

Towards Robustness in Query Auditing

Shubha U. Nabar¹

Bhaskara Marthi²

Krishnaram Kenthapadi¹

Nina Mishra³

Rajeev Motwani¹

¹{sunabar, kngk, rajeev}@cs.stanford.edu ²bhaskara@cs.berkeley.edu ³nmishra@cs.virginia.edu

ABSTRACT

We consider the *online query auditing problem* for statistical databases. Given a stream of aggregate queries posed over sensitive data, when should queries be denied in order to protect the privacy of individuals? We construct efficient auditors for `max` queries and bags of `max` and `min` queries in both the *partial* and *full disclosure* settings. Our algorithm for the partial disclosure setting involves a novel application of probabilistic inference techniques that may be of independent interest. We also study for the first time, a particular dimension of the *utility* of an auditing scheme and obtain initial results for the utility of `sum` auditing when guarding against full disclosure. The result is positive for large databases, indicating that answers to queries will not be riddled with denials.

1. INTRODUCTION

A statistical database (SDB) is a database that allows its users to retrieve only aggregate statistics (such as mean or count) over subsets of its data. An example is the database maintained by the U.S. Census Bureau. Such databases generally contain sensitive information about individuals and there is often a need to enable the computation of useful statistics from such data while protecting the privacy of individuals. Consider for example, a company database containing salaries of its employees or a hospital database containing medical records of its patients. A statistician may want to determine the average salary of all the female employees in a company or the number of occurrences of a disease in a particular county. He cannot, however, be allowed to glean the salary of any one female employee in particular or the disease of any one individual.

Common approaches to tackling this problem include perturbing either the data itself [26, 3, 15, 4, 7, 23] or the results supplied to the user [10, 14, 6, 13, 12]. After much discussion with statisticians [18] however, we found that they are averse to potential biases introduced by adding noise. One

commonly stated reason is that the data collection process is already prone to biases and imperfections due to factors such as too few respondents, the cost of gathering data, and inaccurate answers provided by respondents. Since important decisions are made based on this data, they prefer to receive answers without additional noise. It is in this context that query restriction techniques become relevant.

In this paper, we consider an SDB with one sensitive attribute and several public attributes. A user can specify a subset of records in the database via predicates on the public attribute values and aggregates are taken over the corresponding sensitive attribute values. In the case of the company database an example query would be

```
SELECT sum(Salary)
FROM CompanyTable
WHERE ZipCode = 94305
```

Consider an SDB containing n records. Let $X = \{x_1, \dots, x_n\}$ be the multiset of sensitive attribute values in the database. Each x_i is the sensitive value of the i th individual. In our scenario the x_i s are taken to be real-valued from a bounded or unbounded range. A statistical query $q = (Q, f)$ specifies a subset of the records $Q \subseteq \{1, \dots, n\}$ and a function f (such as `sum`, `max` or `median`). The result, $f(Q)$, is f applied to the subset $\{x_i \mid i \in Q\}$. We call Q the *query set* of q .

The *online query auditing problem* is defined as follows: Given a sequence of queries q_1, \dots, q_{t-1} that have already been posed, answers a_1, \dots, a_{t-1} to these queries that have already been supplied and a new query q_t , should q_t be answered or should it be denied to prevent a privacy breach. Here each of the previous answers a_i , is either an exact answer, $f_i(Q_i)$, or a denial. In this paper, we attempt to enhance the robustness of the notion of online auditing by exploring algorithms for auditing new kinds of queries under different notions of compromise and examining the heretofore unexamined dimension of *utility*.

The auditing problem may be viewed as a game between the auditor and an attacker. The auditor monitors queries posed by the attacker and denies queries whenever answers to these and previous queries may be stitched together to cause compromise. It is the policy of the system as set by the DBA (database administrator) that determines the criterion for compromise. In most previous work, compromise corresponds to the notion of *full disclosure* and occurs when

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 VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '06, September 12-15, 2006, Seoul, Korea.

Copyright 2006 VLDB Endowment, ACM 1-59593-385-9/06/09

the private data of any individual can be exactly determined. We call this *classical compromise*. In [21], the authors introduce *probabilistic compromise* for bounded range data where a significant change in the attacker’s confidence about the range of a data point constitutes a privacy breach. This corresponds to the notion of *partial disclosure*. The question of auditing many different kinds of queries under probabilistic compromise remains wide open and in this paper, we introduce new algorithms for auditing **max** queries and bags of **max** and **min** queries under this definition. In the case of classical compromise, algorithms are known for auditing **sum**, **avg**, **min** and **max** queries separately, but combinations of these queries are hard to audit. We present an auditor for bags of **max** and **min** queries in the case of classical compromise. The authors in [21] show how even such simple types of queries can be used against a naive auditor to reveal considerable amounts of private data and building a robust auditor for such queries is thus important.

A naive solution to the general online auditing problem is to deny all queries. This is not a very satisfying solution as it does not provide much utility to the user. To the best of our knowledge, no previous work has attempted to quantify the utility of an auditing scheme. In this paper, we consider a particular dimension of utility and obtain initial results for the auditing algorithm for **sum** queries described in [9, 21]. The result pertains to classical compromise and is a positive result for large databases.

For smaller databases, users may be able to derive more utility because databases are frequently updated. Intuitively, this is because past information gathered by the user becomes irrelevant and more queries can now be posed. Historically, all research in auditing has focused on static databases and known algorithms for auditing do not work in the presence of updates. Simple modifications to the algorithms are however sufficient and we conduct experiments to demonstrate how utility improves with updates. In summary,

- In Section 3 we introduce new algorithms for auditing **max** queries and bags of **max** and **min** queries to prevent partial disclosure. Working within the framework for probabilistic auditors introduced in [21], we show an interesting link to the problem of sampling graph colorings from a distribution.
- In Section 4 we introduce a new algorithm for auditing bags of **max** and **min** queries to prevent full disclosure.
- In Section 5 we provide initial utility results for **sum** auditing to prevent full disclosure.
- In Section 6 we present experiments on the utility of schemes for auditing **sum** and **max** queries.

While the kinds of queries we examine may seem simplistic, the auditing problem in all its generality is hard, and considering a restricted set of queries helps build our understanding of the problem. Note that useful information can already be derived via simple queries. For example, when releasing contingency tables, **sum** queries are the only type of queries that are answered. Moreover, as shown in [21], with very simple queries we can already illustrate significant privacy breaches.

We discuss related work and preliminaries in Section 2. A more general overview can be found in [1, 7]. We conclude with future work in Section 7.

2. BACKGROUND

2.1 Related Work

Online auditing was first studied in [11] and [25]. The authors look at **sum** queries in particular and protect privacy by restricting sizes and pairwise overlaps of allowable queries. In this scheme, if each query set is restricted to be of size at least k and if each pair of query sets is allowed to overlap in at most r elements, then $(2k - (l + 1))/r$ distinct queries can be answered where l is the number of x_i s known to the attacker beforehand. So if $k = n/c$ for some constant c and $r = 1$, then after only a constant number of distinct queries, the auditor would have to deny all further queries since there are only about c queries where no two overlap in more than one element. This motivates a search for auditing schemes that could provide greater utility.

In [8], the authors consider the *offline auditing problem* for **sum**, **max**, **max-and-min** and **sum-and-max** queries. In the offline version of the problem, we are given a sequence of queries q_1, \dots, q_t that have already been posed and truthfully answered and are required to determine whether compromise has already occurred. The **sum-and-max** auditing problem was shown to be NP-hard while efficient algorithms were derived for all the others.

In [20], the authors consider **sum** auditing for subcube queries where queries are specified as strings of 0s, 1s and *s (don’t cares). The elements to be summed up are those whose public attribute values match the query string pattern. The authors in [22] consider the boolean auditing problem for **sum** queries where the private attribute values are boolean. They also provide a **max** auditor for real-valued data. These results pertain to the offline auditing problem. In [2] the authors provide an offline auditing framework for determining whether a database system adheres to its data disclosure policies. The auditor detects queries that accessed sensitive data by formulating an *audit expression* that declaratively specifies sensitive table cells.

The authors in [5] look at the online auditing problem in a logic-oriented model of information systems and devise a hybrid approach of modifying query answers and denying queries in order to enforce security policies. In [21], the online auditing problem is looked at again. The authors illustrate how denials that depend on the answer to the current query can leak information and introduce the notion of simulatability to tackle this problem. They provide simulatable algorithms for auditing **sum** queries and **max** queries under classical compromise. In addition, they introduce a probabilistic notion of compromise and provide an algorithm for auditing **sum** queries over real-valued data drawn uniformly from a bounded range under this notion.

2.2 Preliminaries

We first define the different notions of compromise that we consider and then give a brief overview of tools that are used in our solutions.

Classical Compromise/Full Disclosure: Under this definition, a compromise occurs if any one private data point can be uniquely determined, i.e., in all datasets X with answers a_1, \dots, a_t , to queries q_1, \dots, q_t , some data point x_i would be the same. The drawback of this definition is obvious — even though a private value may not be uniquely determined, it may still be deduced to lie in a tiny interval or in a large interval with a skewed distribution, and some may consider this to be sufficient compromise. Nevertheless, we study this case as it is conceptually clean and has an appealing combinatorial structure.

Probabilistic Compromise/Partial Disclosure: This notion of privacy defined in [21] aims to mitigate the problems with the classical definition of compromise by modeling the change in the attacker’s confidence about the values of data points. It bounds the ratio of the posterior to prior probabilities that an x_i lies in an interval I ¹. The dataset, $X = \{x_1, \dots, x_n\}$, is assumed to be drawn according to a distribution D from $[\alpha, \beta]^n$. We assume that the distribution D is public, and in particular is known to the attacker. This is a reasonable assumption since, in practice, data such as age or salary have known probability distributions. Given queries q_1, \dots, q_t , corresponding answers a_1, \dots, a_t and predefined security parameters, λ and γ , we define

$$S_{\lambda, i, I}(q_1, \dots, q_t, a_1, \dots, a_t) = \begin{cases} 1 & \text{if } (1 - \lambda) \leq \frac{\Pr(x_i \in I | q_1, \dots, q_t, a_1, \dots, a_t)}{\Pr(x_i \in I)} \leq 1/(1 - \lambda) \\ 0 & \text{otherwise} \end{cases}$$

Let \mathcal{I} be the set of intervals $[\alpha + \frac{(j-1)(\beta-\alpha)}{\gamma}, \alpha + \frac{j(\beta-\alpha)}{\gamma}]$ for $j = 1, \dots, \gamma$.

$$S_\lambda(q_1, \dots, q_t, a_1, \dots, a_t) = \bigwedge_{i \in [n], I \in \mathcal{I}} S_{\lambda, i, I}(q_1, \dots, q_t, a_1, \dots, a_t).$$

$S_\lambda(q_1, \dots, q_t, a_1, \dots, a_t)$ thus evaluates to 1 if every single data point, x_i is “safe” with respect to every single interval, $I \in \mathcal{I}$. The attacker poses a query, q_t in each round, t for up to T rounds. The auditor can choose to deny a query and the attacker wins if $S_\lambda(q_1, \dots, q_t, a_1, \dots, a_t) = 0$ in some round. This is the (λ, γ, T) -privacy game and an auditor is $(\lambda, \delta, \gamma, T)$ -private if for any attacker, A :

$$\Pr[A \text{ wins the } (\lambda, \gamma, T)\text{-privacy game}] \leq \delta$$

Simulatable Auditing: The notion of simulatable auditing was introduced in [21]. The need for simulatability arose out of the authors’ discovery that denials based on the answer to the current query can leak considerable information.

Example: Suppose an attacker asks for $q_1 = \max\{x_a, x_b, x_c\}$ and is returned the answer 9. Later the attacker asks for $q_2 = \max\{x_a, x_b\}$. If the answer to q_2 is less than 9, then the attacker can determine that x_c must be 9 and q_2 should be denied. If however, the answer is exactly 9, answering q_2 would not leak information under the classical definition of compromise. In this situation, if the auditor does look at the answer to q_2 when choosing to deny, a denial would immediately imply that x_c must be 9 and privacy is breached.

¹This is similar to the definition used to prove that the one-time pad is secure [17] and is related to the γ -amplification definition of privacy suggested in [15].

The auditor should not therefore look at the true answer to the current query when making a decision. In fact, the attacker should be able to “simulate” the auditor and predict on his own when queries will be denied. This would ensure that privacy is never breached. The algorithms we look for in this paper should thus be online *and* simulatable. In the case of classical compromise, it suffices that the auditor determine if there is any possible answer to the current query that is consistent with past queries that could cause compromise. In the case of probabilistic compromise, it suffices that the auditor determine if compromise would occur in a large fraction of datasets drawn from the original distribution D conditioned on past query answers.

We now introduce a tool that is used in Sections 3 and 4 to maintain query logs.

Synopsis Computing Blackbox \mathcal{B} : This blackbox is introduced in [8] for the *offline* auditing of \max queries over duplicate-free real-valued data. It takes as input a set of \max queries and corresponding answers and converts them to a synopsis B_{\max} containing predicates of the form $[\max(S_i) = M_i]$ and $[\max(S_j) < M_j]$. Because there are no duplicates, \mathcal{B} can ensure that query sets of predicates in the synopsis are pairwise disjoint, thereby ensuring that the size of the synopsis is $O(n)$.

Example: Consider the queries $q_1 = \max\{x_a, x_b, x_c\} = 9$ and $q_2 = \max\{x_a, x_b\} = 9$. Since there are no duplicates, the intersection of the two query sets must contain the max value of 9. The queries would thus be converted to predicates $[\max\{x_a, x_b\} = 9]$ and $[\max\{x_c\} < 9]$ in the synopsis.

The authors show that it suffices to consider just these predicates generated by \mathcal{B} in detecting compromise instead of the entire sequence of past queries. \mathcal{B} can similarly be given a set of \min queries with answers and it returns a synopsis B_{\min} of predicates of the form $[\min(S_i) = m_i]$ and $[\min(S_j) > m_j]$ where S_i and S_j are disjoint subsets of X . The advantage of \mathcal{B} is that we can compress a potentially long audit trail to one of size $O(n)$ where n is the number of elements in the dataset. The synopses can be incrementally maintained — when a new query q_t arrives, the old synopsis together with q_t is combined in to a new synopsis in $O(|Q_t|)$ time.

3. AUDITING TO PREVENT PARTIAL DISCLOSURE

The authors in [21] provide an algorithm for auditing \sum queries over data points taken uniformly from the range $[\alpha, \beta]$ under probabilistic compromise (see Section 2.2), leaving open the problem of auditing other types of queries for data taken from other distributions. They also show how \max queries, when posed against naive auditors, can be used to reveal large fractions of the private data. Building a robust auditor for such queries is thus essential. In this part of the paper, we show how \max queries and bags of \max and \min queries can be audited under probabilistic compromise. Our algorithms use the general framework for probabilistic auditors introduced in [21], however they also use techniques that are interesting in their own right for the non-trivial problem of inferring posterior distributions of data values. Since the case of \max queries is simpler, we start with this.

