 4. We now give an example to illustrate our algorithm. Again, consider the example given in Section 5. Given the simplex tableau result of Figure 4, if the new query is $x_3 + x_5 = 5$, we need to solve a linear programming as follows:

$$\begin{aligned} \min Z &= x_1 \\ \text{s.t.} \quad &\begin{cases} x_1 + x_2 = 5 \\ x_1 + x_3 = 4 \\ x_2 + x_3 + x_4 = 7 \\ x_3 + x_5 = 5 \\ x_1, \dots, x_5 \geq 0 \end{cases} \end{aligned} \quad (7)$$

Following Step 1 of Algorithm 4, put the new query into the tableau and do the associated linear transformation. We get a new tableau as in Figure 12. Because $t' = 2$, the current solution (1 4 3 0 2) is the optimal solution and $\min x_1 = 1$.

If the new query is $x_3 + x_5 = 2$, after step 1, the new tableau is shown in Figure 13. Because $t' = -1$, the current solution is not the optimal solution, but a primal feasible basic solution. Thus, employ Algorithm 2, choose the cell, labeled *,

X_B	x_{n+1}	$x_{n+2} \dots x_{n+k}$	X_N	Result
I	0	0	$B^{-1}N$	$B^{-1}b$
0	1	1...1	T'_N	t'

Figure 10: Current Coefficient Matrix (II)

Algorithm 4 Algorithm for Vertical Updating of Mixed Variables

- 1: Given the new query $t_1x_1 + \dots + t_nx_n + x_{n+1} + \dots + x_{n+k} = t$, update the previous simplex tableau to get the new tableau in the form of Figure 11.
- 2: If $t' \geq 0$, terminate with the optimal solution $((B^{-1}b)', t', 0 \dots 0)$ and the optimal objective value $c_B B^{-1}b$.
- 3: If not, employ Algorithm 2 on the tableau of Figure 11 to get the optimal solution.

as the pivot. After pivoting, the new tableau is Figure 14 It satisfies the requirement of a optimal solution. Hence, the optimal solution is (2 3 2 2 0) and $\min x_1 = 2$.

6.3 All Old Variables

If the new query contains all old variables (i.e., seen in the prior queries), then there are two cases. One possibility is that releasing the new query does not affect the known bounds of any of the variables. Thus, it may simply be that the new query is simply a linear combination of previous queries (i.e., can easily be derived from the old queries). For example, given the old queries, $x_1 + x_2 = 5$, $x_3 + x_4 = 6$, the new query is $x_1 + x_2 + x_3 + x_4 = 11$. In this case, there is no problem. The other possibility is that releasing the new query tightens the known bounds on some or all of the variables. Hence, the problem that we now face is that given a new query, how do we know whether it tightens the known bounds or not. If it does, what are the new bounds?

The first case can be easily verified simply by putting the parameters of the new query at the bottom of the previous simplex tableau and executing Gaussian elimination. If all parameters become zeroes, the new query is duplicate and the bounds for all variables do not change at all. Otherwise, further work needs to be done. Note that, since in a simplex tableau, the coefficients A are represented by $(I, B^{-1}N)$, it only takes $O(mn)$ operations to implement Gaussian eliminations on the new query. In terms of computation efficiency, it is same as audit expert system in [10]. However, it is better than audit expert system in terms of storage efficiency, because in the real practice of simplex methods, only $B^{-1}N$ and the indices of variables corresponding to I are stored.

In the second case, we need to do more work. Sometimes, even if the query is not duplicate, the optimal value does not change, although it happens rarely. So first we put the previous optimal solution in the new query. If the solution matches the new query, the previous optimal solution is the current optimal solution. Otherwise, similar to earlier, we need to find a basic feasible solution or a primal infeasible basic solution for the new optimization problem with the new query. However, differing from the previous problems, there is no direct way to find it. Therefore, we have to resort back to the two-phase method – first, find a feasible solution; then starting with that solution, run the simplex

		X_B	x_{n+1}	$x_{n+2} \dots x_{n+k}$	X_N
Z	$C_B B^{-1} b$	0	0	0...0	$C_B B^{-1} N - C_N$
X_B	$B^{-1} b$	1	0	0...0	$B^{-1} N$
x_{n+1}	t'	0	1	1...1	T'_N

Figure 11: Simplex Tableau for Mixed Variables

		x_1	x_2	x_3	x_5	x_4
Z	1	0	0	0	0	-1/2
x_1	1	1	0	0	0	-1/2
x_2	4	0	1	0	0	1/2
x_3	3	0	0	1	0	1/2
x_5	2	0	0	0	1	-1/2

Figure 12: Simplex Tableau I

method to get the optimal solution. However, due to the special structure of our problem, we only need to include one slack variable for the constructed optimization problem for the first phase, instead of m variables (m is the rank of A) as in equation 3. Thus, we still spend less time than standard simplex method.

Given the new query is $T_B X_B + T_N X_N = t$, simple linear transformation with the previous queries can make it to be $T'_N X_N = t'$ such that $t' \geq 0$. Then add a relaxing variable x_{add} to make it to be $T'_N X_N + x_{add} = t'$. To get a feasible solution of the updated problem, we need to solve an extra optimization problem:

$$\begin{aligned} \min Z' &= x_{add} \\ \text{s.t. } \begin{cases} AX + 0x_{add} = b \\ T'_N X_N + x_{add} = t' \\ X \geq 0 \end{cases} \end{aligned} \quad (8)$$

The reason why we create such an extra problem is that now we can have a feasible solution, $(B^{-1}b, t', 0)$, directly. In addition, the optimal solution of the extra problem is a feasible solution of its original problem.

To solve the extra problem, we insert this new query into the previous simplex tableau, let $\{X_B, x_{add}\}$ to be the basic variables and apply the primal simplex method to find the optimal solution. The starting tableau is as in Figure 15.

When getting the optimal solution, to solve the original problem, we delete the column and row corresponding to the variable x_{add} from the tableau. Suppose the current basic variables are $X_{B'}$ and the remaining variables are $X_{N'}$. Then the new tableau is shown in Figure 16.

Putting it all together, the formal algorithm for vertical update of all old variables is shown in Algorithm 5.

Next, we will give an example to illustrate our algorithm. Suppose the previous queries and the objective function are

$$\begin{aligned} \min x_1 \\ \begin{cases} x_1 + x_2 = 5 \\ x_2 + x_3 + x_4 = 4 \\ x_1 + x_3 + x_5 = 7 \end{cases} \end{aligned} \quad (9)$$

Its optimal solution is (1 4 0 0 6) and its simplex tableau is Figure 17.

If the new query is $x_2 + x_5 = 10$, the previous optimal so-

		x_1	x_2	x_3	x_5	x_4
Z	1	0	0	0	0	-1/2
x_1	1	1	0	0	0	-1/2
x_2	4	0	1	0	0	1/2
x_3	3	0	0	1	0	1/2
x_5	-1	0	0	0	1	-1/2*

Figure 13: Simplex Tableau II

		x_1	x_2	x_3	x_4	x_5
Z	2	0	0	0	0	-1
x_1	2	1	0	0	0	-1
x_2	3	0	1	0	0	1
x_3	2	0	0	1	0	1
x_4	2	0	0	0	1	-2

Figure 14: Simplex Tableau III

lution satisfies the equation. Thus, $\min x_1$ is still 1. If the new query is $x_2 + x_5 = 4$, the optimal solution does not satisfy. Following the Algorithm 5, we need to create an extra problem by adding one relaxing variable x_6 . Thus the starting simplex tableau for the extra problem is shown in Figure 18. According to the pivoting rule of the primal simplex algorithm, the cell labeled * is chosen as the pivot. After pivoting, the simplex tableau is shown in Figure 19. It shows that the optimal solution for the extra problem is (3 2 2 0 2 0). This implies that (3 2 2 0 2) is a feasible solution for the original problem. Based on the step 3 of the algorithm 5, delete the column corresponding to the x_6 and recalculate the data on the first row, which are objective value and criterion value, to get Figure 20. As the criterion value is $-1/3$, which means the current solution is the optimal solution, we get the optimal solution for the original problem (3 2 2 0 2), without further pivoting.

7. ONLINE AUDITING ALGORITHM

From the previous two sections, we know how to perform horizontal updates and vertical updates. In this section, we will integrate both to construct an efficient online auditing algorithm, which we call the online updating algorithm, shown in Algorithm6.

The basic idea of this algorithm is that we choose one old variable as a default variable and keep its corresponding simplex tableaus for min and max as default simplex tableaus.

Algorithm 5 Algorithm for Vertical Updating of all Old Variables

- 1: Given the new query $T_B X_B + T_N X_N = t$, do Gaussian elimination on it. If all parameters become zeroes, terminate. Otherwise, go to the next step.
 - 2: Put the previous optimal solution in the new query. If it satisfies the equation, terminate since the previous optimal solution is also currently optimal. Otherwise, go to the next step.
 - 3: Create an extra problem of the objective function $\min x_{add}$, where x_{add} is the relaxing variable. Construct the corresponding simplex tableau with $\{X_B, x_{add}\}$ as the basic variables. Then run Algorithm 1.
 - 4: Delete the column and row corresponding to the variable x_{add} and modify other cells to get a starting simplex tableau as shown in Figure 16. Then run Algorithm 1 again.
-

		X_B	x_{add}	X_N
Z'	T'_N	0	0	T'_N
X_B	$B^{-1}b$	I	0	$B^{-1}N$
x_{add}	t'	0	1	T'_N

Figure 15: Simplex Tableau for the Extra Problem

		$X_{B'}$	$X_{N'}$
Z	$C_{B'}B'^{-1}b$	0	$C_{B'}B'^{-1}N - C_{N'}$
$X'_{B'}$	$B'^{-1}b$	I	$B'^{-1}N'$

Figure 16: Vertical Updating of All Old Variables

It also keeps a record of all old variables that are involved in the old queries (i.e., the queries that have been previously answered by the auditing algorithm). When a new query arrives, the algorithm checks whether it contains all new variables, mixed variables, or all old variables and chooses the corresponding vertical update method accordingly. By using the vertical updating method, the algorithm obtains the minimal and maximal values of the default variable and the updated simplex tableaus as well. Then, starting from the default variable, the algorithm uses the horizontal update method to solve the minimal and maximal values for every other variable that has appeared in either the old queries or the new query. While doing so, if the algorithm detects that the difference between the minimum and maximum bounds of any variable becomes smaller than the predefined threshold, the new query is denied. Otherwise, the algorithm releases the answer to the query and chooses the last variable processed as the default variable and the corresponding tableaus as the default tableaus.

In the implementation, we use a binary vector α of length $|X|$ to record the old variables (i.e., the variables in X that have been involved in the old queries) for each user. A binary value 1 in the vector indicates the corresponding variable is involved in the old queries. Similarly, we use another binary vector β of length $|X|$ to represent the new query asked by the user. We use the following method to determine which vertical auditing method is used in step 2 of our online updating algorithm (Algorithm 6). If $(\alpha \text{ AND } \beta) = 0$, then the new query contains all new variables; else if $(\alpha \text{ AND } \beta) = \beta$, then the new query contains all old variables; else, the new query contains mixed variables. At the end of step 3 in the online updating algorithm, α is updated as $(\alpha \text{ OR } \beta)$.

Intuitively, our online auditing algorithm is superior to the existing approach of applying standard linear programming in auditing each user’s queries due to several reasons. First, except for the case in which a new query contains all old variables, our algorithm avoids the phase of finding the first basic feasible solution. Even in the case that a new query contains all old variables, the phase of finding the first basic feasible solution is performed *for the default variable only* in vertical updates (see Algorithm 5), while it is avoided *for all other variables* which are involved in the old queries and the new query in horizontal updates. The improvement of our online auditing algorithm over the standard linear programming is significant since the phase of finding the first basic feasible solution can be as expensive as the phase of finding the optimal solution. Furthermore, even in vertical updating

		x_1	x_2	x_5	x_3	x_4
Z	1	0	0	0	-1	-1
x_1	1	1	0	0	-1	-1
x_2	4	0	1	0	1	1
x_5	6	0	0	1	2	1

Figure 17: Simplex Tableau for (9)

		x_1	x_2	x_3	x_6	x_4	x_5
Z	6	0	0	0	0	3	2
x_1	1	1	0	0	0	-1	-1
x_2	4	0	1	0	0	1	1
x_3	6	0	0	1	0	2	1
x_6	6	0	0	0	1	3*	2

Figure 18: Step 1 for Updating on (9)

for all old variables, the constructed optimization problem in first phase is simpler than standard simplex methods.

Second, in the phase of finding an optimal solution, our online auditing algorithm can quickly determine whether a newly constructed solution is optimal (see step 2 in Algorithm 3 and in Algorithm 4) or whether a new query would improve the existing optimal solution (see step 1 in Algorithm 5) before it employs the traditional primal simplex method (Algorithm 1) or dual simplex method (algorithm 2) to get the optimal solution. In most cases, as we observe in our numerical experiments (see the next section), our online auditing algorithm can get the optimal solution without incurring the cost of calling the traditional simplex method.

Third, our online auditing algorithm requires recording the binary vector α of length $|X|$ as well as one default variable and two default simplex tableaus between auditing different queries. In comparison, standard linear programming requires that the set of all old queries is recorded in auditing. Our online auditing algorithm is more storage-efficient since a simplex tableau usually requires much less storage than the set of old queries does. A simplex tableau takes at most $O(|X|^2)$ bits to store, while the set of previous queries may require as much as $O(2^{|X|}|X|)$ bits in the worst case. In this sense, it is also better than any solution (not based on linear programming), due to the efficacy of storage of past history.

8. EXPERIMENTAL VALIDATION

To assess the relative performance of our algorithm versus standard linear programming in auditing, we performed several experiments on an IBM T43 laptop with a CPU clock rate of 1.86 GHz, 512 MB of RAM, and running Microsoft Windows XP Professional 2002. We first describe the experimental settings, including the synthetic data sets used in our performance evaluation. We then show the performance results over a range of data characteristics.

8.1 Experimental Settings

We ran our algorithm and the standard linear programming for auditing a set Q of queries over a set X of real values, each of which is randomly selected from $\{1, 2, \dots, |X|\}$. Since our algorithm is more efficient when a new query is denied rather than answered in auditing (while the standard linear programming remains the same in the two cases), we tested essentially the worst case for our algorithm (time complexity

		x_1	x_2	x_5	x_3	x_6	x_4
Z	0	0	0	0	0	0	-1
x_1	3	1	0	0	0	1/3	-1/3
x_2	2	0	1	0	0	-1/3	1/3
x_5	2	0	0	1	0	-2/3	-1/3
x_3	2	0	0	0	1	1/3	2/3

Figure 19: Step2 for Updating on (9)

		x_1	x_2	x_5	x_3	x_4
Z	3	0	0	0	0	-1/3
x_1	3	1	0	0	0	-1/3
x_2	2	0	1	0	0	1/3
x_5	2	0	0	1	0	-1/3
x_3	2	0	0	0	1	2/3

Figure 20: Step 3 for Updating on (9)

wise) in which all queries in Q are not denied in auditing. For this purpose, we set the threshold to be zero. In general, our algorithm will perform significantly better as queries can be quickly denied.

We generated the set Q of queries in a random manner. The size of each query is determined by a random variable of Poisson distribution with mean μ . If the generated number is greater than $|X|$, we generate it again till the number is not greater than $|X|$. Note that in this way, the generated data is not real poisson distribution, but of the density function $Prob(x = k) = \frac{\lambda^k e^{-\lambda}}{k!} / \sum_{i=1}^{|X|} \frac{\lambda^i e^{-\lambda}}{i!}$, $1 \leq k \leq |X|$. To model the phenomenon that queries often involve common variables, some fraction of variables in subsequent queries are chosen from the previous queries generated. We use an exponentially distributed random variable with mean equal to the correlation level γ to decide this fraction for each query. Thus, if we assume the previous query is q_i , the size of new query is s , and fraction level is r , then the new query is generated by randomly choosing $|q_i| * r$ values from q_i and the rest $s - |q_i| * r$ values from $X \setminus q_i$. In case of $s < |q_i| * r$ (i.e. the size of new query is smaller than the fraction of the previous query), we choose s values randomly from q_i . In case of $s - |q_i| * r > |X| - |q_i|$ (i.e., there is no enough data from $X \setminus q_i$ for the new query), we choose all data from $X \setminus q_i$ to form the new query. However, these two cases happen rarely in our experiments.

parameter	meaning	default value
$ X $	number of values (variables)	100
$ Q $	number of queries	30
γ	correlation level	0.5
μ	average size of queries	10

Table 1: Parameters and default values

The four parameters, $|X|$, $|A|$, μ and γ that are used to generate data and queries in our experiment is summarized in Table 1. Our experiment is executed by varying one of four parameter while keeping the rest as the default values, which are shown in Table 1. Since the final result (allow or deny) is the same, for each set of data that is generated, we only compare the execution time of our online updating algorithm with the standard linear programming.

Algorithm 6 Online Updating Algorithm

- 1: Take the first query and get each variable's minimum and maximum (which are obviously 0 and the result of the first query, respectively). If the difference between these values is no greater than a threshold, deny the query; otherwise, answer the query, choose any one of the variables that are involved in the first query as the default variable, and build corresponding simplex tableaus as the default tableaus.
- 2: When a new query arrives, employ the appropriate vertical update algorithm to obtain the minimum and maximum of the default variable and the updated simplex tableaus as well. If the difference between the minimum and maximum is smaller than the predefined threshold, deny the query.
- 3: Starting from the default variable, employ the horizontal update algorithm to obtain the minimum and maximum values for every other variable that is involved in either the old queries or the new query. Again, if the difference between the minimum and maximum values for any variable is smaller than the predefined threshold, deny the query. Otherwise, answer the query, update the last variable processed as the default variable, update the corresponding tableaus as the default tableaus, and go to Step 2.

8.2 Experimental Results

Figure 21 shows our experimental results by varying the number of queries issued by a user from 5 to 50. It is observed that the execution time of standard linear programming increases in linear proportion to the number of queries, while the execution time of our online updating algorithm is much less sensitive to the change of the number of queries. In the default case where $|Q| = 30$, our online updating algorithm is about 30 times faster than standard linear programming. The more queries are involved in auditing, the more efficient is our algorithm as compared to the standard linear programming. This feature makes our algorithm more prominent for auditing a large number of cumulative queries for each frequent user in the long run.

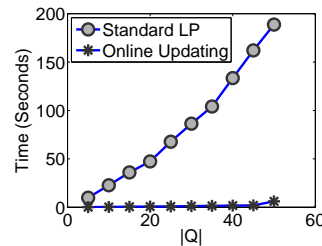


Figure 21: Varying $|Q|$

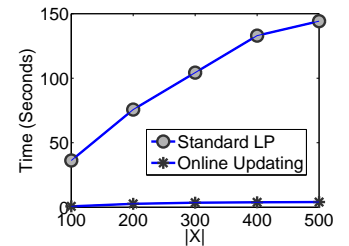


Figure 22: Varying $|X|$

Figure 22 compares our algorithm with the standard linear programming by scaling up the number of values in the protected attribute from 100 to 500. It is observed that our algorithm is affected little, while the standard linear programming increases evidently with $|X|$. The reason is that the more values are in the protected attribute, the more variables are involved in the queries; thus, the standard linear programming takes more time to solve the minimum and maximum for these variables.

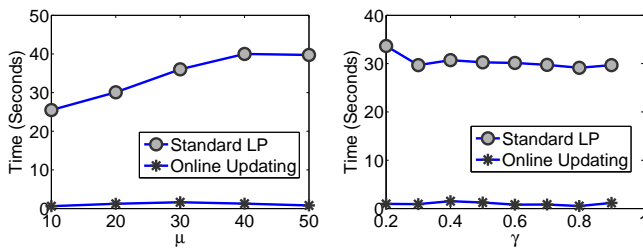


Figure 23: Varying μ

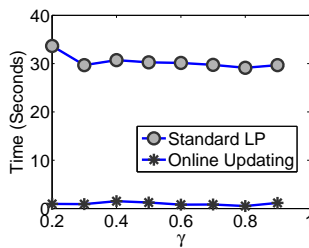


Figure 24: Varying γ

In Figure 24, the correlation level is varied from 0.2 to 0.9. The larger the correlation level, the more overlap between two consecutive queries. However, the total number of variables involved in 30 queries ($|Q| = 30$ in the default case) does not change significantly with the correlation level. This is mainly due to the random choice of each query besides the overlapping portion taken from the previous query. As a result, the standard linear program is not sensitive to the change of the correlation level. Our algorithm is around 30 times faster than the standard linear programming consistently in this set of experiments. This is due to the number of queries, and is consistent with our prior result. Thus, the conclusion is that overlap factor does not affect running time of either algorithm.

The last set of experiments scales up the average size μ of each query from 10 to 50, as shown in Figure 23. The standard linear programming increase with the average size of each query because the larger each query is, the more variables are involved in auditing; thus, it takes more time for the standard linear programming to solve the minimum and maximum for these variables. Notice that this increase is not linear, since the number of variables involved in a fixed number of queries (i.e., $|Q| = 30$ in Figure 23) may be saturated if the average size of each query is large enough ($\mu \geq 40$ in Figure 23). In comparison, our auditing algorithm takes the advantage of vertical updating and horizontal updating almost equally effectively in any cases. In other words, our algorithm is more efficient for auditing larger queries as compared to the standard linear programming.

We note that the experiments were performed exactly following our proposed algorithms. In fact, there are some methods which can be used to further reduce the computing time without increasing the storage space. All state-of-art algorithms (e.g., the revised simplex method) that improve the simplex method can be directly incorporated in our algorithm. We also note that there exist multiple stopping rules for simplex methods. We chose to use the widely used rule in our experiments in step 4 of the Algorithm 1 and step 3 of the Algorithm 2 [44]. This rule has a potential problem in theory: it may cause recycling in searching for a basic feasible solution. Certain other stopping rules, such as the Bland rule and the dictionary order rule [44], can be used to avoid the potential recycling problem. Nonetheless, the potential recycling problem happens very rarely in practice. We did not encounter any such problem in our experiments.

9. CONCLUSIONS

Online auditing is a critical component in database systems for protecting privacy information stored in databases with-

out adding noises or over-restricting user queries. Traditional auditing approach exploits linear programming for monitoring the exact bounds of protected values that each user can obtain from a set of queries. The efficiency of this approach is restricted by applying standard linear programming repetitively for auditing all involved values with respect to the set of cumulative queries each time a new query is checked. This paper shows that the efficiency of online auditing can be improved by opening the black box of linear programming and discovering short cuts between auditing different values. Numerical experiments show that our algorithm is significantly more efficient than the standard linear programming, especially more efficient for auditing a larger number of larger queries over more protected values. In the future, we plan to investigate the online auditing problem for mixed queries, including not only SUM queries, but also min and max queries. For mixed queries, Chin investigated the auditing problem with respect to the exact-value disclosure [7]. It is interesting to extend our study to auditing mixed queries with respect to the interval-based disclosure.

10. REFERENCES

- [1] ADAM, N. R., AND WORTMANN, J. C. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys* 21, 4 (1989), 515–556.
- [2] AGRAWAL, D., AND AGGARWAL, C. C. On the design and quantification of privacy preserving data mining algorithms. In *PODS* (2001).
- [3] AGRAWAL, R., AND SRIKANT, R. Privacy-preserving data mining. In *SIGMOD Conference* (2000), pp. 439–450.
- [4] BADROS, G. J., BORNING, A., AND STUCKEY, P. J. The cassowary linear arithmetic constraint solving algorithm. *ACM Trans. Comput.-Hum. Interact.* 8, 4 (2001), 267–306.
- [5] BECK, L. L. A security mechanism for statistical databases. *ACM Trans. Database Syst.* 5, 3 (1980), 316–338.
- [6] CHEN, K., AND LIU, L. Privacy preserving data classification with rotation perturbation. In *ICDM* (2005), pp. 589–592.
- [7] CHIN, F. Y. L. Security problems on inference control for sum, max, and min queries. *J. ACM* 33, 3 (1986), 451–464.
- [8] CHIN, F. Y. L., KOSSOWSKI, P., AND LOH, S. C. Efficient inference control for range sum queries. *Theor. Comput. Sci.* 32 (1984), 77–86.
- [9] CHIN, F. Y. L., AND ÖZSOYOĞLU, G. Statistical database design. *ACM Trans. Database Syst.* 6, 1 (1981), 113–139.
- [10] CHIN, F. Y. L., AND ÖZSOYOĞLU, G. Auditing and inference control in statistical databases. *IEEE Trans. Software Eng.* 8, 6 (1982), 574–582.
- [11] COX, L. H. Suppression methodology and statistical disclosure control. *Journal of American Statistical Association* 75 (1980), 377–385.
- [12] COX, L. H. Network models for complementary cell suppression. *Journal of the American Statistical Association* 90 (1995), 1453–1462.
- [13] DENNING, D. E., AND SCHLORER, J. Inference controls for statistical databases. *IEEE Computer* 16,

- 7 (1983), 69–82.
- [14] DOBKIN, D. P., JONES, A. K., AND LIPTON, R. J. Secure databases: Protection against user influence. *ACM Trans. Database Syst.* 4, 1 (1979), 97–106.
- [15] DOMINGO-FERRER, J., AND MATEO-SANZ, J. M. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Trans. Knowl. Data Eng.* 14, 1 (2002), 189–201.
- [16] FARKAS, C., AND JAJODIA, S. The inference problem: A survey. *SIGKDD Explorations* 4, 2 (2002), 6–11.
- [17] FISCHETTI, M., AND SALAZAR, J. J. Solving the cell suppression problem on tabular data with linear constraints. *Management Sciences* 47 (2000), 1008–1026.
- [18] FISCHETTI, M., AND SALAZAR, J. J. Partial cell suppression: a new methodology for statistical disclosure control. *Statistics and Computing* 13 (2003), 13–21.
- [19] GOLDMAN, A. J., AND TUCKER, A. W. Linear programming. *Princeton University Press* (1956).
- [20] HUANG, Z., DU, W., AND CHEN, B. Deriving private information from randomized data. In *SIGMOD Conference* (2005), pp. 37–48.
- [21] IYENGAR, V. S. Transforming data to satisfy privacy constraints. In *KDD* (2002), pp. 279–288.
- [22] KARGUPTA, H., DATTA, S., WANG, Q., AND SIVAKUMAR, K. On the privacy preserving properties of random data perturbation techniques. In *ICDM* (2003), pp. 99–106.
- [23] KENTHAPADI, K., MISHRA, N., AND NISSIM, K. Simulatable auditing. In *PODS* (2005), pp. 118–127.
- [24] KLEINBERG, J. M., PAPADIMITRIOU, C. H., AND RAGHAVAN, P. Auditing boolean attributes. *J. Comput. Syst. Sci.* 66, 1 (2003), 244–253.
- [25] LI, N., LI, T., AND VENKATASUBRAMANIAN, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE* (2007), pp. 106–115.
- [26] LI, Y., LU, H., AND DENG, R. H. Practical inference control for data cubes (extended abstract). In *Proceedings of the IEEE Symposium on Security and Privacy* (2006).
- [27] LI, Y., WANG, L., AND JAJODIA, S. Preventing interval-based inference by random data perturbation. In *Privacy Enhancing Technologies* (2002), pp. 160–170.
- [28] LI, Y., WANG, L., WANG, X. S., AND JAJODIA, S. Auditing interval-based inference. In *CAiSE* (2002), pp. 553–567.
- [29] LIU, K., KARGUPTA, H., AND RYAN, J. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. Knowl. Data Eng.* 18, 1 (2006), 92–106.
- [30] MACHANAVAJJHALA, A., AND GEHRKE, J. On the efficiency of checking perfect privacy. In *In Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 2006*.
- [31] MACHANAVAJJHALA, A., GEHRKE, J., KIFER, D., AND VENKITASUBRAMANIAM, M. l-diversity: Privacy beyond k-anonymity. In *ICDE* (2006), p. 24.
- [32] MALVESTUTO, F. M., MEZZINI, M., AND MOSCARINI, M. Auditing sum-queries to make a statistical database secure. In *ACM Transactions on Information and System Security (TISSEC)* (2006), vol. 9, pp. 31–60.
- [33] MIKLAU, G., AND SUCIU, D. A formal analysis of information disclosure in data exchange. *Journal of Computer and System Sciences.* (2006).
- [34] MOTWANI, R., NABAR, S. U., AND THOMAS, D. Auditing sql queries. In *Proceedings of the 21th International Conference on Data Engineering* (2008).
- [35] MURALIDHAR, K., AND SARATHY, R. A general additive data perturbation method for database security. *Management Sciences* 45 (2002), 1399–1415.
- [36] NABAR, S. U., KENTHAPADI, K., MISHRA, N., AND MOTWANI, R. A survey of query auditing techniques for data privacy. In *Privacy-Preserving Data Mining: Models and Algorithms, Springer, 2008 (to appear)*.
- [37] NABAR, S. U., MARTHI, B., KENTHAPADI, K., AND MISHRA, N. Towards robustness in query auditing. In *Proceedings of the 32nd Very Large Data Bases* (2006).
- [38] R. AGRAWAL, R. BAYARDO, C. F. J. K. R. R., AND SRIKANT, R. Auditing compliance with a hippocratic database. In *In Proceedings of the International Conference on Very Large Databases (VLDB), 2004*.
- [39] SAMARATI, P., AND SWEENEY, L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, 1998.
- [40] SCHLÖRER, J. Security of statistical databases: Multidimensional transformation. *ACM Trans. Database Syst.* 6, 1 (1981), 95–112.
- [41] SCHLÖRER, J. Information loss in partitioned statistical databases. *Comput. J.* 26, 3 (1983), 218–223.
- [42] SWEENEY, L. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10, 5 (2002), 571–588.
- [43] TRAUB, J. F., YEMINI, Y., AND WOZNIAKOWSKI, H. The statistical security of a statistical database. *ACM Trans. Database Syst.* 9, 4 (1984), 672–679.
- [44] VANDERBEI, R. J. *Linear Programming: Foundations and Extensions (3rd edition)*. Springer, 2008.
- [45] WANG, K., YU, P. S., AND CHAKRABORTY, S. Bottom-up generalization: A data mining solution to privacy protection. In *ICDM* (2004), pp. 249–256.
- [46] WANG, L., JAJODIA, S., AND WIJESKERA, D. Securing olap data cubes against privacy breaches. In *IEEE Symposium on Security and Privacy* (2004), pp. 161–175.
- [47] WANG, L., LI, Y., WIJESKERA, D., AND JAJODIA, S. Precisely answering multi-dimensional range queries without privacy breaches. In *ESORICS* (2003), pp. 100–115.
- [48] WILLENBORG, L., AND DE WAAL, T. *Statistical Disclosure Control in Practice*. Springer Verlag, 1996.