Towards Robustness in Query Auditing

Shubha U. Nabar[1] Bhaskara Marthi[2] Krishnaram Kenthapadi[1] Nina Mishra[3]
Rajeev Motwani[1]

[1]{sunabar, kngk, rajeev}@cs.stanford.edu [2]bhaskara@cs.berkeley.edu [3]nmishra@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 recieve 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 $X = \{x_1, \ldots, x_n\}$ be the multiset of sensitive attribute values in the database. Each x_i is the sensitive value of the ith individual. In our scenario the x_is are taken to be real-valued from a bounded or unbounded range. A statistical query $q = (Q, f)$ specifies a subset of the records $Q \subseteq \{1, \ldots, n\}$ 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 $q_1, \ldots, q_{t-1}$ that have already been posed, answers $a_1, \ldots, a_{t-1}$ 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

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

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_is 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 $q_1, \ldots, q_t$ 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 $a_1, \ldots, a_t$, to queries $q_1, \ldots, q_t$, 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, $X = \{x_1, \ldots, x_n\}$, 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 $q_1, \ldots, q_t$, corresponding answers $a_1, \ldots, a_t$ and predefined security parameters, λ and γ, we define

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

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

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

$S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t)$ 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 $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ 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:

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

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 $q_1 = \texttt{max}\{x_a, x_b, x_c\}$ and is returned the answer 9. Later the attacker asks for $q_2 = \texttt{max}\{x_a, x_b\}$. 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.

[1]This 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 $[\texttt{max}(S_i)] = M_i]$ and $[\texttt{max}(S_j) < M_j]$. 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 $q_1 = \texttt{max}\{x_a, x_b, x_c\} = 9$ and $q_2 = \texttt{max}\{x_a, x_b\} = 9$. 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 $[\texttt{max}\{x_a, x_b\} = 9]$ and $[\texttt{max}\{x_c\} < 9]$ 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 $[\texttt{min}(S_i) = m_i]$ and $[\texttt{min}(S_j) > m_j]$ 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.

3.1 Max Auditing

We would like to build a simulatable auditor for `max` queries that is $(\lambda, \delta, \gamma, T)$-private for any attacker, A. We assume that the dataset X of sensitive values is taken uniformly at random from the set of duplicate-free points in the unit cube $[0,1]^n$. [2] While the uniform distribution assumption is not usually a realistic assumption to make in practice, we believe that our techniques can be extended to other more practical distributions in the future.

Recall that under probabilistic compromise, a query is safe to answer if doing so is not likely to cause a significant change in the attacker's confidence that an x_i lies in a particular interval. Also, the decision to deny must be simulatable. The basic idea used here and in the next section is to have the auditor generate random datasets (of n sensitive values) uniformly from all datasets consistent with answers to past queries. The auditor then checks to see if answering the new query on the random datasets causes a significant change in the attacker's confidence about any x_i. If the answer is no for a sizable fraction of the generated datasets, the query is safe to answer. It turns out that generating a consistent random dataset and determining posterior confidences in the case of `max` queries can be done quite efficiently.

We first consider the problem of determining whether knowing $\texttt{max}(Q_t)$ causes a significant change in the attacker's confidence about a particular x_i. For this we make use of the blackbox $\mathcal{B}$ described in Section 2.2. Given queries $q_1, \ldots, q_t$ and answers $a_1, \ldots, a_t$, $\mathcal{B}$ returns a synopsis, $B_{\max}$ that represents all derivable information as predicates of the form $[\texttt{max}(S) = M]$ or $[\texttt{max}(S) < M]$. Recall that due to the absence of duplicates, each x_i occurs in at most one such predicate. For any x_i and any interval I, we would thus like to find $Pr\{x_i \in I \mid B_{\max}\}$. The prior probability that an x_i lies in an interval I of length $1/\gamma$ is given by $Pr\{x_i \in I\} = 1/\gamma$ as x_i is initially uniformly distributed in $[0,1]$. [3]

Posterior distribution of the x_is: The posterior probability distribution of x_i given $B_{\max}$ is part continuous and part discrete and the computations here require measure-theoretic justifications that we omit. But it can be shown that if $x_i \in S$ and $[\texttt{max}(S) = M] \in B_{\max}$, then x_i is uniformly distributed in $[0, M)$ with probability $1 - 1/|S|$ and $x_i = M$ with probability $1/|S|$. On the other hand, if $x_i \in S$ and $[\texttt{max}(S) < M] \in B_{\max}$ then x_i is just uniformly distributed in $[0, M)$ with probability 1. This is because, since x_i can occur in at most one predicate in B_{max}, other predicates do not affect x_i, i.e.,
$Pr\{x_i \in I \mid B_{max}\} = Pr\{x_i \in I \mid \text{predicate containing } x_i\}$.

Example: If $[\texttt{max}\{x_a, x_b, x_c\} = 0.75] \in B_{\max}$, then since any one of the data points is equally likely to be the max value, $x_a = 0.75$ with probability $1/3$ and with the remaining $2/3$ probability, it is uniformly distributed in $[0, 0.75)$.

Thus for any interval in $[0,1]$ it is possible to determine the ratio of the posterior to prior probability that an x_i lies in it. This leads to Algorithm 1 that returns false if $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ and true otherwise. For each x_i, the algorithm considers three cases — either x_i belongs to an equality predicate, or it belongs to a predicate with strict inequality, or it does not appear in any predicate at all. Let M be the max value of the predicate to which x_i belongs. Then for each interval, I_j, the algorithm considers three cases — I_j occurs to the left of the interval containing M, or I_j contains M, or I_j occurs to the right of the interval containing M. For each of these cases, we work out the math to get the posterior probability that x_i lies in I_j.

Input: Queries and answers q_l and a_l for $l = 1, \ldots, t$ and parameters λ, γ, n.
Let safe = *true*
For each x_i and each interval I_j in $\mathcal{I}$
  // x_i belongs to a predicate with equality
  If $x_i \in S$ for S s.t. $[\texttt{max}(S) = M] \in B_{max}$
    Let $y = \frac{1-1/|S|}{M\gamma}$ // $Pr\{x_i \in I_j | B_{max}\}$ if $M \notin I_j$
    If $j < \lceil M\gamma \rceil$ // I_j is to the left of M
      If $\gamma y \notin [1-\lambda, \frac{1}{1-\lambda}]$
        Let safe = *false*
    Else If $j = \lceil M\gamma \rceil$ // I_j contains M
      If $\gamma(y(M\gamma - \lceil M\gamma \rceil + 1) + \frac{1}{|S|}) \notin [1-\lambda, \frac{1}{1-\lambda}]$
        Let safe = *false*
    Else // I_j is beyond M
        Let safe = *false*
  // x_i belongs to a predicate with strict inequality
  Else If $x_i \in S$ for S s.t. $[\texttt{max}(S) < M] \in B_{max}$
    Let $y = \frac{1}{M\gamma}$ // $Pr\{x_i \in I_j | B_{max}\}$ if $M \notin I_j$
    If $j < \lceil M\gamma \rceil$ // I_j is to the left of M
      If $\gamma y \notin [1-\lambda, \frac{1}{1-\lambda}]$
        Let safe = *false*
    Else If $j = \lceil M\gamma \rceil$ // I_j contains M
      If $\gamma y(M\gamma - \lceil M\gamma \rceil + 1) \notin [1-\lambda, \frac{1}{1-\lambda}]$
        Let safe = *false*
    Else // I_j is beyond M
        Let safe = *false*
Return safe.

Algorithm 1: Safe

Simulatable auditor: We now construct the simulatable auditor by estimating the probability that answering q_t would breach privacy. The probability is taken over the distribution from which the dataset is drawn and is conditioned on the past queries and their answers. To estimate this probability, we draw a random dataset X' that is consistent with the answers already given, compute an answer a'_t according to X' and evaluate Algorithm 1 on $q_1, \ldots, q_t, a_1, \ldots, a_{t-1}, a'_t$ to see if this answer would cause a considerable change in the attacker's confidence about any one data point. The sampling is then repeated to get a good estimate.

In order to sample X' uniformly from all datasets consistent with the past answers, the auditor looks at each predicate $[\texttt{max}(S) = M]$ or $[\texttt{max}(S) < M]$ in $B_{\max}$. If the predicate has an equality, then it sets some $x_i \in S$ to M and assigns to other x_i's in S a value drawn uniformly at random from $[0, M)$. If the predicate has a strict inequality it assigns each $x_i \in S$ a value drawn uniformly at random from $[0, M)$. Every other x_i not represented by a predicate is given a value drawn uniformly at random from $[0,1]$. In case $x_i = x_j$ for some $i \neq j$, the sample is thrown out and another is chosen.

[2] The algorithm can easily be extended to other ranges.
[3] The prior distribution may not seem uniform since we are restricting ourselves to datasets with no duplicates. However, the set of points in the n dimensional cube where $x_i = x_j$ for some $i \neq j$ has measure 0.

But this event happens with probability zero. Algorithm 2 gives the details for the auditor.

1: **Input:** past queries $q_1, \ldots, q_{t-1}$, answers $a_1, \ldots, a_{t-1}$, new query q_t, parameters $\lambda, \gamma, n, \delta, T$.
2: **For** $O(\frac{T}{\delta} \log \frac{T}{\delta})$ times
3: Sample a data set X' consistent with past answers
4: Evaluate Algorithm 1 on input $q_1, \ldots, q_t, a_1, \ldots, a'_t$ and parameters λ, γ, n
5: **If** the fraction of sampled data sets for which Algorithm 1 returned *false* is more than $\delta/2T$
6: Return "denied"
7: **Else** return $a_t = \texttt{max}_X(Q_t)$

Algorithm 2: Simulatable auditor for `max` queries

THEOREM 1. *Algorithm 2 is a $(\lambda, \delta, \gamma, T)$-private simulatable auditor for max queries.*

PROOF. An attacker wins the game in round t if he poses a query q_t for which $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, f_t(Q_t)) = 0$ and the auditor does not deny q_t. The probability that the attacker wins in round t given the answers to previous queries is given by:

$$p_t = Pr\{S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_{t-1}, \texttt{max}_{X'}(Q_t)) = 0 \mid q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}\}$$

where X' is a dataset drawn uniformly from all datasets consistent with the previous answers. Thus Algorithm 2 essentially estimates p_t via multiple draws of random data sets X'. When $p_t > \delta/T$, by the Chernoff bound, the fraction of sampled data sets for which Algorithm 1 returns *false* is larger than $\delta/2T$ with probability at least $1 - \delta/T$. Hence if $p_t > \delta/T$ the attacker wins with probability at most δ/T. When $p_t < \delta/T$, the attacker wins only if the query is answered and even then with probability p_t. In both cases the attacker wins with probability at most δ/T. Thus the probability that the attacker wins in any of the T rounds is less than δ by the union bound. □

This concludes our construction of the `max` auditor. Note that this auditor is decidedly more efficient than the probabilistic `sum` auditor of [21] which needs to estimate volumes of convex polytopes in making decisions. Indeed, the running time is $O(\frac{T}{\delta}\gamma n \log \frac{T}{\delta})$. The critical insight is that each x_i is uniformly distributed between 0 and its upper bound as determined by answers to past queries, with a point mass at this max value. In case of bags of `max` and `min` queries the posterior distribution is not as easy to obtain.

3.2 Max-and-Min Auditing

Once again we assume that the dataset is taken uniformly from points in the unit cube $[0, 1]^n$ that do not have $x_i = x_j$ for any $i \neq j$. We consider the problem of building a $(\lambda, \delta, \gamma, T)$-private simulatable auditor for bags of `max` and `min` queries. Due to space constraints we give just a sketch of the main ideas involved in constructing the auditor.

As before, we would like to determine if answering the current query would lead to a significant change in the attacker's confidence about the value of any data point. We first consider the problem of determining whether knowing the answer a_t to q_t causes a significant change in the attacker's knowledge. We make use of the blackbox $\mathcal{B}$ described in Section 2.2. Given queries $q_1, \ldots, q_t$ and corresponding answers $a_1, \ldots, a_t$, $\mathcal{B}$ returns a synopsis $B = (B_{\max}, B_{\min})$. $B_{\max}$ represents all derivable information from the `max` queries as predicates of the form $[\texttt{max}(S) = M]$ or $[\texttt{max}(S) < M]$ where every data point can occur in at most one such predicate. And $B_{\min}$ represents all derivable information from the `min` queries as predicates of the form $[\texttt{min}(S) = m]$ or $[\texttt{min}(S) > m]$ where every data point can occur in at most one such predicate. We further modify the predicates as follows: if any `max` predicate and `min` predicate have the same value, M, then the corresponding query sets, S_1 and S_2, must have exactly one element, x_j, in common due to the assumption of no duplicates. We remove the two predicates and replace them with the predicates $[\texttt{max}(x_j) = M]$, $[\texttt{max}(S_1 - x_j) < M]$ and $[\texttt{min}(S_2 - x_j) > M]$. At the end of this process no two `max` and `min` predicates will have the same answer and each x_i can be determined to lie in a range R_i. Let ℓ_i be $1/|R_i|$.

Posterior distribution of the x_is: The prior probability that an x_i lies in an interval I of length $1/\gamma$ is $Pr\{x_i \in I\} = 1/\gamma$ as x_i is initially uniformly distributed in $[0, 1]$. We would like to find the posterior probability $Pr\{x_i \in I \mid B\}$.

Example: Consider two predicates of the form $[\texttt{max}\{x_a, x_b, x_c\} = 1]$ and $[\texttt{min}\{x_a, x_b\} = 0.2]$. We know that x_a and x_b must lie in the range $[0.2, 1]$ and x_c must lie in the range $[0, 1]$. Moreover, one of x_a or x_b must be exactly 0.2, and one of x_a, x_b or x_c must be exactly 1. Now if $x_a = 1$, then x_b must be 0.2 and x_c can lie anywhere in the range $[0, 1)$. This determines a line segment of (one-dimensional) volume 1. Enumerating over the other three possible cases, we find that the volume covered by all possible points that satisfy the two constraints is 3.6. Since the volume covered by points that satisfy the two constraints and the condition that $x_a = 1$ is 1, it follows that $Pr\{x_a = 1 \mid B\} = 1/3.6 = 5/18$.

Equivalent graph coloring problem: The above procedure can be generalized, but the number of cases to sum over may be exponential in the number of queries. So instead, we approximate the required probability by *sampling* from the joint distribution over the entire dataset X. As an intermediate step, consider the following graph coloring problem. Let $\mathcal{V}$ be the set of predicates in B with a strict equality and let there be k such predicates. Recall that each x_i can appear in at most one `max` predicate and at most one `min` predicate in $\mathcal{V}$. Moreover, no two predicates in $\mathcal{V}$ have the same answer. We construct a graph $\mathcal{G}$ with a node corresponding to each predicate in $\mathcal{V}$. Let $S(v)$ denote the query set of a predicate $v \in \mathcal{V}$ and let $A(v)$ denote the answer of the predicate. $\mathcal{G}$ has an edge between v_1 and v_2 iff $S(v_1) \cap S(v_2) \neq \emptyset$, i.e., if the two query sets have some elements in common. Each x_i in the dataset corresponds to a color, and the set of colors available at node v is just the query set $S(v)$. A *valid* coloring of $\mathcal{G}$ is a mapping c that assigns to each node v a color $c(v) \in S(v)$, such that if there is an edge between v_1 and v_2, then $c(v_1) \neq c(v_2)$. We define a probability distribution over valid colorings $\tilde{P}(c) = \frac{1}{Z} \prod_{v \in \mathcal{V}} \ell_{c(v)}$, where Z is a constant[4] chosen to make $\tilde{P}$ sum to 1, and where again l_i is 1 over the length of the interval R_i that defines the legitimate range of x_i.

The intuition is that the process of generating a dataset

[4]Note that our algorithm does not explicitly compute Z.

according to the posterior distribution can be split into two parts. First, for each `max` or `min` query with equality, we choose an item from its query set which will achieve the bound. This is exactly the information contained in the coloring. After doing this for every query, the values of the remaining items may now be chosen uniformly at random from their respective ranges. More formally,

LEMMA 1. *The following procedure generates a sample from $P(X \mid B)$:*

1. *Sample a coloring according to $\tilde{P}(c)$.*
2. *For each v, set $x_{c(v)} = A(v)$.*
3. *For each remaining unassigned item x_i, sample a value uniformly at random from its range R_i.*

PROOF. Given a dataset X that satisfies the queries, we can associate to it a unique coloring c as follows: for each predicate v with strict equality, there must be exactly one item x_i in $S(v)$ such that $x_i = A(v)$. Set $c(v) = x_i$. Conversely, a coloring c specifies the values of a subset of the variables as described in step 2 above, and the remaining ones can take any value in their range. We can therefore write $P(X \mid B) = P(c \mid B)P(X \mid c, B)$. Now, given B and c, the remaining unassigned variables are independent and uniformly distributed over the ranges determined by B. Thus, step 3 above generates samples from $P(X \mid c, B)$.

It remains to be shown that step 1 above generates samples from $P(c \mid B)$. To see this, we examine the geometry more closely. Suppose there are n items in the dataset, and k predicates with equality. Which datasets are compatible with the evidence? Once we choose a coloring, this determines the values of k items, and the remaining items may vary across their ranges. The posterior is thus uniform over a union of rectangles of dimension $n - k$. Each rectangle corresponds to a coloring c, and $P(c \mid B)$ is therefore proportional to the volume of this rectangle, which is the product of $1/\ell_i$ for those x_i that are not set by c. Equivalently, the probability is proportional to the product of ℓ_i for those x_i that *are* set by c, which is exactly the definition of $\tilde{P}$. □

We now describe a Markov chain $\mathcal{M}$ over valid colorings, which is a slightly modified version of ones found in the Markov Chain Monte Carlo literature (e.g. [16]). The Markov chain is initialized by looking at the actual state of the database and constructing the corresponding coloring. At each successive step, given a valid c, the chain generates c' as follows:

- Pick v uniformly from $\mathcal{V}$.
- For each $v' \neq v$, set $c'(v') = c(v')$.
- Pick a color x_i from $S(v)$ with probability proportional to ℓ_i.
- Set $c'(v) = x_i$ if this results in a valid coloring, and set it to $c(v)$ otherwise.

We now show the following lemma.

LEMMA 2. *If for every $v \in \mathcal{V}, |S(v)| \geq d_v + 2$, then the unique stationary distribution of $\mathcal{M}$ is $\tilde{P}$. Here d_v is the degree of node v.*

PROOF. If for every $v \in \mathcal{V}, |S(v)| \geq d_v + 2$, then $\mathcal{M}$ has a unique stationary distribution as shown in [16]. It remains to be shown that the stationary distribution is actually $\tilde{P}$. Let $\mathcal{M}_v$ denote the transition distribution in the case when we pick v in the first step above, i.e., each entry $\mathcal{M}_v[c_1 c_2]$ of the matrix denotes the probability of transitioning from coloring c_1 to coloring c_2 if v is chosen in the first step. Since $\mathcal{M}$ is a convex combination of the $\mathcal{M}_v$s, it suffices to show that each $\mathcal{M}_v$ preserves $\tilde{P}$, i.e., sampling from $\tilde{P}$ and then applying one step of $\mathcal{M}_v$ results in a coloring whose distribution is also $\tilde{P}$.

Let c_1 be a coloring sampled from $\tilde{P}$ and let c_2 be the resulting coloring after transitioning according to $\mathcal{M}_v$. Let c be any fixed coloring. We would like to show that $Pr\{c_2 = c\} = \tilde{P}(c)$. Let p be the total probability of colorings that agree with c on $\mathcal{V} \backslash v$ in $\tilde{P}$. Assume without loss of generality that $c(v) = 1$, the other valid colors for v given the rest of c are $2, \ldots, r$, and the remaining colors in $S(v)$ are $r+1, \ldots, s$. To get c after one step of $\mathcal{M}_v$, either we must have begun with c and chosen $c(v)$ or an invalid color, or we must have begun with a coloring that differed from c only on v, and then chosen the color $c(v)$. Use the notation $\ell_{a:b}$ to denote $\ell_a + \ell_{a+1} + \ldots + \ell_b$. The total probability of these events is

$$\begin{aligned} p\frac{\ell_1}{\ell_{1:r}}\frac{\ell_1 + \ell_{r+1:s}}{\ell_{1:s}} + p\frac{\ell_{2:r}}{\ell_{1:r}}\frac{\ell_1}{\ell_{1:s}} &= p\frac{\ell_1}{\ell_{1:r}}(1 - \frac{\ell_{2:r}}{\ell_{1:s}}) + p\frac{\ell_{2:r}}{\ell_{1:r}}\frac{\ell_1}{\ell_{1:s}} \\ &= p\frac{\ell_1}{\ell_{1:r}} \\ &= \tilde{P}(c) \end{aligned}$$

□

We can also show the following:

LEMMA 3. *Let Δ be the maximum degree of $\mathcal{G}$, and p_{max} and p_{min} be, respectively, the maximum and minimum conditional probabilities for the color of some v given a coloring of the rest of the graph. Let m be the minimum over all nodes of the number of colors allowed for that node. If $m > \Delta(1 + 2\frac{p_{max}}{p_{min}})$, then $\mathcal{M}$ has mixing time $O(k \log(k))$, where k is the number of equality predicates in $\mathcal{B}$ or equivalently the number of nodes in the graph.*

The (omitted) proof is an adaption of that given, e.g., in [16], the main difference being that in our case the distribution over colorings is non-uniform — each color can not be assigned to each vertex and the limiting distribution is $\tilde{P}$. Together, these lemmas show that by starting with any valid coloring and running $\mathcal{M}$ for $O(k \log(k))$ steps, we can generate a sample from a distribution that is close to $\tilde{P}$, provided the synopsis graph satisfies the condition in Lemma 2 (we will show how the auditor can enforce this condition after giving an overview of how the simulatable auditor functions if this condition is satisfied). We may thus answer probability questions up to accuracy ϵ by generating a set of such samples and forming a Monte Carlo estimate. In particular, for any x_i and interval I, we can get arbitrarily close estimates of $Pr\{x_i \in I \mid B\}$.

In our application, m is the minimum size of a query set and Δ is the maximum, over all predicates v, of the number of other predicates v' whose query set intersects that of v. The condition on m in Lemma 3 thus states that the more overlap there is between queries, the larger the query sets have to be to ensure mixing. If the condition fails to hold, it is also possible to convert the problem to one of inference in probabilistic graphical models, and use one of several standard techniques for approximate inference [19]. We omit the details for lack of space.

Simulatable auditor: As in the case of the `max` auditor, before answering q_t, the auditor generates random datasets consistent with answers to past queries by generating colorings according to distribution $\tilde{P}$ as described above. For each such dataset, X', he checks whether answering q_t in X' is likely to cause a privacy breach. This is done by generating random datasets consistent with all past answers as well as the answer to q_t in X' and estimating the posterior probability that each x_i lies in each interval. A significant difference in the posterior and prior probabilities that any x_i lies in any interval implies that q_t is not safe to answer in the dataset X'. If for a large fraction of sampled datasets, q_t is deemed unsafe, the query is rejected. If we ignore the ϵ error of the Monte Carlo estimates, we can show, just as in Theorem 1, that the above auditor is $(\lambda, \delta, \gamma, T)$-private.

Now in order to be able to generate datasets according to $\tilde{P}$, we need to ensure that the graph generated from the synopsis B satisfies the condition in Lemma 2, i.e., the number of colors available at each node (equivalently the size of each predicate) should be at least two more than the degree of the node (equivalently the number of other predicates that intersect this predicate). Note that the size of each predicate is trivially at least as large as the number of predicates that intersect it since every element can occur in at most one `max` query and at most one `min` query. Lemma 2 however, imposes a slightly stricter condition and we can ensure that it holds by denying, outright, queries that could possibly cause the condition to be violated. Since new queries affect the synopsis in different ways depending on their answers, and since we would like to avoid looking at the answer to a query for simulatability reasons, we deny a query if there is any answer to this query, consistent with the past synopsis that would cause the updated synopsis to violate the condition of Lemma 2. We can do this efficiently by looking at only a finite number of possibly consistent answers. The general idea is similar to one that we will see in Section 4 but we omit the details for lack of space. Note that these outright denials do not affect the probability of an attacker winning the privacy game and putting all these ideas together we get the following theorem.

THEOREM 2. *There exists a $(\lambda, \delta, \gamma, T)$-private simulatable auditor for combinations of* `max` *and* `min` *queries when the dataset is drawn uniformly at random from the set of duplicate-free points in the cube $[\alpha, \beta]^n$.*

This concludes our study of probabilistic auditors. In some cases, exact disclosure may be all that the DBA wishes to guard against. In such a scenario auditing can be made more efficient. We therefore turn to the problem of auditing the same queries under classical compromise next. Note that algorithms for classical compromise are not in general implied by algorithms for probabilistic compromise.

4. AUDITING TO PREVENT FULL DISCLOSURE

Prior to this work, no algorithm was known for auditing bags of `max` and `min` queries in an online fashion even for the basic case of full disclosure. In this section, we make the assumption that the database does not contain any duplicates and provide a simulatable auditor for the problem. Auditing in the presence of duplicates still remains open.

Given a set of previously posed `max` and `min` queries, $q_1, \ldots, q_{t-1}$ with answers, $a_1, \ldots, a_{t-1}$, we would like to check if there is any possible answer to a new query q_t that is consistent with past answers and would cause an x_i to be uniquely determined. Checking all possible answers a_t in $(-\infty, +\infty)$ would be impossible but we show that it is sufficient to check only a finite number of points. Let $Q'_1, \ldots, Q'_l$ be the query sets of previous queries that intersect with Q_t, ordered according to their corresponding answers, $a'_1 \leq \ldots \leq a'_l$. Let $a'_{lb} = a'_1 - 1$ and $a'_{ub} = a'_l + 1$ be the bounding values. Our algorithm only checks $2l + 1$ points — the bounding values, the above l answers and the mid-points of the intervals determined by them [5]. Algorithm 3 gives the details.

1: **For** $a_t \in \{a'_{lb}, a'_1, \frac{a'_1+a'_2}{2}, a'_2, \ldots, a'_{l-1}, \frac{a'_{l-1}+a'_l}{2}, a'_l, a'_{ub}\}$
2: **If** a_t is consistent with $a_1, \ldots a_{t-1}$ AND if $\exists 1 \leq j \leq n$ such that x_j is uniquely determined
3: Output "Deny" and return
4: Output $f(Q_t)$ and return

Algorithm 3: Simulatable auditor for `max`-and-`min` queries

We next address the questions of checking whether a value is uniquely determined and whether an answer a_t is consistent with answers to previous queries.

Checking if a value is uniquely determined: The idea here is to determine the set of *extreme elements* for every single query posed by the attacker. The extreme elements of a query set, Q_i, are the x_js whose values could potentially be the answer a_i to the query q_i. For each query set, we would like to eliminate those x_js that could certainly not be extreme elements and in the end if a query set has only one extreme element x_j, we know that x_j must take on the max value for that query set and privacy has been breached. If a query set has only one extreme element, then the element is said to be *strictly extreme* for the query set. This idea is similar in spirit to ideas in [21, 22] for auditing `max` queries, however the process of determining extreme elements has to be modified since additional inferences can be made due to the presence of `min` queries. In addition, we do not in fact need to examine all query sets posed by the user due to our assumption of no duplicates. In what follows however, we describe the algorithms as though all query sets were being maintained for ease of exposition, and later show how the size of the audit trail can be reduced.

[5]Note that since the total number of possible distinct answers to queries is n, $2l + 1$ is $O(n)$.

Given a set of queries and answers, the upper bound μ_j of an element x_j is the minimum over the answers to the `max` queries containing x_j, i.e., $\mu_j = \min\{a_k \mid j \in Q_k \text{ and } Q_k \text{ is a } \texttt{max} \text{ query}\}$. Similarly, the lower bound λ_j of an element x_j is defined as $\lambda_j = \max\{a_k \mid j \in Q_k \text{ and } Q_k \text{ is a } \texttt{min} \text{ query}\}$. We determine the extreme elements E_k for a query set Q_k as shown in Algorithm 4.

1: For $k = 1$ to t, $E_k = \emptyset$.
2: If Q_k is a `max` query set, $j \in Q_k$ and $\mu_j = a_k$ then $E_k = E_k \cup \{x_j\}$. Similarly if Q_k is a `min` query set, if $j \in Q_k$ and $\lambda_j = a_k$ then $E_k = E_k \cup \{x_j\}$.
3: For a `max` query Q_k look at all other `max` query sets with answer a_k and look at the largest set of extreme elements E common to all the query sets. Set $E_k = E$. This follows because there are no duplicates in the database. For all other x_j that were previously in E_k, we know that $x_j < \mu_j = a_k$, i.e., a_k is a strict upper bound for x_j. The analogous procedure is applied for `min` query sets.
4: If any extreme element, x_j, of a `max` query set, Q_M, is strictly extreme for some `min` query set, Q_m and $a_M \neq a_m$ then $x_j < a_M$ and $E_M = E_M - \{x_j\}$ (and similarly for extreme elements of a `min` query set).

Algorithm 4: Determining extreme elements

The last step in this procedure could spark a *trickle effect* and computing the final set of extreme elements for each query set takes $O(t^2\Sigma_{i=1}^{t}|Q_i|)$ time. Once extreme elements have been computed in this way, we can show the following theorem (proved in the full paper [24]).

THEOREM 3. *Given a set of queries, $q_1, \ldots, q_t$ and corresponding answers $a_1, \ldots, a_t$, the database is secure if and only if every* `max` *or* `min` *query set has more than one extreme element and there does not exist any* `max` *query, q_i and* `min` *query, q_j such that $a_i = a_j$.*

One direction of the theorem is simple enough to prove — if any one of the conditions in the theorem is violated, then clearly some element is uniquely determined. For the other direction, we show that if both conditions are satisfied, then for each element, x_i, we can construct two different datasets consistent with answers to the queries such that x_i takes on a different value in the two datasets.

Checking for consistency: Checking for consistency can also be done in polynomial time.

THEOREM 4. *Given a set of queries $q_1, \ldots, q_t$, responses $a_1, \ldots, a_t$ are consistent if and only if a) every* `max` *and* `min` *query set has at least one extreme element b) for every element x_i, $\mu_i > \lambda_i$ if either the upper or lower bound for x_i is a strict inequality and $\mu_i \geq \lambda_i$ otherwise and c) if $a_i = a_j$ for some* `min` *query q_i and some* `max` *query q_j, then Q_i and Q_j should have only one extreme element in common.*

This theorem is proved in the full paper — once again, if any one of these conditions is violated, it is easy to see that the answer to the query cannot be consistent with past answers. On the other hand, if all conditions are satisfied, we provide an algorithm to construct a consistent dataset. We now make the following claim also proved in the full paper.

THEOREM 5. *For $1 \leq j \leq n$, x_j is uniquely determined for some value of a_t in (a'_s, a'_{s+1}) if and only if x_j is uniquely determined when $a_t = \frac{a'_s + a'_{s+1}}{2}$. Furthermore, the values of a_t in (a'_s, a'_{s+1}) are either all consistent or inconsistent.*

This completes our proof of correctness. Since we run through the for loop in Algorithm 3 $2l + 1$ times, the running time of the algorithm is $O(t^3\Sigma_{i=1}^{t}|Q_i|)$. We now briefly outline how the running time can be further reduced due to the no-duplicates assumption.

The "no duplicates" assumption: Note that the assumption of no duplicates can be achieved by perturbing a dataset by negligible amounts if it does contain duplicates. This, however, may lead to a conservative auditor.

Example: Consider a scenario where a user asks for $\texttt{max}\{x_a, x_b, x_c\}$ and later for $\texttt{max}\{x_a, x_d, x_e\}$. Due to the restriction that there be no duplicates in the dataset, the second query will be denied since if both queries were to have the same answer, the value of x_a would be revealed. If duplicates are allowed however, both queries can be answered. This indicates that the kinds of queries that will be allowed with the no duplicates restriction, will be those with either no overlap or lots of overlap.

Although the "no duplicates" assumption thus reduces the utility delivered to the user, it does have an advantage. The size of the audit trail can be significantly reduced using blackbox $\mathcal{B}$ (see Section 2.2). $\mathcal{B}$ can be used to maintain separate synopses $B_{\max}$ and $B_{\min}$ for past queries $q_1, \ldots, q_{t-1}$ and answers $a_1, \ldots, a_{t-1}$. When a new query is asked, the new query along with a possible answer can be submitted to $\mathcal{B}$ to obtained updated synopses $B'_{\max}$ and $B'_{\min}$. Since $B'_{\max}$ captures all information contained in the `max` queries and their answers and $B'_{\min}$ captures all information contained in the `min` queries and their answers, we only need to consider these query sets in determining extreme elements as in Algorithm 4. We thus no longer need to maintain the entire sequence of queries that have been posed and an audit trail of size $O(n)$ suffices. This was not possible in the `max` auditing algorithm in [21] where duplicates were allowed.

That said, the question of auditing combinations of `max` and `min` queries in the presence of duplicates is an interesting one and remains an open problem. To see why duplicates make the problem harder, consider the following predicates: $\texttt{max}\{x_a, x_b\} = 9$ and $\texttt{max}\{x_c, x_d\} = 9$. Now if it is revealed that $\texttt{min}\{x_b, x_d\} = 1$, then in addition to the previous information, we also know that $\texttt{max}\{x_a, x_c\}$ must be 9 since one of x_b or x_d has to be 1. So in addition to examining extreme elements for query sets that have actually been posed, the auditor needs to examine extreme elements for such inferred query sets as well, and there can be a blow up in the number of query sets that need to be maintained. This could not happen in the absence of duplicates since the first two queries could never have the same answer. Finding an efficient algorithm that works in the presence of duplicates is an interesting avenue for future work.

We conclude our study of auditors for new kinds of queries for the different notions of compromise. We now move on to quantifying utility.

5. UTILITY

As a case study we examine the utility of the existing algorithm for auditing `sum` queries over real-valued data. Later in Section 6 we provide experimental results on the utility of `sum` auditing and `max` auditing for real-valued data under classical compromise.

While there are several dimensions along which utility could be measured (and we discuss these later), the dimension that we consider is the number of queries posed by a user that are actually answered. In particular we would like to measure the expected number of queries that will be answered in a sequence of random queries [6] posed by the user. The larger this number, the greater the utility. We show a surprisingly positive result for large databases - the number of queries that can be answered is at least a constant fraction of the size of the database. In practice, it is not likely that queries would be drawn from a uniform distribution. We believe, however, that our techniques could be extended to investigate more practical distributions in the future.

The simulatable `sum` auditing algorithm can be found in [9, 21]. Each query is expressed as a row in a matrix with a 1 for every record that is accessed by the query and a 0 for every record that is left out. The columns of the matrix thus correspond to the records and the rows to the queries. The row corresponding to a query is called its "query vector". Gaussian elimination on this matrix can be used to determine if an x_i can be solved for. For the sake of efficiency, the matrix is maintained in upper triangular form through a series of elementary row-space-preserving row operations and column changes. If the upper triangular form of the matrix contains a row with a 1 in only one column and 0s in all others, then some element can be uniquely determined. When a new query is posed, the auditor checks to see if the new query vector is already in the vector space spanned by the rows of the matrix in $O(nm)$ time, where m is the rank of the matrix. If it is, then the query is answered (the new query vector is not added to the matrix). If not, then adding the new query vector and rediagonalizing the matrix also takes $O(nm)$ steps. We consider a series of random queries posed to such an auditor and compute the expectation of the time to first denial, T_{denial}.

THEOREM 6. *$E[T_{denial}] \geq \frac{n}{4}(1 - o(1))$*

PROOF.

$$\begin{aligned} E[T_{\text{denial}}] &= \sum_{m=1}^{\infty} mPr\{T_{\text{denial}} = m\} \\ &= \sum_{m=0}^{\infty} Pr\{T_{\text{denial}} > m\} \\ &= 1 + \sum_{m=1}^{\infty} Pr\{T_{\text{denial}} > m\} \\ &= 1 + \sum_{m=1}^{\infty} Pr\{\text{No denial in } q_1, \ldots, q_m\} \end{aligned}$$

[6] A random query is a query drawn independently and uniformly at random from the set of all `sum` queries that could be formulated over the data.

We would thus like to get a lower bound on the probability that there is no denial in the first m of a sequence of random queries. Let the query vectors of the posed queries be $v_1, \ldots, v_m$ and let A be the matrix of query vectors. Note that $v_1, \ldots, v_m$ are random 0-1 vectors. Compromise occurs if we can find any real-valued λ_is such that $\Sigma_i \lambda_i v_i$ gives us an axis-parallel vector (a vector with only one 1 and $n-1$ 0s). Let A^T be the transpose of the matrix A. A^T is thus an $n \times m$ matrix of 0s and 1s. Now if we could find some m-dimensional vector w such that $A^T w$ gives us an axis-parallel vector, then the elements of w would correspond to the λ_is that give rise to the privacy breach. w must thus be perpendicular to all the rows in A^T except for the one row which gives rise to the 1 in the axis-parallel vector. We can now state that there is no denial amongst the first m queries if and only if there is no m-dimensional vector that is perpendicular to all but one of the rows in A^T. We thus need to lower bound the probability that there is no such m-dimensional vector.

Note that if we could divide the rows of A^T in to two disjoint sets, each of which forms a basis for $\mathbb{R}^m$, then there could be no such vector. This is because there can be no m-dimensional vector that is perpendicular to all the vectors that form the basis of $\mathbb{R}^m$. So if we take any row in the matrix and claim that there is some vector w perpendicular to all other rows but this, we know that this is not possible since a subset of the remaining rows must form a basis for $\mathbb{R}^m$. We thus need to lower bound the probability that A^T contains such a double basis. For this we consider the rows of A^T one after the other and calculate the probability that both the first $n/2$ and the last $n/2$ rows span $\mathbb{R}^m$ separately. The following lemma (proved in the full paper) is useful in finding this probability:

LEMMA 4. *A hyperplane H in $\mathbb{R}^m$ can intersect B^m in at most $2^{d(H)}$ points where $d(H)$ is the dimension of the hyperplane.*

Here B is the set $\{0, 1\}$ and B^m is the boolean hypercube in m dimensions. Note that the rows of A^T correspond to vertices of B^m. Let Y denote the rank of the matrix formed by the first $n/2$ rows of A^T. Let X denote the number of heads obtained in $n/2$ independent tosses of an unbiased coin. We will show that the random variable, Y, stochastically dominates the random variable, X, until its value reaches m, i.e., $Pr\{Y < t\} \leq Pr\{X < t\}\ \forall t \leq m$. As we consider the rows of A^T in order, we would like to find the probability that a new row raises the rank of the matrix, assuming that the rows so far together have rank $< m$. Suppose i rows have been considered thus far and the rank of the matrix formed by them is $l < m$. Then they span a hyperplane of dimension l. From our lemma, this can intersect B^m in at most 2^l points. Thus there are at most 2^l points in B^m that are linearly dependent on the i rows and at least $2^m - 2^l$ points that are linearly independent. So the probability that the $(i+1)^{st}$ row raises the rank of the matrix is $\geq 1 - 2^l/2^m \geq 1/2$. Thus, until its value reaches m, Y stochastically dominates X. Hence, the probability that the first $n/2$ rows of A^T span $\mathbb{R}^m$ is given by

$$Pr\{Y = m\} = Pr\{Y \geq m\} \geq Pr\{X \geq m\} \geq 1 - \frac{1}{n^2}$$

The last inequality follows from Chernoff bound for $m \leq n/4 - \sqrt{n \ln n}$. Similarly, the remaining $n/2$ rows also span $\mathbb{R}^m$ with high probability. Thus

$$\begin{aligned} E[T_{\text{denial}}] &\geq 1 + \sum_{m=1}^{n/4-\sqrt{n \ln n}} Pr\{\text{No denial in } q_1, \ldots, q_m\} \\ &\geq 1 + \sum_{m=1}^{n/4-\sqrt{n \ln n}} \left(1 - \frac{1}{n^2}\right)^2 = \frac{n}{4}(1 - o(1)) \end{aligned}$$

□

So the first denial occurs after $\Omega(n)$ queries. We next show that it is expected to occur within $O(n)$ queries.

THEOREM 7. $E[T_{denial}] \leq n + \lg n + 1$

PROOF.

$$E[T_{\text{denial}}] = \sum_{m=0}^{\infty} Pr\{T_{\text{denial}} > m\}$$

For values of m below $m^* = n + \lg(n \ln 2)$, we upper-bound $Pr\{T_{\text{denial}} > m\}$ by 1. This gives us

$$\begin{aligned} E[T_{\text{denial}}] &\leq n + \lg(n \ln 2) + \sum_{m=m^*}^{\infty} Pr\{T_{\text{denial}} > m\} \\ &= n + \lg(n \ln 2) + \\ &\quad \sum_{m=m^*}^{\infty} Pr\{\text{No denial in } q_1, \ldots, q_m\} \end{aligned}$$

Once again we consider the $n \times m$ matrix A^T and the probability that there is no vector perpendicular to all but one of its rows. Note that if the n rows were linearly independent, there would always be such a vector. So we would like to upper-bound the probability that the n random 0-1 rows of the matrix are not linearly independent. If we call this probability $P_{n,m}$, we can show that $P_{n,m} \leq n/2^{m-n+1}$. So

$$\begin{aligned} E[T_{\text{denial}}] &\leq n + \lg(n \ln 2) + \sum_{m=m^*}^{\infty} \frac{n}{2^{m-n+1}} \\ &= n + \lg(n \ln 2) + \frac{n}{2^{m^*-n}} \\ &= n + \lg(n \ln 2) + \frac{1}{\ln 2} \\ &\leq n + \lg n + 1 \end{aligned}$$

□

Thus we can expect the first denial to occur in $\Theta(n)$ queries. Note that our lower bound on T_{denial} is a high probability result — with high probability $(1 - \frac{1}{n^2})^2$, there will be no denial in the first $\frac{n}{4}(1 - o(1))$ queries posed to the database. This is a positive result for very large datasets — since n is large, many queries will be answered before even the first denial occurs. Our experiments in Section 6 attest to this.

The next question to ask is what happens after the first denial. Note that once the rank of the query matrix reaches $n - 1$, denials will occur with probability at least 1/2. This is because at least half the points in B^m will be linearly independent of the rows of the query matrix and if any of these are answered, the rank of the query matrix becomes n, enabling the attacker to solve the system of equations for all points in the dataset. The question to answer is thus, how quickly does the rank of the query matrix become $n - 1$ after the first denial. It seems that this would happen rapidly since we can show, using similar arguments, that the expected time till $n - 1$ linearly independent queries are posed is $\Theta(n)$. Not all these queries would be answered however, and an exact analysis remains to be done. Thus a problem arises when n is small.

Yet another issue that diminishes utility is that all users would have to be considered as one in order to prevent collusion attacks (note that this is a problem with perturbation approaches as well). Thus the queries of all the users would have to be pooled together and this may result in a user receiving more than his fair share of denials. There are two things that might help us out in this situation. The first is that users are unlikely to be able to pose queries over arbitrary subsets of the data and queries will more likely come from non-uniform distributions. The second is that databases such as hospital databases or the census database frequently experience updates in the form of insertions, deletions and modifications. For example, in the company database containing salaries of its employees, new employees may join the company, old ones may get fired, and employees often receive raises. In this scenario, as old information gathered by a user or by other users becomes out of date, more queries can be answered. For example, if a user asks for $x_a + x_b + x_c$ and x_a is subsequently modified, the user can now ask for $x_a + x_b$ and get an answer whereas without the update he would have been denied. The existing algorithms for auditing `sum`, `max` or `min` queries do not work as such in the presence of updates. However simple modifications, presented in the full version of the paper suffice. In the next section we present experiments to verify our theoretical results and to measure the increase in utility with updates and queries taken from non-uniform distributions.

6. EXPERIMENTS

We conducted experiments to measure the utility of `sum` auditing in various settings.

Time to first denial: The first experiment we performed was to issue random `sum` queries to databases of varying sizes and count the number of denials at regular intervals. Averaging over many trials, we obtained the probability of denial as more and more queries were posed. We found that this probability was more or less a step function. There would be no denials up to a point and then the probability of denial would rapidly increase to 1. Figure 1 shows a plot of the database size versus the threshold beyond which the probability of denial increases. The graph confirms our theoretical results — the number of queries that were answered before the first denial was in fact almost exactly equal to the size of the databases in all cases. This is good news for large databases. However, the utility of the databases rapidly deteriorated thereafter. Plot 1 of Figure 2 demonstrates how essentially every query is denied after roughly n queries. We

therefore experimented with updates and non-uniform query distributions to measure their impact on utility.

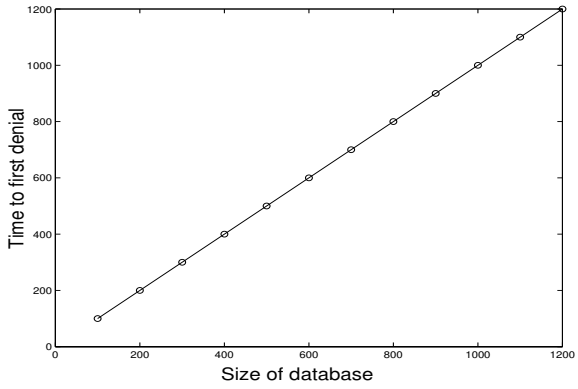

Figure 1: Time to first denial for `sum` queries

Updates: This next experiment aimed at measuring the increase in utility of the `sum` auditing scheme in the presence of updates. We allowed updates in the form of modifications to be made to the database once in every 10 queries and measured the probability of denial by averaging over many trials once again. The queries themselves were taken uniformly from the set of all possible `sum` queries that could be posed over the data. A query would be denied if answering it would enable the determination of any past or present value of the sensitive attribute for some individual. The algorithm is presented in the full version of the paper. Figure 2 shows the probability of denial for a dataset containing 500 elements as more and more queries were posed. Plot 1 shows the probability of denial when no updates were made to the system while Plot 2 shows the probability of denial in the presence of updates. Note that not only did the time to first denial shift to the right but also the long run probability of denial remained below Plot 1 because of the continuous updates. We consistently noted this impact on utility across database sizes.

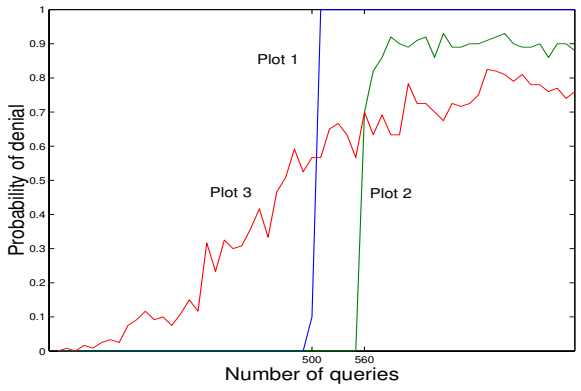

Figure 2: Probability of denial for `sum` queries
Plot 1: Queries drawn uniformly at random
Plot 2: In the presence of updates
Plot 3: 1-dimensional range `sum` queries

Non-uniform query distribution: Next we measured utility for SQL-like queries. We ordered the sensitive values in the dataset on a particular public attribute such as "age" and allowed only range queries on age to be posed over them. Moreover, each query was made to access somewhere between 50 and 100 elements in the dataset. Plot 3 in Figure 2 shows the probability of denial for queries taken from such a distribution. Once again the probability of denial never reached the worst case scenario that it did when queries were chosen uniformly at random and this behaviour was noted consistently over many trials on databases of varying sizes.

Max queries: We also experimented with the utility of the `max` auditing algorithm presented in [21]. This algorithm is similar to the algorithm for bags of `max` and `min` queries presented in Section 4, the difference being in the way that extreme elements are determined. Figure 3 depicts the probability of denial for random `max` queries posed over a dataset of 500 elements. The first few queries were never denied and then the probability of denial quickly rose to around 0.68 and stayed in that region. One encouraging observation is that the probability of denial never reaches the worst case scenario as in `sum` queries. An exact analysis of utility for `max` queries is an open problem.

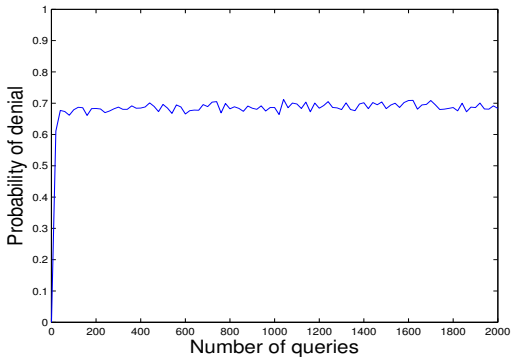

Figure 3: Probability of denial for `max` queries

The experiments confirm our theoretical results as well as our conjecture on the effect of updates and SQL-like queries on the utility of auditing schemes.

7. DISCUSSION AND FUTURE WORK

We view this paper as an early exploration of the online auditing problem, laying theoretical foundations for an area with potential practical promise. Many interesting directions lie ahead.

Perhaps the most obvious next step is to generalize the class of queries with simulatable auditors to handle more complex data computations. Alternatively, it may also be interesting to specialize the queries that can be audited if it leads to more realistic settings and/or more efficient auditors. For example, while the boolean auditing problem is known to be coNP-hard for arbitrary queries, if the queries are restricted to a one-dimensional form, such as how many individuals are between the ages of 15 and 25, then the auditing problem is known to have a linear-time solution [22]. Thus by

exploiting the fact that queries cannot range over arbitrary subsets of points — which may be realistic in some settings — we may be able to design efficient auditing algorithms that were previously not possible.

The utility of auditing algorithms has many interesting components including whether we deny too much, whether we should deny more, and what utility actually means. Intuitively, it seems that the more an auditing scheme denies, the less useful it is. Indeed, simulatability is conservative and could deny more often than necessary. One could try to analyze the *price of simulatability* — how many queries were denied when they could have been safely answered because we did not look at the true answers when choosing to deny. On the other hand, we could deny more often than is necessary for simulatability and privacy in the hope that doing so would enable more queries to be answered in the future. Such an approach could potentially ward off denial of service attacks where a malicious user poses queries in such a way that would cause many innocuous queries to be denied in the future. But a more fundamental question, is how to even define utility. There may be some important, fairly generic queries that the world would always like to have answered, e.g., the total number of cancer patients in a particular hospital. An auditing scheme that denies such a query could be construed as providing weak utility. In our schemes, we could add such important queries to the pool of queries already answered, thereby ensuring that these queries will always be answered in the future. But this solution is just one practical way to overcome the problem. The larger question here is: what is utility?

Finally, collusion is an interesting, important, open problem present in most privacy results to date. Our scheme implicitly assumes that queries from all users are treated as coming from one user. This strong assumption that all parties can collude with each other is made in almost all perturbation results that we are aware of. This means that we may deny far more often than necessary. A thorough understanding of the collusion problem would be a major advance.

Acknowledgements

This work was supported in part by NSF grants ITR-0331640 and EIA-0137761 and also in part by TRUST (The Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: Cisco, ESCHER, HP, IBM, Intel, Microsoft, ORNL, Qualcomm, Pirelli, Sun, and Symantec. The first author was supported in part by a Stanford Graduate Fellowship from Sequoia Capital. The authors would also like to thank T. K. Satish Kumar for helpful discussions and the anonymous reviewers for their comments.

8. REFERENCES

[1] N. Adam and J. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[2] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant. Auditing Compliance with a Hippocratic Database. In *Proc. of VLDB*, pages 516–527, 2004.

[3] R. Agrawal and R. Srikant. Privacy-preserving Data Mining. In *Proc. of ACM SIGMOD*, pages 439–450, 2000.

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

[5] J. Biskup and P. Bonatti. Controlled Query Evaluation for Known Policies by Combining Lying and Refusal. *Annals of Mathematics and Artificial Intelligence*, 40(1-2):37–62, 2004.

[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS*, pages 128–138, 2005.

[7] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *TCC*, pages 363–385, 2005.

[8] F. Chin. Security Problems on Inference Control for SUM, MAX, and MIN Queries. *J. ACM*, 33(3):451–464, 1986.

[9] F. Chin and G. Ozsoyoglu. Auditing for Secure Statistical Databases. In *Proceedings of the ACM '81 conference*, pages 53–59, 1981.

[10] I. Dinur and K. Nissim. Revealing Information while Preserving Privacy. In *PODS*, pages 202–210, 2003.

[11] D. Dobkin, A. Jones, and R. Lipton. Secure Databases: Protection against User Influence. *ACM Trans. Database Syst.*, 4(1):97–106, 1979.

[12] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our Data, Ourselves: Privacy via Distributed Noise Generation. In *EUROCRYPT*, 2006.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, pages 265–284, 2006.

[14] C. Dwork and K. Nissim. Privacy-Preserving Data Mining on Vertically Partitioned Databases. In *CRYPTO*, pages 528–544, 2004.

[15] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. In *PODS*, pages 211–222, 2003.

[16] A. Frieze and E. Vigoda. Survey of Markov Chains for Randomly Sampling Colorings. *Combinatorics, Complexity and Chance*, To Appear.

[17] O. Goldreich. *Foundations of Cryptography, Volumes I and II*. Cambridge University Press, 2004.

[18] http://www.stat.cmu.edu/∼hwainer/bertinoro.htm. CS-Statistics Workshop On Privacy and Confidentiality. 2005.

[19] M. I. Jordan. *Learning in Graphical Models*. MIT Press, 1998.

[20] J. Kam and J. Ullman. A model of statistical databases and their security. *ACM Trans. Database Syst.*, 2(1):1–10, 1977.

[21] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable Auditing. In *Proc. of ACM PODS*, pages 118–127, 2005.

[22] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing Boolean Attributes. *Journal of Computer and System Sciences*, 6:244–253, 2003.

[23] N. Mishra and M. Sandler. Privacy via Pseudorandom Sketches. In *PODS*, 2006.

[24] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards Robustness in Query Auditing. http://dbpubs.stanford.edu/pub/2006-16. *Technical Report*, 2006.

[25] S. Reiss. Security in Databases: A Combinatorial Study. *J. ACM*, 26(1):45–57, 1979.

[26] S. Warner. Randomized response: A survey technique for eliminating error answer bias. *J. of American Statistical Association*, 60(309):63–69, 1965.