Chapter 1

A SURVEY OF QUERY AUDITING TECHNIQUES FOR DATA PRIVACY

Shubha U. Nabar
Stanford University
sunabar@cs.stanford.edu

Krishnaram Kenthapadi
Microsoft Search Labs
krishnaram.kenthapadi@microsoft.com

Nina Mishra
University of Virginia
nmishra@cs.virginia.edu

Rajeev Motwani
Stanford University
rajeev@cs.stanford.edu

Introduction

This chapter is a survey of query auditing techniques for detecting and preventing disclosures in a database containing private data. Informally, auditing is the process of examining past actions to check whether they were in conformance with official policies. In the context of database systems with specific data disclosure policies, auditing is the process of examining queries that were answered in the past to determine whether answers to these queries could have been used by an individual to ascertain confidential information forbidden by the disclosure policies. Techniques used for *detecting* disclosures could potentially also be used or extended to *prevent* disclosures, and so in addition to the retroactive auditing mentioned above, researchers have also studied an online variant

of the auditing problem wherein the task of an online auditor is to *deny* queries that could potentially cause a breach of privacy.

Other common approaches to tackling the disclosure prevention problem include adding noise to the the data or otherwise perturbing the query results supplied to the user. However statisticians are generally averse to potential biases introduced by adding noise. One commonly stated reason is that the data collection process is already prone to biases and imperfections due to factors such as too few respondents, the cost of gathering data, and inaccurate answers provided by respondents. Since important decisions are made based on this data, they prefer to receive answers without additional noise. It is in this context that query restriction techniques become relevant in disclosure prevention. The work on offline (or retroactive) auditing has also similarly focused on the case where answers supplied to users are exact.

The main focus of this chapter will be on statistical databases with a single private attribute that only permit aggregate queries such as `sum`, `max`, `min` or `median` over this private attribute. An instructive example is a company database with *employee salary* as a private attribute. Or a set of medical records with a boolean private attribute indicating whether or not a patient was HIV-positive. We will first review the most commonly used notion of disclosure in the statistical database literature called *full disclosure* and then review algorithms and hardness results for offline auditing that have been developed for different classes of queries under this definition.

A natural question to ask is whether offline auditors could directly be used as online auditors as well. The answer to the question, as we shall see, is *no* due to the fact that query denials can leak information. Researchers have proposed the paradigm of *simulatability* to surmount this problem, and developed simulatable auditors for different classes of queries to prevent full disclosure. We will review some of them.

The notion of full disclosure is not entirely satisfactory as a measure of disclosure, so we will next present a recently proposed measure called *partial disclosure* as well as simulatable online auditors that have been proposed for different classes of queries under this definition. We will conclude the chapter with a brief survey of results in another auditing scenario where the information to be protected is an arbitrary view of the database; and finally end with a discussion of the limitations of present day auditing techniques.

1. Auditing Aggregate Queries

Most work on aggregate queries has focused on the case of a single numerical private attribute that is either real valued (from a bounded or unbounded range) or boolean. Additionally, most auditing algorithms developed are for queries of only one kind, with hardness results for auditing combinations of queries. Before proceeding further, we will formalize some of the terminology used in the remainder of this section.

Let $X = \{x_1, \ldots, x_n\}$ be the set of private attribute values of n individuals in a database. An aggregate query $q = (Q, f)$ specifies a subset of the records $Q \subseteq \{1, \ldots, n\}$ and a function f such as `sum`, `max`, `min` 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.

1.1 Offline Auditing

We now survey some of the results in the offline auditing literature.

1.1.1 Full Disclosure.

Given the set of private values X and a set of aggregate queries $\mathcal{Q} = \{q_1, \ldots, q_t\}$ posed over this data set that were correspondingly answered $\{a_1, \ldots, a_t\}$, the goal of an offline auditor is to determine if an individual's private value can be deduced. Traditionally, the definition of disclosure that has been used is the notion of full disclosure defined below.

Definition 1.1 (Full Disclosure) *An element $x_i \in X$ is fully disclosed by a query set $\mathcal{Q}$ if it can be uniquely determined, i.e., in all possible data sets X consistent with the answers $a_1, \ldots, a_t$, to queries $q_1, \ldots, q_t$, x_i is the same.*

As a simple example, if the query set consisted of a single query asking for the sum of the salaries of all the female employees in the company, and Alice was the only female employee in the company, then the answer to this query uniquely determines Alice's salary.

In general the answers to many different queries can be stitched together by a user to uniquely determine an individual's private value. The goal of the auditor then is to prevent such a full disclosure.

1.1.2 Examples of Offline Auditors.

As one example of such an auditor, consider a set of `sum` queries posed over X, the elements of which are real-valued from an unbounded range. To determine if the answers to these queries can be used to uniquely deduce some private value, the auditor essentially needs to solve a system of linear equations. It maintains a matrix where the rows correspond to queries

and the columns to private values. Each query is represented by a vector of 1s and 0s, indexing the private elements that were in the `sum` query. The matrix of query vectors is diagonalized via a series of elementary row operations and column interchanges. If the resulting matrix has a row with only one 1 and $n-1$ 0s, then some element is uniquely determined. Since only a linearly independent set of query vectors need to be examined, the matrix is of size at most $n \times n$, and the diagonalization can be carried out in time $O(n^3)$. Since finding a maximal set of linearly independent query vectors requires $O(n^2|\mathcal{Q}|)$ time, sum queries can be audited in polynomial time.

THEOREM 1.2 *Let $X \in \mathbb{R}^n$ be a data set of private values. There is an algorithm to determine if an $x_i \in X$ is fully disclosed by a set of* `sum` *queries $\mathcal{Q}$ and corresponding answers $\mathcal{A}$ that runs in time $O(n^3+n^2|\mathcal{Q}|)$.*

Besides `sum` queries, offline auditors for exact determination of full disclosure also exist for combinations of `max` and `min` queries, `median` queries and `average` queries over real-valued data. Unfortunately, no significant progress has been made in auditing arbitrary combinations of aggregate queries. For example, the following hardness result has been proved via a reduction from set partition.

THEOREM 1.3 *There is no polynomial time full-disclosure auditing algorithm for* `sum` *and* `max` *queries unless P=NP.*

The auditing problem has also been examined when the private attribute is boolean. Surprisingly, full-disclosure auditing of `sum` queries over boolean data is coNP-hard. There exists an efficient polynomial time algorithm, however, in the special case where the queries are 1-dimensional, i.e., for some ordering of the elements in X, the query set for each query involves a consecutive sequence of x_i's. Considering such restrictions of the general auditing problem is useful in practice, since in reality, users would rarely be able to pose queries over arbitrary subsets of the data. Rather, they would use conditions over some attribute or combinations of attributes to select specific records in the data set to aggregate. For example, a realistic query would ask for the total number of HIV-positive people in a particular age group. The set of queries asking for the total number of HIV-positive people in various age groups would form a set of 1-dimensional `sum` queries over a boolean private attribute. Such assumptions about the structure of queries can yield even more efficient auditors. For example, the `sum` auditor over real-valued data can be made to run in linear time over 1-dimensional `sum` queries.

1.2 Online Auditing

In recent years, researchers have also become interested in the online auditing problem as a means of preventing data disclosure. Given a sequence of queries, $q_1, \ldots, q_{t-1}$ that have already been posed, corresponding answers $a_1, \ldots, a_{t-1}$ that have already been supplied, and a new query q_t, the task of an online auditor is to determine if the new query should be answered as such, or denied in order to prevent a privacy breach. Here each of the previous answers a_i, is itself either the true answer $f_i(Q_i)$ to query q_i, or a "denial".

The earliest online auditors prevented disclosures by restricting the size and overlap of queries that could be answered. For the case of `sum` queries, for instance, it was shown that for queries with query sets of exactly k elements, each pair of query sets overlapping in at most r elements, any data set can be compromised in $(2k - (l + 1))/r$ queries by an attacker who knows l values a priori. For fixed k, r and l, if the auditor denies answers to query $(2k - (l + 1))/r$ and on, then the data set is definitely not compromised, i.e., no private value can be uniquely determined. Such an auditing scheme is rather limited: 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 motivated a search for auditors that could provide greater utility.

The next natural question is whether offline auditors can directly solve the online auditing problem. Whenever a new query is posed, the online auditor checks to see if the answer to this query in combination with all previous query responses can be used to uniquely determine a private value. If so, the query is denied, else it is answered exactly. While it would seem that such an approach should work, in actuality it does not as we demonstrate next.

Example where Denials Leak: Suppose that the underlying data set is real-valued and that a query is denied only if some value is fully disclosed. Suppose that the attacker poses the first query $\texttt{sum}(x_1, x_2, x_3)$ and the auditor answers 15. Suppose also that the attacker then poses a second query $\texttt{max}(x_1, x_2, x_3)$ 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 $\texttt{max}(x_1, x_2, x_3) \not< 5$ since then the sum could not be 15. Further, if $\texttt{max}(x_1, x_2, x_3) > 5$ then the query would not have been denied since no value could be uniquely determined. Consequently, $\texttt{max}(x_1, x_2, x_3) = 5$ and the attacker learns that $x_1 = x_2 = x_3 = 5$ — a privacy breach of all

three entries. The issue here is that denials reduce the space of possible consistent solutions, and we have not explicitly accounted for this.

In this example only a few values were compromised. However, it is possible to construct examples where a large fraction of private values can be uniquely determined. Intuitively, denials that depend on the answer to the current query leak information because users can ask why a query was denied, and the reason is in the data. If the decision to answer or deny a query depends on the actual data, it reduces the set of possible consistent solutions for the underlying data.

Another naive solution to the leakage problem is to deny whenever the offline algorithm does, 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. As a work around to this problem, the *simulation* paradigm (used vastly in cryptography) was proposed and is described next.

1.2.1 Simulatable Auditing. The idea for simulatable auditing came from the following observation: Query denials have the potential to leak information if in choosing to deny, the auditor uses information that is unavailable to the attacker (the answer to the newly posed query). A successful attacker capitalizes on this leakage to infer private values. The requirement of a simulatable auditor then, is that the attacker should be able to simulate or mimic the auditors decisions to answer or deny a query. In such a scenario, because the attacker can equivalently determine for himself when his queries will be denied, denials provably do not leak information. More formally, let $\mathcal{Q} = \{q_1, \ldots, q_t\}$ be any sequence of queries and $\mathcal{A} = \{a_1, \ldots, a_t\}$ be their corresponding answers. Here each a_i is either the exact answer $f_i(Q_i)$ to query q_i on the data set X, or a denial.

Definition 1.4 (Online Auditor) *An online auditor B is a function of $\mathcal{Q}, \mathcal{A}$ and X that returns as output either an exact answer to q_t or a denial.*

Definition 1.5 (Simulatable Auditor) *An online auditor B is simulatable, if there exists another auditor B' that is a function of only $\mathcal{Q}$ and $\mathcal{A} \setminus a_t$ and whose output on q_t is always equal to that of B.*

An attractive property of simulatable auditors 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 future decisions that the auditor makes.

Note that the auditor that restricted the size and overlap of queries was simulatable since it never actually looked at the answers to queries in choosing to deny. As another example of a simulatable auditor, the `sum` auditor over real-valued data from Section 1.1.2 is also simulatable since all that is examined in making the decision to deny or answer is the matrix of query vectors and never the actual answers to any of the queries, let alone the answer to the current query. In contrast to the query-size-and-overlap-restricting auditor, this auditor has also been shown to provide fairly high utility for large data sets — in a sequence of random sum queries over a data set, the first denial can be expected to occur only after a linear number of queries.

A more general sufficient condition for ensuring simulatability is that in making its decision, with each new query, the auditor should determine if there is any possible data set, consistent with all past responses, in which the answer to the current query would cause some element to be fully disclosed. If so, the query should be denied, else it can be answered. Since this is a condition that an attacker could check for himself and predict denials, denials leak no information. Using this idea, simulatable online auditors have been constructed for `max` and `min` queries.

In the example from the previous section, the query $q_1 = \texttt{sum}\{x_1, x_2, x_3\}$ would be answered, since no matter the answer, no element from the data set could be uniquely pinned down. The second query $q_2 = \texttt{max}\{x_1, x_2, x_3\}$ would always be denied, since there is a possible answer to this query, consistent with the answer to q_1 that would cause a private value to be uniquely determined. Note that if the actual answer to q_2 had been greater than $\frac{1}{3}f_1(Q_1)$, q_2 would in reality have been safe to answer, and thus we lose some utility due to the requirement of simulatability.

1.2.2 Partial Disclosure. The notion of full disclosure as a measure of privacy breach has certain shortcomings. Even if a private value cannot be uniquely determined, it might still be determined to lie in a tiny interval, or even in a large interval with a heavily skewed distribution — and some might consider this to be sufficient disclosure. Researchers proposed a new definition of privacy to mitigate this issue by modeling the change in an attacker's confidence about the values of private data points. In this definition, it is assumed that the data is drawn from some distribution $\mathcal{D}$ on $(-\infty, \infty)^n$ that is known to both the

attacker and the auditor. See Section 1.2.3 for some discussion about this assumption.

Let $\mathcal{Q} = \{q_1, \ldots, q_t\}$ be a sequence of queries on the data set X and let $\mathcal{A} = \{a_1, \ldots, a_t\}$ be the corresponding answers. Here each a_i is either the true answer to query q_i on X or a denial. We allow the auditor to be randomized, i.e., it's decision to answer or deny a query need not be deterministic.

DEFINITION 1.6 (RANDOMIZED AUDITOR) *A randomized auditor is a randomized function of $\mathcal{Q}$, $\mathcal{A}$, X and $\mathcal{D}$ that returns as output either an exact answer to q_t on X or a denial.*

We say that the sequence of queries and corresponding answers is λ-safe for an element x_i and an interval $I \subseteq (-\infty, \infty)$ if the attacker's confidence that $x_i \in I$ does not change significantly upon seeing the queries and answers. Consider for example a private value such as salary: if a sequence of queries and answers does not change an attacker's confidence about a private individual's salary, then the sequence is safe.

DEFINITION 1.7 (λ-SAFE) *The sequence of queries and answers, $q_1, \ldots, q_t$, $a_1, \ldots, a_t$ is said to be λ-safe with respect to a data element x_i and an interval $I \subseteq (-\infty, \infty)$ if the following Boolean predicate evaluates to 1:*

$$\text{Safe}_{\lambda,i,I}(q_1, \ldots, q_t, a_1, \ldots, a_t) = \begin{cases} 1 & \text{if } 1/(1+\lambda) \leq \frac{\Pr_{\mathcal{D}}(x_i \in I | q_1, \ldots, q_t, a_1, \ldots, a_t)}{\Pr_{\mathcal{D}}(x_i \in I)} \leq (1+\lambda) \\ 0 & \text{otherwise} \end{cases}$$

Partial disclosure is defined in terms of the following predicate that evaluates to 1 if and only if $q_1, \ldots, q_t$, $a_1, \ldots, a_t$ is λ-safe for all entries and all intervals[1]:

$$\text{AllSafe}_{\lambda}(q_1, \ldots, q_t, a_1, \ldots, a_t) = \begin{cases} 1 & \text{if } \text{Safe}_{\lambda,i,I}(q_1, \ldots, q_t, a_1, \ldots, a_t) = 1, \text{ for every } i \in [n] \text{ and} \\ & \text{every interval } I \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

We now turn to the privacy definition. Consider the following (λ, 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 $q_t = (Q_t, f_t)$.

[1]In reality, the privacy definition only considers all intervals that have a significant prior probability mass.

2 The auditor determines whether q_t should be answered. The auditor responds with $a_t = f_t(Q_t)$ if q_t is allowed and with $a_t =$ "denied" otherwise.

3 The attacker wins if $\text{AllSafe}_\lambda(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$.[2]

DEFINITION 1.8 (PRIVATE RANDOMIZED AUDITOR) *An auditor is* (λ, δ, T)-private *if for any attacker* A

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

Here the probability is taken over the distribution $\mathcal{D}$ *that the data comes from and the coin tosses of the auditor and the attacker.*

Since here too, one would like to ensure that denials leak no information, the condition of simulatability is imposed on auditors that are designed. Consider $\mathcal{Q}$, $\mathcal{A}$ and X as before. Then,

DEFINITION 1.9 (SIMULATABLE RANDOMIZED AUDITOR) *A randomized auditor* B *is simulatable, if there exists another auditor* B' *that is a randomized function of only* $\mathcal{Q}$, $\mathcal{A} \setminus a_t$ *and* $\mathcal{D}$ *such that the output of* B' *on* q_t *is computationally indistinguishable from that of* B.

1.2.3 Discussion on Privacy Definition. Note that the above definition of privacy makes the assumption that the distribution from which the data is drawn is known to the attacker. In reality it need not be. In this scenario, the predicate AllSafe needs to be evaluated with respect to the attacker's prior distribution, since compromise occurs only if there is a substantial change in his beliefs. However, if the attacker's distribution can be arbitrarily far from the true data distribution, there is not much that the auditor can release without causing partial disclosure of some private value, since it is required to release exact answers if at all. For example, consider a database that contains *height* as a private attribute, and consider an attacker whose prior belief is that all men are less than a foot tall. If by querying the data, the attacker suddenly learns that this is not true and there is substantial change in his posterior distribution, the privacy breach would be massive. In reality, his prior beliefs are so far off the mark, that there is no aggregate query about the heights that the auditor can truthfully answer without compromising privacy, not even the average height of all people in the database.

Instead the data distribution that we assume the auditor and the attacker share is supposed to represent such common sense facts and it

[2]Hereafter, we will refer to the predicates without mentioning the queries and answers for the sake of clarity.

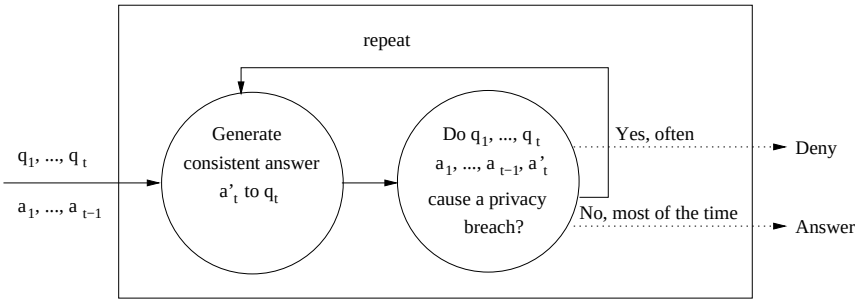

Figure 1.1. Skeleton of a simulatable private randomized auditor

allows for more useful information to be released. There are many circumstances where such an assumption is realistic. For example, distributions of attributes such as *age* or *salary* may be known from previous data releases or even published by the auditor itself.

1.2.4 A General Approach for Constructing Private Randomized Auditors. A query is thus 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 any interval. Also, the decision to deny must be simulatable. We now describe a general approach that could be used to construct such simulatable randomized auditors. Figure 1.1 gives a high level picture.

The basic idea is to have the auditor generate random data sets (of n private values) consistent with answers to past queries. The data sets are generated according to the distribution $\mathcal{D}$ conditioned on the past answers. The auditor then checks to see if answering the new query on these random data sets 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 data sets, the query is safe to answer. Since the true answer to the query is never looked at in this process, the auditors are simulatable and denials provably do not leak information.

The left circle in Figure 1.1 thus represents the process of generating a possible answer a'_t to the new query according to $\mathcal{D}$ conditioned on past answers, and the right circle represents the evaluation of the predicate AllSafe (Equation 1.1) that checks to see whether privacy is violated for any x_i and any interval I if a'_t were revealed in conjunction with all previous answers. For each new query this procedure is repeated many times, and the decision to deny is based on the fraction of sampled consistent answers that cause a privacy breach. By repeating often enough and choosing an appropriate cut-off for denials, it can be shown using

Chernoff bounds, that the above procedure gives us a (λ, δ, T)-private auditor.

One technicality arises from the fact that the AllSafe predicate needs to be evaluated with respect to an infinite number of intervals. It can be shown that requiring the a-priori and posteriori probabilities of a private value to be close on arbitrarily small intervals would cause no queries to be answered at all, and therefore existing literature focuses on protecting the privacy of only intervals that have a significant a-priori probability mass. While there may also be an infinite number of such intervals, it can be shown that if each x_i is drawn independently according to some distribution $\mathcal{H}$ on $(-\infty, \infty)$, then we only need to check for privacy with respect to a finite number of non-overlapping intervals $\mathcal{I}$. Thus far randomized auditors have only been designed for data sets where the private values are drawn iid from such an underlying distribution.

1.2.5 Randomized Auditor for Sum Queries. We will now briefly describe how the above generic approach can be tuned to obtain a private randomized auditor for `sum` queries (where each query is of the form $\texttt{sum}(Q_j)$ for some query set, Q_j).

Prior to describing the solution, we give some intuition. Assume for simplicity that each private value is drawn uniformly at random from the range $[0, 1]$. Then the data set $X = \{x_1, \ldots, x_n\}$ can be any point in the unit cube $[0, 1]^n$ with equal probability. A sum query and its corresponding answer induce a hyperplane. The data sets consistent with one sum query and its answer are then those points in $[0, 1]^n$ that fall on this hyperplane. Each successive query and answer reduces the space of possible consistent data sets to those points in $[0, 1]^n$ that fall in the intersection of the induced hyperplanes, i.e., the consistent data sets lie in a convex polytope. Because the prior distribution is uniform, the posterior distribution (given the queries and answers) inside the convex polytope is also uniform. Thus it would suffice to sample data sets uniformly at random from this convex polytope to generate the consistent answers required in the left circle of Figure 1.1. Further we can determine if the answer to the query in a sampled data set would cause a privacy breach (in the right circle of Figure 1.1): Suppose that $\mathcal{P}$ is the current convex polytope. To determine if a partial disclosure has occurred for a particular individual x_i and a particular interval $I \in \mathcal{I}$, consider the definition of privacy breach:

$$\frac{Pr_{\mathcal{D}}\{x_i \in I | q_1, \ldots q_t, a_1, \ldots a_t\}}{Pr_{\mathcal{D}}\{x_i \in I\}} = \frac{\Pr_{\mathcal{D}}\{x_i \in I | \vec{x} \in \mathcal{P}\}}{|I|}$$

The probability in the numerator can be estimated by sampling from the convex polytope $\mathcal{P}$ and counting the fraction of the sampled points for which x_i lies inside I. If the fraction above is greater than $(1+\lambda)$ or less than $\frac{1}{1+\lambda}$ then the query is unsafe for this sampled data set.

Rather than a uniform prior distribution, we can assume an even more general log-concave distribution, since algorithms exist for sampling from it. The class of log-concave distributions forms a common generalization of uniform distributions on convex sets and Gaussian distributions. A distribution over a domain T is said to be log-concave if it has a density function g such that the logarithm of g is concave on its support. That is, the density function $g : T \to \mathbb{R}_+$ is *log-concave* if it satisfies $g(\alpha x + (1-\alpha)y) \geq g(x)^{\alpha} g(y)^{1-\alpha}$ for every $x, y \in T$ and $0 \leq \alpha \leq 1$. These distributions constitute a broad class and play an important role in stochastic optimization.

Assume that each element x_i is independently drawn according to the same log-concave distribution H over $\mathbb{R}$. Let $\mathcal{D} = H^n$ denote the joint distribution. Using the properties of log-concave functions, it can be shown that the joint distribution $\mathcal{D}$ is also log-concave and further, the posterior distribution, $\mathcal{D}$ conditioned on $\wedge_{j=1}^{t}(\texttt{sum}(Q_j) = a_j)$ is also log-concave. In addition, there exist randomized, polynomial-time algorithms for sampling (with a small error) from a log-concave distribution.

Without going into the technical details, we will sketch how one can adapt the generic randomized auditor from Section 1.2.4 for the problem of auditing `sum` queries. An algorithm for sampling from a log-concave distribution can be used to estimate the posterior probabilities required for evaluating the AllSafe predicate in the right circle of Figure 1.1. This algorithm can also be used in the left circle of Figure 1.1 for sampling data sets and hence consistent answers from the posterior distribution $\mathcal{D}$ conditioned on previous answers. The AllSafe predicate is evaluated for a λ' smaller than λ to accommodate the sampling algorithm's inability to sample exactly from the underlying log-concave distribution.

Besides `sum` queries, randomized auditors have also been developed for `max` queries where the sampling procedure for uniform priors is much more efficient, and for combinations of `max` and `min` queries. We do not go in to the details in this chapter, instead we next very briefly discuss auditing in another scenario when the queries are not aggregate queries.

2. Auditing Select-Project-Join Queries

Other than aggregate queries, auditing has also been studied in the context of select-project-join queries when the information to be kept confidential is a forbidden view of the database. The secret view itself is

also specified via a select-project-join query. For example the database may consist of a single relation, Employee(*name, department, phone*), and the forbidden view may be of the form $\pi_{name,phone}$(Employee). Here π represents the projection of the table on to the *name* and *phone* attributes. The forbidden view thus represents that the *name* and *phone* attributes of the Employee relation, or perhaps some combination of them, are sensitive and should not be revealed. The task of an offline auditor then is to determine whether a set of select-project-join queries answered in the past disclosed any information about the forbidden view, and the task of an online auditor is to deny queries when their answers could disclose information about the forbidden view.

The precise semantics of what the forbidden view represents in terms of what should be kept private could vary from system to system. For example, the above forbidden view could represent the requirement that not a single phone number or name in the database should be disclosed. Alternatively, it could represent the requirement that it is only the association between the name and phone number of any individual in the database that should be kept private and so on. The first ever formal notion of forbidden view privacy suggested in the literature was the notion of perfect privacy defined below. It assumes an underlying distribution $\mathcal{D}$ that the tuples of the database are drawn from.

Definition 1.10 (Perfect Privacy) *Let $\mathcal{D}$ be the underlying distribution according to which tuples of the database are drawn. A set of queries, $\mathcal{Q}$, are said to respect* perfect privacy *of a forbidden view $\mathcal{V}$ if for any set of answers to the queries, $\vec{a}$, and any instantiation of the forbidden view, v,*

$$Pr_{\mathcal{D}}\{\mathcal{V} = v | \mathcal{Q} = \vec{a}\} = Pr_{\mathcal{D}}\{\mathcal{V} = v\}$$

If the distribution $\mathcal{D}$ is such that each tuple t_i from the (finite) domain of possible tuples is included in the database with some probability p_i, independently of other tuples, the condition of checking for perfect privacy of a set of queries reduces to a purely logical statement. We will introduce some definitions before stating the result.

Definition 1.11 (Critical Tuple) *A tuple t from the finite domain of possible tuples is critical for a query Q, if there exists a possible instance of the database, I, where the presence or absence of t makes a difference to the result of Q, i.e., $Q(I - \{t\}) \neq Q(I)$.*

We then get the following characterization of query-view privacy which applies for queries that follow a set semantics.

THEOREM 1.12 *A set of queries, $\mathcal{Q}$, violates perfect privacy of a forbidden view, $\mathcal{V}$, if and only if there exists a tuple in the domain of possible tuples that is critical to both $\mathcal{V}$ and some query in $\mathcal{Q}$.*

This useful result implies that for a set of queries to violate perfect privacy of the forbidden view, some query in the set must violate it. Thus an offline auditor auditing a set of queries to check for violations of perfect privacy needs to audit each query in turn, and an online auditor interested in maintaining perfect privacy of the forbidden view can make its decisions to answer or deny each new query independently of past queries. Collusion between users is not a problem. In addition, since tuple criticality and therefore query denials are independent of the actual database instance, such an online auditor is simulatable and denials do not leak information.

Unfortunately, checking the condition in Theorem 1.12 is Π_2^P-complete, even when the forbidden view and the queries are conjunctive. Auditors have been developed, however, for particular subclasses of conjunctive queries. Even so an online auditor that maintains perfect privacy of its forbidden view would result in a very strict denial policy. For instance, going back to the example of the Employee relation, suppose the forbidden view is π_{phone}(Employee), then even just the query π_{name}(Employee) asking for the names of all employees would be denied, even though it does not access a single phone number. This is because every single tuple in the domain of possible tuples would be critical to both the forbidden view and the query. The idea is that just by revealing information about the size of the relation, the query reveals some information about the forbidden view and should be denied. The notion of perfect privacy of the forbidden view may thus be a little too strong.

Ongoing research aims to relax the notion of privacy of a forbidden view, thereby permitting auditors that would provide more utility to a user. These new notions of privacy also permit more efficient auditors that can run in polynomial time for large classes of queries. See Section 4 for recommended reading on this topic.

3. Challenges in Auditing

We describe challenges and future directions in auditing where further research is warranted.

Privacy Definition: There has been a steady evolution of privacy definitions and notions of compromise over the years starting from full disclosure (Definition 1.1) to more recent notions of partial disclosure (Definition 1.8) and perfect privacy (Definition 1.10). But there is certainly room for further improvement. One assumption made by the more re-

cent definitions is that there is one probability distribution $\mathcal{D}$ from which the data is generated and which is known to both the attacker and the auditor. In reality, there are two other distributions, the attacker's prior and the auditor's prior. While it may be reasonable to assume that these three distributions are close, current definitions and auditors all assume that these three distributions are the same. In the case of aggregate queries, another problem is that current definitions only consider the privacy of a single individual to be important, whereas in reality, it may be important to protect the privacy of groups of individuals such as families. In the case of select-project-join queries, the notion of perfect privacy is far too strong causing many seemingly innocuous queries to be deemed suspicious.

Algorithmic Limitations: Online simulatable algorithms for auditing aggregate queries following the general framework suggested in this chapter have several limitations. They require sampling a data set consistent with a given set of queries and answers. In practice, this procedure may be computationally prohibitive given the massive size of data sets, although such sampling algorithms have been steadily improving over the years. In addition, as already mentioned, it is assumed that both the attacker and auditor know the distribution $\mathcal{D}$ from which the data is generated. Algorithms that could overcome these sampling requirements would make great improvements.

Section 1 largely focused on auditing one kind of query: `sum`. In reality, a large variety of queries are posed to data sets. While there has been some investigation into auditing `max`, `min`, `median` queries, intermingling these queries has proven to be a greater challenge. For example, under full disclosure, it is NP-hard to audit intermingled `sum` and `max` queries, while polynomial time algorithms are known for auditing exclusively `sum` queries and exclusively `max` queries. While there are situations in which only one kind of query need be considered (e.g. when releasing contingency tables `sum` queries are the only kind of queries that are answered), ultimately, in order for auditing to be truly useful, we will need to allow richer queries of varied types, such as those posed in data mining applications such as clustering or decision tree classification.

As mentioned in Section 2 checking for perfect privacy violations of the forbidden view for a very simple kind of probability distribution is Π_2^P-complete even just for conjunctive queries and views. While, auditors have been developed for various subclasses of conjunctive queries, weakening the requirement of "perfect privacy" may go a long way in enabling the design of efficient auditors for larger classes of queries. There has already been some effort in this direction, where assumptions are imposed on the distribution from which the data is drawn.

Collusion: Collusion is a largely unaddressed issue in most interactive data sharing mechanisms today. In the absence of any obstacles to collusion, the online auditors from Section 1 would need to pool together aggregate queries posed by all users in the past in order to determine potential privacy breaches. This could result in a user receiving more than his fair share of denials. On a related note, online auditors might need to maintain a large audit trail of queries posed in the past. While the auditors we saw in this chapter were able to maintain a query history of bounded size, or even no query history at all, this need not be true in general, and with the possibility of collusion, larger query histories may need to be stored for longer periods of time. The notion of perfect privacy (1.10) is so strong that past queries need not even be considered in determining privacy breaches — a set of queries leak information about a forbidden view only if some query in the set leaks some (potentially negligible) amount of information about the view. However, strengthening the privacy definition in this way, results in only more denials, and is not a satisfactory solution to the collusion problem.

Utility: While there have been some initial analyses on the utility of online auditors, utility is a dimension that is not well understood. How should we even define utility? One line of work attempts to study the expected number of denials in a random sequence of aggregate queries. However, it is unlikely that users would be able to pose aggregate queries over arbitrary subsets of the data and queries are likely to come from a non-uniform distribution. Furthermore, there might be some important, fairly generic queries, that should always be answered, such as the total number of HIV-positive people in the country. An auditor that would deny such a query could be construed as providing weak utility.

In general, we would like to ensure that a database will not be rendered useless with too many denials. To this end, it might well be worthwhile to sacrifice some privacy for greater utility.

4. Reading

A general, though somewhat dated, overview of disclosure control methods for statistical databases can be found in [1]. Some of the representative work in offline auditors for aggregate queries and full disclosure can be found in [4, 8, 19, 12]. [4] describes offline auditors for `sum` and `max` queries over real-valued data. [8] considers auditing subcube queries, [19] considers the case of auditing `average` and `median` queries, while [12] considers the case when the private attribute is boolean. Most of the above work treats online and offline auditing interchangeably — the difference is not made explicit — and in [11] the issue of denials leak-

ing information is uncovered. [11] proposes the simulatable auditing paradigm as a solution, and [10] and [18] construct online simulatable auditors for different kinds of queries and different kinds of data distributions. Chapter 2 in [10] is an extended and refined version of [11]. Algorithms for uniform sampling from convex polytopes and from log-concave distributions can be found in [13, 3, 7, 9, 14]. [18] also contains an initial analysis of the utility of online auditors. The earliest examples of online auditors that restrict the size and overlap of queries can be found at [6].

The work on auditing select-project-join queries presented in this chapter can be found in [16]. [15] contains algorithms for auditing specific classes of conjunctive queries to check for perfect privacy violations of the forbidden view. [5] considers a data distribution that is a variant of that considered in [16] where tuples are drawn independently of one another, but the expected size of the database is a constant. The authors show that privacy violations for conjunctive queries and views can be determined algorithmically in this situation as the size of the domain of the tuples grows to infinity. [2] builds a practical system for detecting "suspicious" select-project-join queries, however the privacy guarantees of their definition of suspiciousness are not made explicit. [17] suggests other notions of suspiciousness that lie in between those of [16] and [2] both in terms of their disclosure detection guarantees and the ease of auditing under them.

References

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

[2] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kieman, R. Rantzau, and R. Srikant. Auditing Compliance with a Hippocratic Database. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2004.

[3] D. Applegate and R. Kannan. Sampling and integration of near log-concave functions. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 156–163, 1991.

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

[5] N. Dalvi, G. Miklau, and D. Suciu. Asymptotic Conditional Probabilities for Conjunctive Queries. In *Proceedings of the International Conference on Database Theory (ICDT)*, 2007.

[6] D. Dobkin, A. Jones, and R. Lipton. Secure Databases: Protection against User Influence. *ACM Transactions on Database Systems (TODS)*, 4(1):97–106, 1979.

[7] A. Frieze and R. Kannan. Log-sobolev inequalities and sampling from log-concave distributions. *Annals of Applied Probability*, 9(1):14–26, February 1999.

[8] J. Kam and J. Ullman. A model of statistical databases and their security. *ACM Transactions on Database Systems (TODS)*, 2(1):1–10, 1977.

[9] 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.

[10] K. Kenthapadi. Models and Algorithms for Data Privacy. *Ph.D. Thesis, Computer Science Department, Stanford University*, 2006.

[11] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable Auditing. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 118–127, 2005.

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

[13] L. Lovasz and S. Vempala. Logconcave functions: Geometry and efficient sampling algorithms. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

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

[15] A. Machanavajjhala and J. Gehrke. On the Efficiency of Checking Perfect Privacy. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2006.

[16] G. Miklau and D. Suciu. A Formal Analysis of Information Disclosure in Data Exchange. *Journal of Computer and System Sciences*, 2006.

[17] R. Motwani, S. U. Nabar, and D. Thomas. Auditing SQL Queries. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2008.

[18] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards Robustness in Query Auditing. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2006.

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