Simulatable Auditing

Krishnaram Kenthapadi* Nina Mishra† Kobbi Nissim‡

ABSTRACT

Given a data set consisting of private information about individuals, we consider the *online query auditing problem*: given a sequence of queries that have already been posed about the data, their corresponding answers – where each answer is either the true answer or "denied" (in the event that revealing the answer compromises privacy) – and given a new query, deny the answer if privacy may be breached or give the true answer otherwise. A related problem is the offline auditing problem where one is given a sequence of queries and all of their true answers and the goal is to determine if a privacy breach has *already* occurred.

We uncover the fundamental issue that solutions to the offline auditing problem cannot be directly used to solve the online auditing problem since query denials may leak information. Consequently, we introduce a new model called *simulatable auditing* where query denials provably do not leak information. We demonstrate that max queries may be audited in this simulatable paradigm under the classical definition of privacy where a breach occurs if a sensitive value is fully compromised. We also introduce a probabilistic notion of (partial) compromise. Our privacy definition requires that the a-priori probability that a sensitive value lies within some small interval is not that different from the posterior probability (given the query answers). We demonstrate that sum queries can be audited in a simulatable fashion under this privacy definition.

*Department of Computer Science, Stanford University, Stanford, CA 94305. Supported in part by NSF Grants EIA-0137761 and ITR-0331640, and a grant from SNRC. kngk@cs.stanford.edu.

†HP Labs, Palo Alto, CA 94304, and Department of Computer Science, Stanford University, Stanford, CA 94305. nmishra@cs.stanford.edu. Research supported in part by NSF Grant EIA-0137761.

‡Department of Computer Science, Ben-Gurion University, P.O.B. 653 Be'er Sheva 84105, ISRAEL. Work partially done while at Microsoft Research, SVC. kobbi@cs.bgu.ac.il.

PODS 2005 June 13-15, 2005, Baltimore, Maryland.

1. INTRODUCTION

Let X be a data set consisting of n entries $X = \{x_1, \ldots, x_n\}$ taken from some domain (for this work we will typically use $x_i \in R$). The data sets X of interest in this paper contain private information about specific individuals. A statistical query $q = (Q, f)$ specifies a subset of the data set entries $Q \subseteq [n]$ and a function f (usually $f : 2^R \to R$). The answer $f(Q)$ to the statistical query is f applied to the subset of entries $\{x_i | i \in Q\}$. Some typical choices for f are sum, max, min, and median. Whenever f is known from the context, we abuse notation and identify q with Q.

We consider the *online query auditing problem*: Suppose that the queries $q_1, \ldots, q_{t-1}$ have already been posed and the answers $a_1, \ldots, a_{t-1}$ have already been given, where each a_j is either the true answer to the query $f_j(Q_j)$ or "denied" for $j = 1, \ldots, t-1$. Given a new query q_t, deny the answer if privacy may be breached, and provide the true answer otherwise.

Many data sets exhibit a competing (or even conflicting) need to simultaneously preserve individual privacy, yet also allow the computation of large-scale statistical patterns. One motivation for auditing is the monitoring of access to patient medical records where hospitals have an obligation to report large-scale statistical patterns, e.g., epidemics or flu outbreaks, but are also obligated to hide individual patient records, e.g., whether an individual is HIV+. Numerous other datasets exhibit such a competing need – census bureau microdata is another commonly referenced example.

One may view the online auditing problem as a game between an auditor and an attacker. The auditor monitors an online sequence of queries, and denies queries with the purpose of ensuring that an attacker that issues queries and observes their answers would not be able to 'stitch together' this information in a manner that breaches privacy. A naive and utterly dissatisfying solution is that the auditor would deny the answer to every query. Hence, we modify the game a little, and look for auditors that provide as much information as possible (usually, this would be "large scale" information, almost insensitive to individual information) while preserving privacy (usually, this would refer to "small-scale" individual information).

To argue about privacy, one should of course define what it means. This is usually done by defining what privacy breaches (or compromises) are (i.e., what should an attacker

achieve in order to win the privacy game). In most previous work on auditing, a 'classical' notion of compromise is used, i.e., a compromise occurs if there exists an index i such that x_i is learned in full. In other words, in all datasets consistent with the queries and answers, x_i is the same. While very appealing due to its combinatorial nature, we note that this definition of privacy is usually not quite satisfying. In particular, it does not consider cases where a lot of information is leaked about an entry x_i (and hence privacy is intuitively breached), as long as this information does not completely pinpoint x_i. One of the goals of this work is to extend the construction of auditing algorithms to guarantee more realistic notions of privacy, such as excluding *partial compromise*.

The first study of the auditing problem that we are aware of is due to Dobkin, Jones, and Lipton [9]. Given a private dataset X, a sequence of queries is said to compromise X if there exists an index i such that in all datasets consistent with the queries and answers, x_i is the same (as mentioned, we refer to this kind of compromise as classical compromise). The goal of this work was to design algorithms that never allow a sequence of queries that compromises the data, regardless of the actual data. Essentially, this formulation never requires a denial. We call this type of auditing *query monitoring* or, simply *monitoring*. In terms of utility, monitoring may be too restrictive as it may disallow queries that, in the context of the underlying data set, do not breach privacy. One may try to answer this concern by constructing auditing algorithms where every query is checked with respect to the data set, and a denial occurs only when an 'unsafe' query occurs.

A related variant of the auditing problem is the *offline auditing problem*: given a collection of queries and true answers to each query, determine if a compromise has already happened, e.g., if some x_i can be uniquely determined. Offline algorithms are used in applications where the goal is to "catch" a privacy breach ex post facto.

It is natural to ask the following question: *Can an offline auditing algorithm directly solve the online auditing problem?* In the traditional algorithms literature, an offline algorithm can always be used to solve the online problem – the only penalty is in the efficiency of the resulting algorithm. To clarify the question for the auditing context, if we want to determine whether to answer q_t, can we consult the dataset for the true answer a_t and then run an offline algorithm to determine if providing a_t would lead to a compromise?

Somewhat surprisingly, we answer this question negatively. The main reason is that denials leak information. As a simple example, suppose that the underlying data set is real-valued and that a query is denied only if some value is fully compromised. Suppose that the attacker poses the first query $\text{sum}(x_1, x_2, x_3)$ and the auditor answers 15. Suppose also that the attacker then poses the second query $\max(x_1, x_2, x_3)$ and the auditor denies the answer. The denial tells the attacker that if the true answer to the second query were given then some value could be uniquely determined. Note that $\max(x_1, x_2, x_3) \not< 5$ since then the sum could not be 15. Further, if $\max(x_1, x_2, x_3) > 5$ then the query would not have been denied since no value could be uniquely determined. Consequently, $\max(x_1, x_2, x_3) = 5$ and the attacker learns that $x_1 = x_2 = x_3 = 5$ — a privacy breach of all three entries. The issue here is that query denials reduce the space of possible consistent solutions, and this reduction is not explicitly accounted for in existing offline auditing algorithms.

In fact, denials could leak information even if we could pose only SQL-type queries to a table containing multiple attributes (i.e., not explicitly specify the subset of the rows). For example, consider the three queries, sum, count and max (in that order) on the 'salary' attribute, all conditioned on the same predicate involving other attributes. Since the selection condition is the same, all the three queries act on the same subset of tuples. Suppose that the first two queries are answered. Then the max query is denied whenever its value is exactly equal to the ratio of the sum and the count values (which happens when all the selected tuples have the same salary value). Hence the attacker learns the salary values of all the selected tuples when denial occurs.

To work around this problem, we introduce a new model called *simulatable auditing* where query denials provably do not leak information. Intuitively, an auditor is simulatable if an attacker, knowing the query-answer history, could make the same decision as the simulatable auditor regarding a newly posed query. Hence, the decision to deny a query does not leak any information beyond what is already known to the attacker.

1.1 Classical vs. Probabilistic Compromise

As we already mentioned, the classical definition of compromise has been extensively studied in prior work. This definition is conceptually simple, has an appealing combinatorial structure, and can serve as a starting point for evaluating solution ideas for privacy. However, as has been noted by many others, e.g., [2, 15, 6, 18], this definition is inadequate in many real contexts. For example, if an attacker can deduce that a private data element x_i falls in a tiny interval, then the classical definition of privacy is not violated unless x_i can be uniquely determined. While some have proposed a privacy definition where each x_i can only be deduced to lie in a sufficiently large interval [18], note that the distribution of the values in the interval matters. For example, ensuring that age lies in an interval of length 50 when the user can deduce that age is between $[-49, 1]$ essentially does not preserve privacy.

In order to extend the discussion on auditors to more realistic partial compromise notions of privacy, we introduce a new privacy definition. Our privacy definition assumes that there exists an underlying probability distribution from which the data is drawn. This is a fairly natural assumption since most attributes like age, salary, etc. have a known probability distribution. The essence of the definition is that for each data element x_i and small-sized interval I, the auditor ensures that the prior probability that x_i falls in the interval I is about the same as the posterior probability that x_i falls in I given the queries and answers. This definition overcomes some of the aforementioned problems with classical compromise.

With this notion of privacy, we demonstrate a simulatable

auditor for sum queries. The new auditing algorithm computes posterior probabilities by utilizing existing randomized algorithms for estimating the volume of a convex polytope, e.g., [11, 19, 16, 20]. To guarantee simulatability, we make sure that the auditing algorithm does not access the data set while deciding whether to allow the newly posed query q_t (in particular, it does not compute the true answer to q_t). Instead, the auditor draws many data sets according to the underlying distribution, assuming the previous queries and answers, then computes an expected answer a'_t and checks whether revealing it would breach privacy for each of the randomly generated data sets. If for most answers the data set is not compromised then the query is answered, and otherwise the query is denied.

1.2 Overview of the paper

The rest of this paper is organized as follows. In Section 2 we discuss related work on auditing. A more general overview can be found in [1]. In Section 3 we give several motivating examples where the application of offline auditing algorithms to solve the online auditing problem leads to massive breaches of the dataset since query denials leak information. We then introduce simulatable auditing in Section 4 and prove that max queries can be audited under this definition of auditing and the classical definition of privacy in Section 5. Then in Section 6 we introduce our new definition of privacy and finally in Section 7 we prove that sum queries can be audited under both this definition of privacy and simulatable auditing.

2. RELATED WORK

2.1 Online Auditing

As the initial motivation for work on auditing involves the online auditing problem, we begin with known online auditing results. The earliest are the query monitoring results due to Dobkin, Jones, and Lipton [9] and Reiss [21] for the online sum auditing problem, where the answer to a query Q is $\sum_{i \in Q} x_i$. With queries of size at least k elements, each pair overlap in at most r elements, they showed that any data set can be compromised in $(2k - (\ell + 1))/r$ queries by an attacker that knows ℓ values a priori. For fixed k, r and ℓ, if the auditor denies answers to query $(2k-(\ell+1))/r$ and on, then the data set is definitely not compromised. Here the monitor logs all the queries and disallows Q_i if $|Q_i| < k$, or for some query $t < i$, $|Q_i \cap Q_t| > r$, or if $i \geq (2k-(\ell+1))/r$.[1] These results completely ignore the answers to the queries. On the one hand, we will see later that this is desirable in that the auditor is simulatable – the decision itself cannot leak any information about the data set. On the other hand, we will see that because the answers are ignored, sometimes only short query sequences are permitted, whereas longer ones could have been permitted without resulting in a compromise.

In addition, Chin [3] considered the online sum, max, and mixed sum/max auditing problems. Both the online sum and the online max problem were shown to have efficient auditing algorithms. But the mixed sum/max problem was shown to be NP-hard.

[1]Note that this is a fairly negative result. For example, if $k = n/c$ for some constant c and $r = 1$, then the auditor would have to shut off access to the data after only a constant number of queries, since there are only about c queries where no two overlap in more than one element.

2.2 Offline Auditing

In the *offline all* problem, the auditor is given an offline set of queries $q_1, \ldots, q_t$ and true answers $a_1, \ldots, a_t$ and must determine if a breach of privacy has occurred. In most of the works, a privacy breach is defined to occur whenever some element in the data set can be uniquely determined. If only sum or only max queries are posed, then polynomial-time auditing algorithms are known to exist [4]. However, when sum and max queries are intermingled, then determining whether a specific value can be uniquely determined is known to be NP-hard [3].

Kam and Ullman [15] consider auditing subcube queries which take the form of a sequence of 0s, 1s, and *s where the *s represent "don't cares". For example, the query 10**1* matches all entries with a 1 in the first position, 0 in the second, 1 in the fifth and anything else in the remaining positions. Assuming sum queries over the subcubes, they demonstrate when compromise can occur depending on the number of *s in the queries and also depending on the range of input data values.

Kleinberg, Papadimitriou and Raghavan [17] consider the Boolean auditing problem, where the data elements are Boolean, and the queries are sum queries over the integers. They showed that it is coNP-hard to decide whether a set of queries uniquely determines one of the data elements, and suggest an approximate auditor (that may refuse to answer a query even when the answer would not compromise privacy) that could be efficiently implemented. Polynomial-time max auditors are also given.

In the *offline maximum* version of the auditing problem, the auditor is given a set of queries $q_1, \ldots, q_t$, and must identify a maximum-sized subset of queries such that all can be answered simultaneously without breaching privacy. Chin [3] proved that the offline maximum sum query auditing problem is NP-hard, as is the offline maximum max query auditing problem.

3. MOTIVATING EXAMPLES

The offline problem is fundamentally different from the online problem. In particular, existing offline algorithms cannot be used to solve the online problem because offline algorithms assume that all queries were answered. If we invoke an offline algorithm with only the previously truly answered queries, then we next illustrate how an attacker may exploit auditor responses to breach privacy since query denials leak information. In all cases, breaches are with respect to the same privacy definition for which the offline auditor was designed.

We assume that the attacker knows the auditor's algorithm for deciding denials. This is a standard assumption employed in the cryptography community known as *Kerkhoff's Principle* – despite the fact that the actual privacy-preserving auditing algorithm is public, an attacker should still not be able to breach privacy.

We first demonstrate how to use a max auditor to breach

the privacy of a significant fraction of a data set. The same attack works also for sum/max auditors. Then we demonstrate how a Boolean auditor may be adversarially used to learn an entire data set. The same attack works also for the (tractable) approximate auditing version of Boolean auditing in [17].

3.1 Max Auditing Breach

Here each query specifies a subset of the data set entries, and its answer is the maximum element in the queried data set. For simplicity of presentation, we assume the data set consists of n distinct elements. [17, 3] gave a polynomial time offline max auditing algorithm. We show how to use the offline max auditor to reveal 1/8 of the entries in the dataset when applied in an online fashion. The max auditor decides, given the allowed queries in $q_1, \ldots, q_t$ and their answers whether they breach privacy or not. If privacy is breached, the auditor denies q_t.

We arrange the data in $n/4$ disjoint 4-tuples and consider each 4-tuple of entries independently. In each of these 4-tuples we will use the offline auditor to learn one of the four entries, with success probability 1/2. Hence, on average we will learn 1/8 of the entries. Let a, b, c, d be the entries in a 4-tuple. We first issue the query $max(a, b, c, d)$. This query is never denied, let m be its answer. For the second query, we drop at random one of the four entries, and query the maximum of the other three. If this query is denied, then we learn that the dropped entry equals m. Otherwise, we let m' $(= m)$ be the answer and drop another random element and query the maximum of the remaining two. If this query is denied, we learn that the second dropped entry equals m'. If the query is allowed, we continue with the next 4-tuple.

Note that whenever the second or third queries are denied, the above procedure succeeds in revealing one of the four entries. If all the elements are distinct, then the probability we succeed in choosing the maximum value in the two random drops is 2/4. Consequently, on average, the procedure reveals 1/8 of the data.

3.2 Boolean Auditing Breach

Dinur and Nissim [8] demonstrated how an offline auditing algorithm can be used to mount an attack on a Boolean data set. We give a (slightly modified) version of that attack, that allows an attacker to learn the entire database.

Consider a data set of Boolean entries, with sum queries. Let $d = x_1, \ldots, x_n$ be a random permutation of the original data set. As in [8], we show how to use the auditor to learn d or its complement. We then show that in a typical ('non-degenerate') database one more query is enough for learning the entire data set.

We make the queries $q_i = sum(x_i, x_{i+1})$ for $i = 1, \ldots, n-1$. Note that $x_i \neq x_{i+1}$ iff the query q_i is allowed[2]. After $n-1$ queries we learn two candidates for the data set, namely, d itself and the bit-wise complement of d. To learn which of the candidates is the right one, we choose the last query to consist of three distinct entries x_i, x_j, x_k of nonequal values, such that for each entry both queries touching it were denied. The query $sum(x_i, x_j, x_k)$ is never denied[3]. The answer to the last query (1 or 2) resolves which of the two candidates for d is the right one.

To conclude the description, we note that if both the number of zeros and the number of ones in the database are, say, $\omega(n^{2/3})$ then indices i, j, k as above exist with high probability. Note that a typical database would satisfy this condition.

Our attack on the Boolean auditor would be typically successful even without the initial random permutation set. In that case, the attacker only uses 'one-dimensional' (or range) queries, and hence a polynomial-time auditor exists (see [17]).

4. SIMULATABLE AUDITING

Evidently these examples show that denials leak information. The situation is further worsened by the failure of the auditors to formulate this information leakage and to take it into account along with the answers to allowed queries. Intuitively, denials leak because users can ask *why* a query was denied, and the reason is in the data. If the decision to allow or deny a query depends on the actual data, it reduces the set of possible consistent solutions for the underlying data.

A naive solution to the leakage problem is to deny whenever the offline algorithm would, and to also randomly deny queries that would normally be answered. While this solution seems appealing, it has its own problems. Most importantly, although it may be that denials leak less information, leakage is not generally prevented. Furthermore, the auditing algorithm would need to remember which queries were randomly denied, since otherwise an attacker could repeatedly pose the same query until it was answered. A difficulty then arises in determining whether two queries are equivalent. The computational hardness of this problem depends on the query language, and may be intractable, or even undecidable.

To work around the leakage problem, we make use of the simulation paradigm which is of vast usage in cryptography (starting with the definition of semantic security [14]). The idea is the following: The reason that denials leak information is because the auditor uses information not available to the attacker (the answer to the newly posed query), furthermore resulting in a computation the attacker could not perform by himself. A successful attacker capitalizes on this leakage to gain information. We introduce a notion of auditing where the attacker *provably* cannot gain any new information from the auditor's decision, so that denials do not leak information. This is formalized by requiring that the attacker is able to simulate or mimic the auditor's decisions. In such a case, because the attacker can equivalently decide if a query would be denied, denials do not leak information.

4.1 A Perspective on Auditing

[2]When $x_i \neq x_{i+1}$ the query is allowed as the answer $x_1 + x_{i+1} = 1$ does not reveal which of the values is 1 and which is 0. When $x_i = x_{i+1}$ the query is denied, as its answer (0 or 2) would reveal both x_i and x_{i+1}.

[3]Here we use the fact that denials are not taken into account.

We cast related work on auditing based on two important dimensions: utility and privacy. It is interesting to note the relationship between the information an auditor uses and its utility – the more information used, the longer query sequences the auditor can decide to answer. That is because an informed auditor need not deny queries that do not actually put privacy at risk. On the other hand, as we saw in Section 3, if the auditor uses too much information, some of this information my be leaked, and privacy may be adversely affected.

The oldest work on auditing includes methods that simply consider the size of queries and the size of the intersection between pairs of queries [9]. Subsequently, the contents of queries was considered (such as the elementary row and column matrix operations suggested in [3, 4]). We call these *monitoring* methods. Query monitoring only makes requirements about the queries, and is oblivious of the actual data entries. To emphasize this, we note that to decide whether a query q_t is allowed, the monitoring algorithm takes as input the query q_t and the previously allowed queries $q_1, \ldots, q_{t-1}$, but ignores the answers to all these queries. This obliviousness of the query answers immediately implies the safety of the auditing algorithm, in the sense that query denials can not leak information. In fact, a user need not even communicate with the database to check which queries would be allowed, and hence these auditors are simulatable.

More recent work on auditing focused on offline auditing methods. These auditors take as input the queries $q_1, \ldots, q_t$ and their answers $a_1, \ldots, a_t$. While this approach yields more utility, we saw in Section 3 that these auditors are not generally safe for online auditing. In particular, these auditors are not generally simulatable.

We now introduce a 'middle point' that we call *simulatable auditing*. Simulatable auditors use all the queries $q_1, \ldots, q_t$ and the answers to only the previous queries $a_1, \ldots, a_{t-1}$ to decide whether to answer or deny the newly posed query q_t. We will construct simulatable auditors that guarantee 'classical' privacy. We will also consider a variant of this 'middle point', where the auditing algorithm (as well as the attacker) has access to the underlying probability distribution[4]. With respect to this variant we will construct simulatable auditors that guarantee a partial compromise notion of privacy.

4.2 A Formal Definition of Simulatable Auditing

We say that an auditor is simulatable if the decision to deny or give an answer is independent of the actual data set $\{x_1, \ldots, x_n\}$ and the real answer a_t[5].

A nice property of simulatable auditors, that may simplify their design, is that the auditor's response to denied queries does not convey any new information to the attacker (beyond what is already known given the answers to the previous queries). Hence denied queries need not be taken into account in the auditor's decision. We thus assume without loss of generality that $q_1, \ldots, q_{t-1}$ were all answered.

DEFINITION 4.1. *An auditor is* simulatable *if the decision to deny or give an answer to the query q_t is made based exclusively on $q_1, \ldots, q_t$, and $a_1, \ldots, a_{t-1}$ (and not a_t and not the dataset $X = \{x_1, \ldots, x_n\}$) and possibly also the underlying probability distribution $\mathcal{D}$ from which the data was drawn.*

Simulatable auditors improve upon monitoring methods in that longer query sequences can now be answered (an example is given in Section 5.3) and improve upon offline algorithms since denials do not leak information.

4.3 Constructing Simulatable Auditors

We next propose a general approach for constructing simulatable auditors that is useful for understanding our results and may also prove valuable for studying other types of queries.

The general approach works as follows: Choose a set of consistent answers to the last query q_t. For each of these answers, check if privacy is compromised. If compromise occurs for too many of the consistent answers, the query is denied. Otherwise, it is allowed. In the case of classical compromise for max simulatable auditing, we deterministically construct a small set of answers to the last query q_t so that if any one leads to compromise, then we deny the answer and otherwise we give the true answer. In the case of probabilistic compromise for sum queries, we randomly generate many consistent answers and if sufficiently many lead to compromise, then we deny the query and otherwise we answer the query.

5. SIMULATABLE AUDITING ALGORITHMS, CLASSICAL COMPROMISE

We next construct (tractable) simulatable auditors. We first describe how sum queries can be audited under the classical definition of privacy and then we describe how max queries can be audited under the same definition.

5.1 Sum Queries

Observe that existing sum auditing algorithms are already simulatable [3]. In these algorithms each query is expressed as a row in a matrix with a 1 wherever there is an index in the query and a 0 otherwise. If the matrix can be reduced to a form where there is a row with one 1 and the rest 0s then some value has been compromised. Such a transformation of the original matrix can be performed via elementary row and column operations. The reason this auditor is simulatable is that the answers to the queries are ignored when the matrix is transformed.

5.2 A Max Simulatable Auditor

We provide a simulatable auditor for the problem of auditing MAX queries over real-valued data. The data consists of a set of n values, $\{x_1, x_2, \ldots, x_n\}$ and the queries $q_1, q_2, \ldots$ are subsets of $\{1, 2, \ldots, n\}$. The answer corresponding to the query q_i is $m_i = \max\{x_j | j \in q_i\}$. Given

[4]This model was hinted at informally in [8] and the following work [10].

[5]For simplicity we require total independence (given what is already known to the attacker). This requirement could be relaxed, e.g. to a computational notion of independence.

a set of queries $q_1, \ldots, q_{t-1}$ and the corresponding answers $m_1, \ldots, m_{t-1}$ and the current query q_t, the simulatable auditor denies q_t if and only if there exists an answer m_t, consistent with $m_1, \ldots, m_{t-1}$, such that the answer helps to uniquely determine some element x_j.

In other words, the *max simulatable auditing problem* is the following: *Given previous queries, previous answers, and the current query, q_t, determine if for all answers to q_t consistent with the previous history, no value x_j can be uniquely determined.* Since the decision to deny or answer the current query is independent of the real answer m_t, we should decide to answer q_t only if compromise does not occur for all consistent answers to q_t (as the real answer could be any of these). Conversely if compromise does not occur for all consistent answers to q_t, it is safe to answer q_t.

We now return to the max auditing breach example of Section 3.1 and describe how a simulatable auditor would work. The first query $max(a, b, c, d)$ is always answered since there is no answer, m_1 for which a value is uniquely determined. Let the second query be $max(a, b, d)$. This would be always denied since c is determined to be equal to m_1 whenever the answer $m_2 < m_1$. In general, under the classical privacy definition, the simulatable auditor has to deny the current query even if compromise occurs for only one consistent answer to q_t and thus could end up denying a lot of queries. This issue is addressed by our probabilistic definition of privacy in Section 6.

Next we will prove the following theorem.

THEOREM 5.1. *There is a max simulatable auditing algorithm that runs in time $O(t\sum_{i=1}^{t}|q_i|)$ where t is the number of queries.*

We now discuss how we obtain an algorithm for max simulatable auditing. Clearly, it is impossible to check for all possible answers m_t in $(-\infty, +\infty)$. We show that it is sufficient to test only a finite number of points. Let $q'_1, \ldots, q'_l$ be the previous queries that intersect with the current query q_t, ordered according to the corresponding answers, $m'_1 \leq \ldots \leq m'_l$. Let $m'_{lb} = m'_1 - 1$ and $m'_{ub} = m'_l + 1$ be the bounding values. Our algorithm checks for only $2l+1$ values: the bounding values, the above l answers, and the mid-points of the intervals determined by them.

Algorithm 1 MAX SIMULATABLE AUDITOR

1: **for** $m_t \in \{m'_{lb}, m'_1, \frac{m'_1+m'_2}{2}, m'_2, \frac{m'_2+m'_3}{2}, m'_3, \ldots, m'_{l-1}, \frac{m'_{l-1}+m'_l}{2}, m'_l, m'_{ub}\}$ **do**
2: **if** m_t is consistent with the previous answers $m_1, \ldots, m_{t-1}$ AND if $\exists 1 \leq j \leq n \quad x_j$ is uniquely determined {using [17]} **then**
3: Output "Deny" and **return**
4: **end if**
5: **end for**
6: Output "Answer" and **return**

Next we address the following questions: How we can tell if a value is uniquely determined? How can we tell if an answer for the current query is consistent with previous queries and answers? Why is it sufficient to check for just the answers to the previous queries that intersect with q_t and the mid-points of the intervals determined by them?

Given a set of queries and answers, the upper bound, μ_j for an element x_j is defined to be the minimum over the answers to the queries containing x_j, i.e., $\mu_j = \min\{m_k | j \in q_k\}$. In other words, μ_j is the best possible upper bound for x_j that can be obtained from the answers to the queries. We say that j is an *extreme* element for the query set q_k if $j \in q_k$ and $\mu_j = m_k$. This means that the upper bound for x_j is realized by the query set q_k, i.e., the answer to every other query containing x_j is greater than or equal to m_k. The upper bounds of all elements as well as the extreme elements of all the query sets can be computed in $O(\sum_{i=0}^{t}|q_i|)$ time (which is linear in the input size). In [17], it was shown that a value x_j is uniquely determined if and only if there exists a query set q_k for which j is the only *extreme* element. Hence, for a given value of m_t, we can check if $\exists 1 \leq j \leq n \quad x_j$ is uniquely determined.

LEMMA 5.1. *For a given value of m_t, inconsistency occurs if and only if some query set has no extreme element.*

PROOF. Suppose that some query set q_k has no extreme element. This means that the upper bound of every element in q_k is less than m_k. This cannot happen since some element has to equal m_k. Formally, $\forall j \in q_k, \quad x_j \leq \mu_j < m_k$ which is a contradiction.

Conversely, if every query set has at least one extreme element, setting $x_j = \mu_j$ for $1 \leq j \leq n$ is consistent with all the answers. This is because, for any set q_k with s as an extreme element, $x_s = m_k$ and $\forall j \in q_k, \quad x_j \leq m_k$. □

In fact, it is enough to check the condition in the lemma for q_t and the query sets intersecting it (instead of all the query sets).

LEMMA 5.2. *For $1 \leq j \leq n$, x_j is uniquely determined for some value of m_t in (m'_s, m'_{s+1}) if and only if x_j is uniquely determined when $m_t = \frac{m'_s+m'_{s+1}}{2}$.*

PROOF. Observe that revealing m_t can only affect elements in q_t and the queries intersecting it. This is because revealing m_t can possibly lower the upper bounds of elements in q_t, thereby possibly making some element in q_t or the queries intersecting it the only extreme element of that query set. Revealing m_t does not change the upper bound of any element in a query set disjoint with q_t and hence does not affect elements in such sets.

Hence it is enough to consider $j \in q_t \cup q'_1 \cup \ldots \cup q'_l$. We consider the following cases:

- $q_t = \{j\}$: Clearly x_j is breached irrespective of the value of m_t.
- j is the only extreme element of q_t and $|q_t| > 1$: Suppose that x_j is uniquely determined for some value

of m_t in (m'_s, m'_{s+1}). This means that each element indexed in $q_t \setminus \{j\}$ had an upper bound $< m_t$ and hence $\leq m'_s$ (since an upper bound can only be one of the answers given so far). Since this holds even for $m_t = \frac{m'_s + m'_{s+1}}{2}$, j is still the only extreme element of q_t and hence x_j is still uniquely determined. A similar argument applies for the converse.

- j is the only extreme element of q'_k for some k: Suppose that x_j is uniquely determined for some value of m_t in (m'_s, m'_{s+1}). This means that $m_t < m'_k$ (and hence $m'_{s+1} \leq m'_k$) and revealing m_t reduced the upper bound of some element indexed in $q'_k \setminus \{j\}$. This would be the case even when $m_t = \frac{m'_s + m'_{s+1}}{2}$. The converse can be argued similarly.

To complete the proof, we will show that values of m_t in (m'_s, m'_{s+1}) are either all consistent or all inconsistent (so that our algorithm preserves consistency when considering only the mid-point value). Suppose that some value of m_t in (m'_s, m'_{s+1}) results in inconsistency. Then, by Lemma 5.1, some query set q_α has no extreme element. We consider two cases:

- $q_\alpha = q_t$: This means that the upper bound of every element in q_t was $< m_t$ and hence $\leq m'_s$. This would be the case even for any value of m_t in (m'_s, m'_{s+1}).

- q_α intersects with q_t: This means that $m_t < m_\alpha$ and the extreme element(s) of q_α became no longer extreme for q_α by obtaining a lower upper bound due to m_t. This would be the case even for any value of m_t in (m'_s, m'_{s+1}).

□

Thus it suffices to check for $m_t = \frac{m'_s + m'_{s+1}}{2}$ $\forall 1 \leq s < l$ together with $m_t = m'_s$ $\forall 1 \leq s \leq l$ and also representative points, $(m'_1 - 1)$ in $(-\infty, m'_1)$ and $(m'_l + 1)$ in (m'_l, ∞). Note that inconsistency for a representative point is equivalent to inconsistency for the corresponding interval.

As noted earlier, the upper bounds of all elements as well as the extreme elements of all the query sets and hence each iteration of the for loop in Algorithm 1 can be computed in $O(\sum_{i=0}^{t} |q_i|)$ time (which is linear in the input size). As the number of iterations is $2l + 1 \leq 2t$, the running time of the algorithm is $O(t \sum_{i=0}^{t} |q_i|)$, proving Theorem 5.1.

We remark that both conditions in step 2 of Algorithm 1 exhibit monotonicity with respect to the value of m_t. For example, if $m_t = \alpha$ results in inconsistency, so does any $m_t < \alpha$. By exploiting this observation, the running time of the algorithm can be improved to $O((\log t) \sum_{i=0}^{t} |q_i|)$. The details can be found in the full version of the paper.

5.3 Utility of Max Simulatable Auditor vs. Monitoring

While both simulatable auditors and monitoring methods are safe, simulatable auditors potentially have greater utility, as shown by the following example.

Consider the problem of auditing max queries on a database containing 5 elements. We will consider three queries and two possible sets of answers to the queries. We will demonstrate that the simulatable auditor answers the third query in the first case and denies it in the second case while a query monitor (which makes the decisions based only on the query sets) has to deny the third query in both cases. Let the query sets be $q_1 = \{1, 2, 3, 4, 5\}$, $q_2 = \{1, 2, 3\}$, $q_3 = \{3, 4\}$ in that order. Suppose that the first query is answered as $m_1 = 10$. We consider two scenarios based on m_2. (1) If $m_2 = 10$, then every query set has at least two extreme elements, irrespective of the value of m_3. Hence the simulatable auditor will answer the third query. (2) Suppose $m_2 = 8$. Whenever $m_3 < 10$, 5 is the only extreme element for S_1 so that $x_5 = 10$ is determined. Hence it is not safe to answer the third query.

While the simulatable auditor provides an answer q_3 in the first scenario, a monitor would have to deny q_3 in both, as its decision is oblivious of the answers to the first two queries.

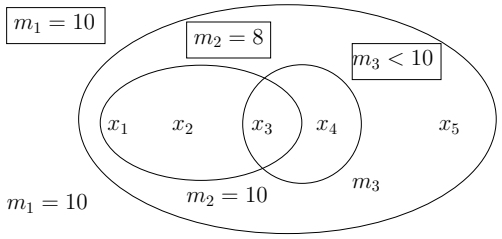

Figure 1: Max simulatable auditor more useful than max query restriction auditor. The values within the boxes correspond to the second scenario.

6. PROBABILISTIC COMPROMISE

We next describe a definition of privacy that arises from some of the previously noted limitations of classical compromise. On the one hand, classical compromise is a weak definition since if a private value can be deduced to lie in a tiny interval – or even a large interval where the distribution is heavily skewed towards a particular value – it is not considered a privacy breach. On the other hand, classical compromise is a strong definition since there are situations where no query would ever be answered. This problem has been previously noted [17]. For example, if the dataset contains items known to fall in a bounded range, e.g., Age, then no sum query would ever be answered. For instance, the query $\text{sum}(x_1, \ldots, x_n)$, would not be answered since there exists a dataset, e.g., $x_i = 0$ for all i where a value, in fact all values, can be uniquely determined.

To work around these issues, we propose a definition of privacy that bounds the change in the ratio of the posterior probability that a value x_i lies in an interval I given the queries and answers to the prior probability that $x_i \in I$. This definition is related to definitions suggested in the output perturbation literature including [13, 10].

For an arbitrary data set $X = \{x_1, \ldots, x_n\} \in [0, 1]^n$ and

queries $q_j = (Q_j, f_j)$, for $j = 1, \ldots t$, let a_j be the answers to q_j according to X. Define a Boolean predicate that evaluates to 1 if the data entry x_i is safe with respect to an interval $I \subseteq [0,1]$ (i.e., whether given the answers one's confidence in $x_i \in I$ changes significantly):

$$S_{\lambda,i,I}(q_1, \ldots, q_t, a_1, \ldots, a_t) =$$
$$\begin{cases} 1 & \text{if } (1-\lambda) \leq \frac{\Pr_{\mathcal{D}}(x_i \in I | \wedge_{j=1}^{t} f_j(Q_j) = a_j)}{\Pr_{\mathcal{D}}(x_i \in I)} \leq 1/(1-\lambda) \\ 0 & \text{otherwise} \end{cases}$$

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

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

i.e., $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 1$ iff $q_1, \ldots, q_t, a_1, \ldots, a_t$ is λ-safe for all entries, for all intervals in $\mathcal{I}$.

We now turn to our privacy definition. Let $X = \{x_1, \ldots, x_n\}$ be drawn according to a distribution $\mathcal{D}$ on $[0,1]^n$, and consider the following (λ, α, T)-*privacy game* between an attacker and an auditor, where in each round t (for up to T rounds):

1. The attacker (adaptively) poses a query $q_t = (Q_t, f_t)$.
2. The auditor decides whether to allow q_t (and then reply with $a_t = f_t(Q_t)$), or $a_t =$ "denied".
3. The attacker wins if $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$.

DEFINITION 6.1. *We say that an auditor is* $(\lambda, \delta, \alpha, T)$-private *if for any attacker* $\mathcal{A}$

$$\Pr[\mathcal{A} \textit{ wins the } (\lambda, \alpha, T)\textit{-privacy game}] \leq \delta .$$

The probability is taken over the randomness in the distribution $\mathcal{D}$ *and the coin tosses of the auditor and attacker.*

Combining Definitions 4.1 and 6.1, we have our new model of simulatable auditing. In other words, we seek auditors that are simulatable and $(\lambda, \delta, \alpha, T)$-private.

Note that, on the one hand, the simulatable definition prevents the auditor from using a_t in deciding whether to answer or deny the query. On the other hand, the privacy definition requires that regardless of what a_t was, with high probability, each data value x_i is still safe (as defined by S_i). Consequently, it is important that the current query q_t be used in deciding whether to deny or answer. Upon making a decision, our auditors ignores the true value a_t and instead make guesses about the value of a_t obtained by randomly sampling data sets according to the distribution.

7. SIMULATABLE FRACTIONAL SUM AUDITING

In this section we consider the problem of auditing sum queries, i.e., where each query is of the form $\text{sum}(Q_j)$ and Q_j is a subset of dimensions. We assume that the data falls in the unit cube $[0,1]^n$, although the techniques can be applied to any bounded real-valued domain. We also assume that the data is drawn from a uniform distribution over $[0,1]^n$.

7.1 Estimating the Predicate $S()$

We begin by considering the situation when we have the answer to the last query a_t. Our auditors cannot use a_t, and we will show how to get around this assumption. Given a_t and all previous queries and answers, Algorithm 2 checks whether privacy has been breached. The idea here is to express the ratio of probabilities in $S_{\lambda,i,I}$ as the ratio of the volumes of two convex polytopes. We then make use of randomized, polynomial-time algorithms for estimating the volume of a convex body (see for example [11, 19, 16, 20]). Given a convex polytope P with volume $\text{Vol}(P)$, these algorithms output a value V such that $(1-\epsilon)\text{Vol}(P) < V < \text{Vol}(P)/(1-\epsilon)$ with probability at least $1-\eta$. The running time of the algorithm is polynomial in n, $\frac{1}{\epsilon}$, and $\log(1/\eta)$. We call such an algorithm $\text{Vol}_{\epsilon,\eta}$ – this algorithm is employed in steps 5 and 7 of Algorithm 2. Note that $\text{Vol}(P)$ without any subscripts is the true volume of P.

Algorithm 2 SAFE

1: Input: Queries and Answers q_j and a_j for $j = 1, \ldots, t$, and parameters λ, η, α, n.
2: Let safe=*true*, $\epsilon = \lambda/6$.
3: **for** each x_i and for each interval I in $\mathcal{I}$ **do**
4: Let P be the (possibly less than n-dimensional) polytope defined by $(\cap_{j=1}^{t}(sum(Q_j) = a_j)) \cap [0,1]^n$
5: $V = \text{Vol}_{\epsilon,\eta}(P)$
6: Let $P_{i,I}$ be the polytope defined by $P \cap (x_i \in I)$
7: $V_{i,I} = \text{Vol}_{\epsilon,\eta}(P_{i,I})$
8: **if** $\left(\frac{\alpha \cdot V_{i,I}}{V} < (1-4\epsilon)\right)$ OR $\left(\frac{\alpha \cdot V_{i,I}}{V} > 1/(1-4\epsilon)\right)$ **then**
9: Let safe=*false*
10: **end if**
11: **end for**
12: Return safe.

LEMMA 7.1. *1. If* $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ *then Algorithm* SAFE *returns 0 with probability at least* $1 - 2\eta$.

2. If $S_{\lambda/3}(q_1, \ldots, q_t, a_1, \ldots, a_t) = 1$ *then Algorithm* SAFE *returns 1 with probability at least* $1 - 2n\alpha\eta$.

PROOF. We first note that

$$\frac{\Pr_{\mathcal{D}}(x_i \in I | \wedge_{j=1}^{t} (sum(Q_j) = a_j))}{\Pr_{\mathcal{D}}(x_i \in I)} =$$
$$= \frac{\Pr_{\mathcal{D}}(x_i \in I \wedge \bigwedge_{j=1}^{t}(sum(Q_j) = a_j))}{\Pr_{\mathcal{D}}(x_i \in I) \Pr_{\mathcal{D}}(\bigwedge_{j=1}^{t}(sum(Q_j) = a_j))}$$
$$= \alpha \cdot \frac{\text{Vol}(P_{i,I})}{\text{Vol}(P)}$$

where P and $P_{i,I}$ are as defined in steps 4 and 6 of Algorithm SAFE, and where $\text{Vol}(P)$ denotes the true volume of the polytope P in the subspace whose dimension equals the dimension of P. (The last equality is due to $\mathcal{D}$ being uniform.)

Using the guarantee on the volume estimation algorithm, we get that with probability at least $1 - 2\eta$,

$$\frac{V_{i,I}}{V} \leq \frac{\text{Vol}(P_{i,I})}{\text{Vol}(P)} \cdot \frac{1}{(1-\epsilon)^2} \leq \frac{\text{Vol}(P_{i,I})}{\text{Vol}(P)} \cdot \frac{1}{1-2\epsilon} .$$

The last inequality follows noting that $(1-\epsilon)^2 = 1-2\epsilon+\epsilon^2 \geq 1-2\epsilon$. Similarly, $\frac{V_{i,I}}{V} \geq \frac{\mathrm{Vol}(P_{i,I})}{\mathrm{Vol}(P)} \cdot (1-2\epsilon)$.

We now prove the two parts of the claim:

1. If $\alpha \cdot \frac{\mathrm{Vol}(P_{i,I})}{\mathrm{Vol}(P)} < 1-\lambda$ then $\alpha \cdot \frac{V_{i,I}}{V} < (1-\lambda)/(1-2\epsilon)$, and by our choice of $\epsilon = \lambda/6$ we get $\alpha \cdot \frac{V_{i,I}}{V} < 1-4\epsilon$, hence Algorithm SAFE returns 0. The case $\alpha \cdot \frac{\mathrm{Vol}(P_{i,I})}{\mathrm{Vol}(P)} > 1/(1-\lambda)$ is treated similarly.
2. If $\alpha \cdot \frac{\mathrm{Vol}(P_{i,I})}{\mathrm{Vol}(P)} > 1-\lambda/3$ then $\alpha \cdot \frac{V_{i,I}}{V} > (1-\lambda/3)(1-2\epsilon) > 1-4\epsilon$. The case $\alpha \cdot \frac{\mathrm{Vol}(P_{i,I})}{\mathrm{Vol}(P)} < 1/(1-\lambda/3)$ is treated similarly. Using the union bound on all $i \in [n]$ and $I \in \mathcal{I}$ yields the proof.

□

The choice of $\lambda/3$ above is arbitrary, and (by picking ϵ to be small enough), one may prove the second part of the claim with any value $\lambda' < \lambda$. Furthermore, taking $\eta < 1/6n\alpha$, and using standard methods (repetition and majority vote) one can lower the error guarantees of Lemma 7.1 to be exponentially small. Neglecting this small error, we assume below that algorithm SAFE always returns 0 when $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ and always returns 1 when $S_{\lambda'}(q_1, \ldots, q_t, a_1, \ldots, a_t) = 1$, for some $\lambda' < \lambda$. From now on we also assume that algorithm SAFE has an additional parameter λ'.

7.2 Constructing the Simulatable Auditor

Our construction follows a paradigm that may prove useful in the construction of other simulatable auditors. We make use of randomized, polynomial-time algorithms for sampling from a conditional distribution. Such a sampling approach has previously been suggested as a useful tool for privacy [7, 12, 5]. In our case, we utilize algorithms for nearly uniformly sampling from a convex body, such as in [11]. Given the history of queries and answers $q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}$, and a new query q_t, the sampling procedure allows us to estimate the probability that answering q_t would breach privacy (the probability is taken over the distribution $\mathcal{D}$ conditioned on the queries and answers $q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}$). To estimate this probability, we sample a random database X' that is consistent with the answers already given, compute an answer a'_t according to X' and evaluate Algorithm SAFE on $q_1, \ldots, q_t, a_1, \ldots, a_{t-1}, a'_t$. The sampling is repeated to get a good enough estimate that the SAFE algorithm returns false. If the probability is less than $\approx \delta/T$, then, via the union bound, our privacy definition is satisfied.

Without loss of generality, we assume that the input $q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}$ contains only queries allowed by the auditor. As the auditor is simulatable, denials do not leak any information (and hence do not change the conditional probability on databases) beyond what the previously allowed queries already leak. For clarity of presentation in this extended abstract, we assume that the sampling is exactly uniform. For a data set X and query Q, let $\mathrm{sum}_X(Q) = \sum_{i \in Q} X(i)$.

Algorithm 3 FRACTIONAL SUM SIMULATABLE AUDITOR

1: Input: Data Set X, (allowed) queries and answers q_j and a_j for $j = 1, \ldots, t-1$, a new query q_t, and parameters $\lambda, \lambda', \alpha, n, \delta, T$.
2: Let P be the (possibly less than n-dimensional) polytope defined by $(\cap_{j=1}^{t-1}(sum(Q_j) = a_j)) \cap [0,1]^n$
3: **for** $O(\frac{T}{\delta} \log \frac{T}{\delta})$ times **do**
4: Sample a data set X' from P { using [11]}
5: Let $a' = sum_{X'}(Q_t)$
6: Evaluate algorithm SAFE on input $q_1, \ldots, q_t, a_1, \ldots, a_{t-1}, a'$ and parameters $\lambda, \lambda', \alpha, n$
7: **end for**
8: **if** the fraction of sampled data sets for which algorithm SAFE returned *false* is more than $\delta/2T$ **then**
9: Return "denied"
10: **else**
11: Return a_t, the true answer to $q_t = sum_X(Q_t)$
12: **end if**

THEOREM 7.1. *Algorithm 3 is a $(\lambda, \delta, \alpha, T)$-private.*

Proof Sketch: By Definition 6.1, the attacker wins the game in round t if he poses a query q_t for which $S_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ and the auditor does not deny q_t.

Consider first an auditor that allows every query. Given answers to the first $t-1$ queries, the true data set is distributed according to the distribution $\mathcal{D}$, conditioned on these answers. Given q_t (but not a_t), the probability the attacker wins hence equals

$$p_t = \Pr_{\mathcal{D}}\left[S_\lambda \begin{pmatrix} q_1, \ldots, q_t, \\ a_1, \ldots, a_{t-1}, \\ sum_{X'}(q_t) \end{pmatrix} = 0 \middle| q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1} \right]$$

where X' is a data set drawn uniformly from $\mathcal{D}$ conditioned on $q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}$. Note that since $a' = \mathrm{sum}_{X'}(Q_t)$ is precisely what the algorithm computes, the algorithm essentially estimates p_t via multiple draws of random data sets X' from $\mathcal{D}$, i.e., it estimates the true probability p_t by the sampled probability.

Our auditor, however, may deny answering q_t. In particular, when $p_t > \delta/T$ then by the Chernoff bound, the fraction computed in step 8 is expected to be higher 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 $p_t \leq \delta/T$, the attacker wins only if the query is allowed, and even then only with probability p_t. We get that in both cases the attacker wins with probability at most δ/T. By the union bound, the probability that the attacker wins any one of the T rounds is at most δ, as desired. □

While the auditor could prevent an attacker from winning the privacy game with probability 1 by simply denying every query, our auditing algorithm is actually more useful in that if a query is very safe then, with high probability, the auditor will provide the true answer to the query (part 2 of Lemma 7.1).

We remark that our simulatable auditing algorithm for fractional sum queries can be extended to any linear combination queries. This is because, as in the case of fractional sum auditing, $(\cap_{j=1}^{t}(\sum_{i=0}^{n} q_{ji}x_i = a_j)) \cap [0,1]^n$ defines a convex polytope where $q_{j1}, \ldots, q_{jn}$ are the coefficients of the linear combination query q_j.

8. CONCLUSIONS AND FUTURE DIRECTIONS

We uncovered the fundamental issue that query denials leak information. While existing online auditing algorithms do not explicitly account for query denials, we believe that future research must account for such leakage if privacy is to be ensured. We suggest one natural way to get around the leakage that is inspired by the simulation paradigm in cryptography – where the decision to deny can be equivalently decided by either the attacker or the auditor.

Next we introduced a new definition of privacy. While we believe that this definition overcomes some of the limitations discussed, there is certainly room for future work. The current definition does not ensure that the privacy of a group of individuals or any function of a group of individuals is kept private. Also, our model assumes that the dataset is static, but in practice data is inserted and deleted over time.

Our sum simulatable auditing algorithm demonstrates that a polynomial-time solution exists. But the volume computation algorithms that we invoke are still not practical – although they have been steadily improving over the years. In addition, it would be desirable to generalize the result from just the uniform distribution. Simulatable sum queries over Boolean data is an interesting avenue for further work, as is the study of other classes of queries like the k^{th} ranked element, variance, clusters and combinations of these.

9. REFERENCES

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

[2] L. Beck. A security machanism for statistical database. *ACM Trans. Database Syst.*, 5(3):316–3338, 1980.

[3] F. Chin. Security problems on inference control for sum, max, and min queries. *J. ACM*, 33(3):451–464, 1986.

[4] F. Chin and G. Ozsoyoglu. Auditing for secure statistical databases. In *Proceedings of the ACM '81 conference*, pages 53–59. ACM Press, 1981.

[5] M. Cryan and M. Dyer. A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 240–249. ACM Press, 2002.

[6] T. Dalenius. Towards a methodology for statistical disclosure control. *Sartryck ur Statistisk tidskrift*, 15:429–444, 1977.

[7] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Ann. Statist*, 26:363–397, 1998.

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

[9] D. Dobkin, A. Jones, and R. Lipton. Secure databases: protection against user influence. *ACM Trans. Database Syst.*, 4(1):97–106, 1979.

[10] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, 2004.

[11] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[12] Martin E. Dyer, Ravi Kannan, and John Mount. Sampling contingency tables. *Random Structures and Algorithms*, 10(4):487–506, 1997.

[13] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222. ACM Press, 2003.

[14] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual Symposium on Theory of Computing (STOC)*, pages 365–377, New York, 1982. ACM Press.

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

[16] R. Kannan, L. Lovasz, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11, 1997.

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

[18] Y. Li, L. Wang, X. Wang, and S. Jajodia. Auditing interval-based inference. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 553–567, 2002.

[19] L. Lovasz and M. Simonovits. Random walks in a convex body and an improved volume algorithm. *Random Structures and Algorithms*, 4:359–412, 1993.

[20] L. Lovasz and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. In *Proceedings 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 650–659, 2003.

[21] S. Reiss. Security in databases: A combinatorial study. *J. ACM*, 26(1):45–57, 1979.