Journal of Privacy and Confidentiality (2010) **2**, Number 1, pp. 109–140

Releasing Private Contingency Tables

Shubha U. Nabar[*] and Nina Mishra[†]

1 Introduction

Statistical agencies such as the US Census Bureau routinely release aggregate statistics about the general population. These statistics are often reported in the form of contingency tables. A 2-dimensional contingency table is an $(m+1) \times (n+1)$ matrix over two attributes that are binned into m rows and n columns. For instance, the attributes could be Age binned into buckets of length 10 and Height binned into buckets of length 5. For each cell (i, j) in the matrix, the table reports an aggregate value called the *cell value*. This is often an aggregate of some private attribute of the population. For example, the table may report the total number of individuals with diabetes in each cell of the table. In this case, the private attribute being aggregated is Boolean, either 1 if the individual has diabetes or 0 if he does not. The final row $i = m + 1$ (resp., column $j = n + 1$) contains the total *row* (resp., *column*) *sums*, e.g., the total number of individuals in each row who have diabetes. Besides the cell values, the total number of individuals that fall in each cell is also publicly known, e.g., the total number of individuals in the age range 10-20 years and height range 165-170 cm, regardless of whether or not they have diabetes.

To preserve the privacy of individuals, statistical agencies suppress the values of so-called "sensitive" cells. These suppressions are called *primary suppressions*. Primary suppressions do not necessarily protect the values of the sensitive cells since the suppressed values may still be deduced via the row and column sums. Consequently, additional cells are suppressed, also known as *complementary suppressions*, in order to protect the primary suppressions. It is important to strike a balance between privacy and utility since the more cells that are suppressed, the less useful the table. Various criteria are employed to measure utility, including the total number of suppressed cells, the total value of the suppressed cells, or some combination of both.

Cell suppression is widely used in practice. It is, for example, the most common technique for protecting the confidentiality of personal data in tabulations of economic censuses and surveys [57]. It has also been the subject of a large literature in the statistical database community. Our work uncovers fundamental problems with the way cell suppression has been approached for the last thirty years.

First of all, the rules for choosing sensitive cells for the primary suppressions have heretofore been somewhat arbitrary, with no concrete privacy guarantees for individuals [79]. For example, cells with values less than some small constant k are often considered sensitive and therefore suppressed. Secondly, we argue that these algorithms heavily rely on security through obscurity — the hope is that by keeping the suppres-

[*]Microsoft Corporation, `mailto:shubhan@microsoft.com`
[†]Search Labs, Microsoft Research, `mailto:ninam@microsoft.com`

sion heuristic a secret, the values of the sensitive cells will be protected. This is often a bad idea (Kerckhoff's principle), and we first show how knowledge of the suppression heuristic can be used to determine the values of the sensitive cells despite the protective presence of secondary suppressions. In this attack, not only are the sensitive cell values revealed, but also actual private values of individuals in these cells can be exactly determined. This highlights another fallacy of existing suppression methodologies: they neglect the fact that cells do not need privacy, people do!

Our first step therefore is to use a rigorous definition to describe what it means for a contingency table to respect the privacy of individuals. By so doing, we no longer rely on an implicit notion of privacy determined by heuristics used for choosing sensitive cells. We then make the observation that the problem of suppressing cells in a table in order to protect individual privacy is equivalent to the problem faced by an online query auditor for aggregate queries over a statistical database. The goal of the auditor in the online auditing problem is to *deny* queries when answers to the queries may cause a privacy breach. The equivalence between suppressions and denials gives us a new set of tools with which to design new cell suppression solutions for releasing private contingency tables.

Using these tools we design a general algorithmic framework for privacy-preserving cell suppression. Applying the algorithm efficiently in a particular situation requires solving an interesting theoretical problem: how do you sample a random data set that is consistent with a partially filled contingency table? We consider the important case where the private attributes are Boolean — a very common scenario for statistical tables. We give two efficient algorithms for this case: one based on dynamic programming and the other on sampling perfect matchings. We thus build a complete cell suppression solution for the case of Boolean private attributes.

Finally we undertake a theoretical and experimental study of the utility of our suppression algorithm: will the algorithm suppress excessively, or can utility yet be derived from released contingency tables? Our results here are preliminary, but give indications as to how attributes should be binned to minimize suppressions.

In this paper, not only do we harness the tools developed in the query auditing literature, but we also make foundational contributions to the query auditing problem itself: (1) We show impossibility results to demonstrate that privacy definitions used in other areas of privacy-preserving data mining can never be satisfied by query auditing techniques without a complete loss of utility. (2) For the first time, we analyze and justify an assumption made by existing query auditing solutions on the prior knowledge of attackers. To summarize, the contributions of this paper are:

- In Section 3, we demonstrate how existing cell suppression methodologies provide inadequate privacy to individuals in a contingency table.
- In Section 4.1, we provide a rigorous definition of a private contingency table, where privacy is defined with respect to individuals in the table.
- In Section 4.2, we make explicit the connection between query auditing and cell

suppression and in Section 4.3, we analyze and justify unexamined assumptions previously made in the query auditing framework.

- In Section 5.1, we describe a general algorithmic framework for privacy-preserving cell suppression.
- In Section 5.2, we study the special case of Boolean private attributes. This results in a new theoretical problem of nonuniform sampling of contingency tables, for which we give two algorithms. While our algorithms are for the special case of Boolean private attributes, the framework from Section 5.1 applies more generally to other kinds of attributes as well.
- In Section 6, we consider the utility of our approach and analyze both theoretically and experimentally the kinds of tables that would be published with very few suppressions.

2 Related Work

Existing cell suppression methodologies are described in [47, 13, 14, 54, 44, 45]. We discuss this work in more detail in the next section. The literature on query auditing [56, 74] is also relevant to our problem as we will show in Section 4.2. A survey of this literature can be found in [73].

With regards to rigorous privacy definitions, there is a large body of work that defines privacy via *differential privacy* [30, 7, 8, 6]: A hypothetical attacker who knows everyone's private value but k's, gains little knowledge about k from the released information. We do not use this privacy definition in our work because our goal is to give exact cell values, if at all — thus an attacker who knows everyone's data except k's will automatically learn k's private value with even just the total table sum.

While our work focuses on cell-suppression in two-dimensional tables, the literature on privately releasing contingency tables is much broader and includes methodologies other than cell suppression for multidimensional tables as well. A survey of these methodologies used in the statistical community can be found in [43]. The approaches followed here include releasing data for only a sample of the population, e.g., [38], releasing only a subset of lower dimensional marginal and conditional tables, e.g., [26], and applying stochastic perturbations to individual records with the constraint that the transformed data is consistent with the released marginals, e.g., [40, 27]. Once again the emphasis is on protecting cells with small counts and the risk of disclosure is measured by computing upper and lower bounds for cell counts, the number of tables satisfying the released marginal and conditional constraints, and the distribution over these tables, with the goal of ensuring that probabilities do not concentrate on a few values between the bounds, e.g., [24, 25, 26]. In [83, 84] and [26] the authors build a general Bayesian framework for balancing the trade-off between the risk of disclosure versus the utility of the released data. This framework, however, does not take into account information that may be gleaned from the marginals that are *not* released. As we show in this paper, such negative decisions can themselves reveal information and therefore need to be made carefully. [41] and [42] demonstrate that the methods for selecting an appropriate subset of marginals also relate to the association rule hiding problem studied in the data mining community. The authors describe how to determine which association rules to hide in order to preserve privacy while permitting statistical inference.

A recent alternative solution to releasing exact cell values was given in [6] where it is shown that differential privacy can be guaranteed by perturbing cell values. Since the perturbed data may not be consistent (e.g., cell values could be negative), further steps are taken to make it consistent while satisfying the privacy requirements. Lower bounds on the minimum amount of perturbation needed to achieve differential privacy are presented in [53]. We do not consider perturbation approaches in this paper. Our focus is on releasing exact cell values, which is valuable when the consumer of the data requires exact numbers to make informed decisions, e.g., medical data. We therefore design a systematic framework to suppress cells while providing strong privacy guarantees.

Suppressions are also considered in the literature on k-anonymity [82, 70, 3, 75, 59], ℓ-diversity [67] and t-closeness [60]. However, in this line of work, it is the public attributes of individuals that are suppressed so that every individual "hides in a crowd", whereas we are interested in suppressing aggregates of private values. Besides this superficial difference, the privacy definitions used in these papers are syntactic in contrast to the semantic privacy definition that we use. Both sets of techniques can be viewed as creating equivalence classes. However, whereas algorithms for k-anonymity heavily use the underlying data to determine the equivalence classes, we take great care to create the classes in a manner that is oblivious to the underlying data since using this data can potentially leak information.

3 Suppressions Leak Information

In this section we make several crucial observations about existing cell suppression methodologies: (1) These methodologies can potentially leak information — in particular, algorithms and heuristics for primary and secondary suppressions should be publicly known and this information can be used by an attacker to breach the privacy of individuals. (2) Releasing row and column sums could at times be detrimental to preserving privacy. (3) Cells do not need privacy, people do. (4) Distributional knowledge about the data should be used in determining which cells to suppress.

In order to demonstrate these issues, we describe privacy definitions currently advocated in the literature. Gusfield [47] uses *exact disclosure* to define compromise. Under this definition, privacy is breached if a sensitive cell's value can be uniquely determined from the table. Gusfield gives polynomial-time algorithms for finding the minimum number of cells to suppress so as to ensure that no sensitive cell value can be exactly disclosed, under certain restrictions.

Subsequent work defined compromise as *interval/partial disclosure*, where a sensitive cell is protected if it cannot be deduced to lie within a specified width. Cox's seminal work [13, 14] modeled the problem as an integer program that minimized the number of suppressed cells as well as the total value of suppressed cells. Other important integer programming formulations include [54, 44, 45].

Now, consider the maximally suppressed table in Figure 1(a) where each individual contributes a value of 0 or 1 to the cell (i.e., the private attribute value is Boolean). Suppose also that the sensitive cells (primary suppressions) are those with value ≤ 5, a simplification for the sake of an example.

	blonde	brown	black	
blue	*	*	*	110
not blue	*	*	*	105
	55	105	55	215

(a) Maximally Suppressed Table

	blonde	brown	black	
blue	[0,55]	[0,105]	[0,55]	110
not blue	[0,55]	[0,105]	[0,55]	105
	55	105	55	215

(b) Legitimate Ranges

Figure 1: (a) A maximally suppressed table where rows correspond to eye color, columns to hair color and entries to the number of people who satisfy the row and column heading who possess a certain gene. (b) A table with inferred, legitimate ranges.

What can an attacker deduce from this table? Since every cell is suppressed, by definition in previous work, privacy is maximally preserved. Indeed, by [47], no cell value can be uniquely determined. Furthermore, via [45], even the range of legitimate values for each cell, shown in Figure 1(b), is very wide. So it would seem that privacy is preserved in the strongest sense possible.

The Attack: However, what previous work ignores is that the primary suppression algorithm is public, or at least it should be (Kerckhoff's principle). The purpose of secondary suppressions is to prevent an attacker from learning the primary suppressions. We now show how the primary suppressions can be revealed. Recall that in this example, cells with a value ≤ 5 are chosen for the initial set of suppressions. This implies that at least one of the cells has a value between 0 and 5. But actually, if there were only one primary suppression, then the minimum number of required secondary suppressions would be three, i.e., a total of four cells in two columns would be suppressed. Thus, an algorithm that minimizes the number of suppressed cells would not output such a table. Similarly, if there were only two primary suppressions then there would also be at most four suppressions in total. Further, note that there cannot be two primary suppressions in a single column, since then the column sum would be at most 10. Observe that this also implies there cannot be four or more primary suppressions in the table. In addition, there cannot be three suppressions in one row since then the row sum would be at most 15. Consequently, the only possible primary suppressions are cells $\{1, 3, 5\}$ or $\{2, 4, 6\}$, where we have numbered the six cells in row major order.

Using distributional knowledge, it is now possible to further determine which of the two primary suppressions is more likely. Knowing that the row headings describe eye color, the column headings hair color and the cell entries the number of individuals in that cell who possess a certain gene, an attacker can use public knowledge about the gene to attack the table. For instance, if the gene is more likely to be present in blue-eyed, blonde-haired individuals then the table with primary suppressions $\{2, 4, 6\}$ is more likely. Further, if it is known that exactly 100 of the brown-haired individuals were not blue-eyed (recall that the total number of individuals that fall in each cell is publicly known), then it is clear that they must all possess the gene (since otherwise the total column sum for brown-haired individuals would be less than 105) and the privacy of all these individuals has been massively breached!

A natural next question then is what table is safe to release? To answer this question it is important that we first discuss the definition of privacy.

4 Framework

4.1 Privacy Definition

As noted in Section 3, cells do not need privacy, people do. Even if no cell value can be uniquely determined, and even if there is a wide range of acceptable values, privacy can still be breached through common sense distributional knowledge and knowledge of the primary/secondary suppression algorithm. Consequently, a new definition of privacy is needed and we adopt one akin to [35, 56, 72, 74, 71, 65]. Intuitively, an individual's value is private if an attacker gains little knowledge about that person from a released table M. We assume that the data is generated from some underlying distribution D that is known to the attacker. We say that privacy is preserved if for each individual k holding private value X_k and for each value v in the domain of private values, $P_D(X_k = v) \approx P_D(X_k = v|M)$. In other words, an attacker's belief about any individual's private value should be the same after seeing the published table as before. More formally,

Definition 1 (ϵ-Semantic Privacy) *Let D be a distribution describing the attacker's prior knowledge which is also the distribution from which the data is drawn. Let M be a released contingency table. We say that M is* ϵ-semantically-private *if for every individual k, with corresponding private value X_k, and for every v in the domain of private values:*

$$|1 - \frac{P_D(X_k = v|M)}{P_D(X_k = v)}| \leq \epsilon$$

We address the assumption about the attacker's prior knowledge in Section 4.3. Given this privacy definition, we now define the cell suppression problem as follows.

Definition 2 (The Cell Suppression Problem) *Consider a data set of individuals with two public attributes A, B, and a private attribute X, drawn according to a distribution D. Release a 2-dimensional contingency table M over the attributes A and B,* *such that for each cell (i,j) of the table, $M_{ij} \in \{\sum_{k\in(i,j)} X_k, *\}$ and M is ϵ-semantically private. Here $*$ corresponds to a cell that is suppressed.*

4.2 Connection to Query Auditing

As mentioned in the Introduction, the cell suppression problem for contingency tables is related to the online query auditing problem for statistical databases. Given a sequence of queries that were posed in the past with corresponding responses and given a new query, the role of a query auditor is to either answer the new query if a privacy breach is not possible or deny it otherwise. We observe that the cell suppression problem is a special case of the auditing problem where the queries are sum queries with a tabular structure and cell suppressions correspond to query denials. A row and column combination, i.e., a cell, corresponds to a sum query. Answering the sum query corresponds to revealing the cell value, and denying it corresponds to suppressing the cell value.

Previous work on auditing has already noted that denials have the potential to leak information. Our work makes the connection to cell suppressions. As the example in Section 3 demonstrates, the reason is that primary suppression decisions are made with information that is unavailable to an attacker. The suppressions therefore reduce the space of possible consistent underlying data sets and the simplistic suppressors fail to explicitly account for this.

Simulatability: We overcome this problem via *simulatability* as proposed in query auditing solutions [56, 74]. As per this paradigm, a suppressor should never look at the true value of a cell when deciding whether or not to suppress it. In fact the attacker should be able to "simulate" the suppression algorithm and predict on his own when cells will be suppressed. This would ensure that suppressions leak no information at all. The suppression algorithms we seek to develop should thus be simulatable.

A relevant question then is what are sufficient conditions to ensure that a suppression algorithm is simulatable and releases a private contingency table? For our definition of privacy, it suffices that for every cell, the suppressor determines if the value of the cell is *likely* to be such as to cause a significant change in the attacker's confidence about any individual's private value. If so, the cell value should be suppressed, or else it can be revealed. In estimating this likelihood of a cell value being unsafe, the suppressor should never look at the actual cell value, but it may make use of the distribution D from which the data is drawn, since this is information that is available to the attacker as well. We thus look for suppressors that are simulatable and guarantee ϵ-semantic privacy.

4.3 Assumptions on D

Note that our privacy definition makes the strong assumption that the distribution D that generates the data is known to the attacker. This assumption has been made in all previous work on query auditing [56, 74, 71, 65]. In practice the attacker may have a prior distribution D_A that is quite different from the true data distribution D. We will now justify the assumption that it suffices to consider only the attacker with $D_A = D$. Beyond just the immediate application to contingency tables, this result furthers our understanding of existing query auditing solutions as well.

In general the attacker's prior distribution D_A on the data may be arbitrarily far from the true data distribution D. The strongest privacy requirement used in perturbation based approaches to privacy-preserving data mining would be that no matter the attacker's prior, his posterior given the released contingency table should not change by much, i.e., $\forall A, k, v \;\; P_{D_A}(X_k = v|M) \approx P_{D_A}(X_k = v)$

Such a stringent privacy requirement, if used in our scenario, would result in the entire contingency table being suppressed since the suppressor must reveal true cell values, if at all. As a simple example, if the private attribute is Height and an attacker believes that all men are dwarves, then even just the total height of all individuals in a population cannot be released without significantly affecting the attacker's posterior beliefs. But should this change in the attacker's beliefs be counted as a privacy violation? Below we give a more formal example.

Impossibility Result: Consider a data set with N people where each individual k has a private value X_k that is independently 1 with probability $p_D = \frac{1}{2}$ and 0 with probability $\frac{1}{2}$. And assume that $\sum_{k=1}^{N} X_k = N/2$, where N is even — which is exactly according to expectation. Suppose further that the attacker's prior distribution D_A is far from D so that $P_{D_A}(X_k = 1) = 0.01$ and $P_{D_A}(X_k = 0) = 0.99$. We show that releasing even just the total table sum, $\sum_{k=1}^{N} X_k = N/2$ would cause the attacker's posterior beliefs to change significantly from his prior beliefs. See the Appendix for a proof.

Thus with such a strong privacy requirement that quantifies over all possible attackers, no table would ever be released. We take an opposing view. We view the distribution D as a common-sense general knowledge distribution of the private attributes of individuals as a function of their publicly known attributes. For example it could be the distribution of heights as a function of age and gender. We want an attacker who does not have common-sense knowledge (D) to learn D — the goal of releasing data is precisely to allow such a handicapped user to gain common sense. We therefore only impose the requirement on the ratio of posterior to prior probabilities on the attacker who already knows D. We justify this by showing that such an attacker is in some sense the strongest.

Strongest Attacker: Consider two attackers A and A' where attacker A knows the distribution that generates the data, i.e., $D_A = D$, and where A' does not, i.e., $D_{A'} \neq D$. We next show that even though A''s posterior distribution may be very far from his prior, A' will ultimately know only as much as A in expectation.

We formalize the notion of an attacker's final knowledge as his *Reward*. Let $v = \{v_1, \ldots, v_N\}$ be the true instantiation of the data set $X = \{X_1, \ldots, X_N\}$, and let M be some function of the data set that is released. In our case, M is the contingency table. We define the Reward, R, of an attacker A to be

$$R = \sum_k \log P_{D_A}(X_k = v_k | M)$$

Thus the greater the posterior probability that an attacker associates with the true private value of an individual, the greater his Reward. Then define the expected Reward of an attacker to be his expected Reward over all possible instantiations of X and M drawn according to D and the coin tosses associated with the release of M. We can show the following proved in the Appendix.

Theorem 1 *The expected Reward of an attacker is maximized when $D_A = D$.*

Thus the attacker A for whom $D_A = D$ is in some sense the strongest attacker one could assume. No other attacker could ultimately know more than A in expectation. This gives us a formal justification for only guaranteeing ϵ-semantic privacy against those attackers who know D to begin with.

The Suppressor's Prior: As we shall see in the following sections, the suppression algorithms we design are assumed to have knowledge of the distribution D as well. Given our view of D as a common-sense general knowledge distribution, we feel that this is a reasonable assumption to make since in many situations the suppressor would have access to enough past and present data sets to be able to form a reasonably accurate estimate of D. Of course a natural question then is, why not publish just the distribution D itself. We feel that if the suppressor knows only a well-informed estimate of D, then there is value to releasing the actual instantiation of D as well. Additionally, D may be difficult to describe (the representation of the entire joint distribution of the attributes may be exponentially large), and in such a case, a contingency table would be a succinct representation of D.

5 Algorithms for Cell Suppression

We now give a general framework for constructing simulatable cell suppressors, and then use this framework to construct a suppressor for the special case of Boolean private attributes.

5.1 A General Framework

If we had a procedure to evaluate the posterior probability that any $X_k = v$ given a set of released cell values, the problem of cell suppression would seem quite simple: First, divide the attributes into bins. Then for each cell check whether revealing the cell value in conjunction with row and column sums and other cell values revealed so far causes a significant shift in the posterior probabilities for any individual. If so, suppress the cell value, or else reveal it.

However, as seen in our example from Section 1, if a cell is suppressed by first peeking at its value then the very act of suppressing can reveal information. So to ensure that suppressions do not reveal information, we take care to ensure that suppressions are simulatable, i.e., an attacker should be able to simulate the releasing agency's decision process and predict which cells will be suppressed. Since an attacker can equivalently decide which cells will be suppressed, suppressions provably do not leak information.

Recall (Section 4.2) that a simulatable suppressor should not look at a cell's actual value when deciding whether or not to suppress it. However, it may make use of the underlying probability distribution D from which the data is drawn. This is because the attacker is already assumed to know D. Therefore instead of checking if the actual value of a particular cell is "safe" to release in conjunction with other released cells, our suppressor makes use of D to determine if the cell's value is *likely to be such* that releasing it would cause a privacy breach. If so, the cell is suppressed, otherwise it is revealed. Note that in doing this, the suppressor never actually looks at what the cell's value is before making its decision. It only uses D to estimate the likelihood that the cell's value would be "unsafe".

The algorithm for releasing an ϵ-semantically-private contingency table proceeds in two phases. First, attributes are binned so that row and column sums can be safely released. Then given the bins, the suppressor goes through individual cells choosing to reveal or suppress. There is a chance that our algorithm will not publish a contingency table at all if, for instance, the underlying distribution is very skewed. For example, if there is only one individual in the table with probability $\frac{1}{2}$ of having diabetes, while all the other individuals have 0 probability, then releasing even just the total table sum will breach privacy, since any released value causes a large change in the posterior distribution for the one individual with the positive probability. It is for this reason, for example, that our algorithm would never permit the binning of individuals in to those with or without brown hair and blue eyes in the example from Section 3. We view this full-blown suppression as a desirable outcome — some tables are just not safe to publish.

In the following, we call the cells corresponding to row and column sums, *row cells* and *column cells*, respectively. T is the total number of cells — row cells, column cells as well as interior cells. So $T = (m+1)(n+1)$. ϵ and δ are pre-defined privacy parameters.

Key Subroutine: Safe For each phase of the cell suppression algorithm, we need the following subroutine that determines if releasing a particular set of cell values is "safe". In the algorithm, I is a collection of indices and V are cell values for indices in I that are already revealed. The algorithm simply checks if releasing I and V causes a significant change in the attacker's belief about any individual's private value. Since our algorithms will be simulatable, suppressed cell values do not need to be considered in this subroutine.

Algorithm 1 Safe

```
1: for each $k$ and every possible private value $v$ do
2:     If $|1 - \frac{P_D(X_k=v|I,V)}{P_D(X_k=v)}| > \epsilon$ then return 0
3: end for
4: return 1
```

Binning: The algorithm SafeToSplit is used in any binning strategy. It takes as input indices I, cell values V that have already been revealed and a row/column interval to split. The goal is to determine if making the split and releasing the resulting row/column sums is likely to be a safe decision. In order to do this in a simulatable fashion, the algorithm repeatedly samples data sets according to the distribution D conditioned on I and V. For each sample, it updates I and V assuming the split is made and then calls the subroutine Safe. If the split is safe (i.e., the resulting row/column sums in conjunction with previously released sums is safe) for most sampled data sets then the algorithm returns that it is safe to split. In doing so, observe that the algorithm never looks at the actual row/column sums that would result from the split and thus is simulatable.

Algorithm 2 SafeToSplit

```
1: count = 0
2: for $\frac{T}{\delta} \ln \frac{T}{\delta}$ times do
3:     $S \leftarrow$ Sample a data set according to $I, V, D$
4:     Increment count if the row/column split is not Safe according to $S$
5: end for
6: If (count $\leq \frac{1}{2} \ln \frac{T}{\delta}$) then return safe to split else return unsafe
```

With this subroutine in hand, any binning strategy can be tested. For example, first split the interval of the first attribute at the median probability mass and check if total sums for the resulting two rows are likely to be safe to release. Then given that this split has been made and the actual row sums for the two sides of the split, continue to split each of the two rows similarly until the limit on the total number of rows is reached. And then proceed to make the column splits in the same way. If we cannot safely perform any split, we return that we cannot find a safe binning and do not release a table.

Note that at each point, once a split has been made, we commit to it and future decisions always take into account actual row/column sums for this and past splits. In committing to it, the suppressor may be making a mistake, however we show that with enough sampling, this is an unlikely outcome. No decision is ever based on the actual row/column sum for the split that is currently being considered.

Cell Suppressor: Once the binning is fixed, we go through each of the remaining mn interior cells, in any fixed public order, deciding whether to release or suppress. Once again, this is done in a simulatable fashion by sampling data sets that are consistent with I and V and calculating the fraction of sampled data sets for which releasing the cell value is not safe.

Algorithm 3 Cell Suppressor

```
1: Initially no cell is examined
2: repeat
3:     Pick any cell that has not been examined, e.g., in row-major order
4:     for $\frac{T}{\delta} \log \frac{T}{\delta}$ times do
5:         $S \leftarrow$ Sample a data set according to $I, V, D$
6:         Increment count if the cell value is not Safe to release according to $S$
7:     end for
8:     If (count $\leq \frac{1}{2} \ln \frac{T}{\delta}$) then release cell value, update $I$ and $V$ accordingly, else
       suppress
9: until all cells examined
```

At Step 6, the algorithm calls the subroutine Safe with I and V temporarily updated to include the value of the cell in consideration for the data set S, just as in SafeToSplit.

We can now show the following about released contingency table.

Theorem 2 *With probability at least $1 - \delta$, the contingency table produced by Algorithm 3 is ϵ-semantically-private.*

The proof of this Theorem can be found in the appendix. We call the suppressor an (ϵ, δ)-suppressor.

5.2 Special Case: Boolean Private Attributes

With this general framework in hand, we now focus on the special case of Boolean private attributes, where each X_k is either 1 or 0. Even though we focus on this case, note that the framework from Section 5.1 can be used more generally for any bounded-range X_k and any distribution D, with the same privacy guarantees. If X_k is continuous instead of discrete, the privacy definition and framework can be suitably modified.

Now in order to use the framework for a particular scenario, we need a way to compute Safe — specifically, Line 2 requires estimating the probability $X_k = 1$ or 0 given previously released cells. We also need a way to sample a data set according to I, V, D (Line 3 of Algorithm 2 and Line 5 of Algorithm 3).

Note that *both* these problems can be solved if we have a way to sample data sets consistent with a set of cell values according to the underlying distribution D. Since we assume that $X_k \in \{0, 1\}$, sampling a data set is equivalent to sampling an integer vertex of a convex polytope defined by the sum constraints on the X_ks according to D (the matrix of sum constraints is totally unimodular).

We consider specific distributions D since there are no known algorithms for sampling integer vertices from arbitrary distributions. Specifically, we focus on the independent

Bernoulli case, where each X_k is 1 with probability p_k and 0 with probability $1 - p_k$. Intuitively each individual can be viewed as being of a particular *type* based on his non-private attributes, and p_k represents the probability that an individual of this type has private value 1. We note that while this assumption on the distribution is reasonable as a starting point, in reality, dependencies do exist between individuals.

We exploit the fact that the polytope is defined by a contingency table in order to obtain polynomial time sampling algorithms thereby completing the description of our suppressor for this Boolean case. Errors stemming from sampling errors can be incorporated in to the bounds of Theorem 2 [55].

In the rest of this section, we introduce some useful notation and then consider two interesting sampling problems. In the event that the X_k's are identically distributed Bernoulli variables, we give a dynamic programming algorithm similar to [32] for drawing a sample. In the event that the p_k's could be different, we show how to sample a data set by building upon Jerrum et al.'s algorithm [48] for sampling perfect matchings.

Note on practicality: The algorithms that we describe all run in polynomial time. For the theoretical proof of privacy to go through, the degree of this polynomial may be large. How does the running time of our algorithm compare to existing work? It is important to note that previous work gives heuristic algorithms for an inherently NP-hard problem. The techniques used are (mixed) integer programs. From a theoretical perspective, these algorithms do not even run in polynomial time unless P=NP. Yet, in practice, these integer programs are able to produce suppressed tables. We could similarly use heuristics to improve the running time of our algorithms. If the solutions we suggest were to be used in practice, then the high-degree polynomial could be overcome by, for example, prematurely stopping the mixing process suggested in [48]. Our solution also has the added benefit of providing stronger privacy guarantees.

Preliminaries: We start with some useful notation. Suppose that during some stage of the suppression, the data has been divided in to m rows and n columns with row and column sums $\vec{r} = \{r_1, \ldots, r_m\}$ and $\vec{c} = \{c_1, \ldots, c_n\}$. Suppose that each cell (i, j) contains b_{ij} individuals and values $V(i, j)$ for some cells are known. Define lower and upper bounds on each cell:

$$l_{ij} = \begin{cases} V(i,j) & \text{if cell } (i,j) \text{ is known} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{ij} = \begin{cases} V(i,j) & \text{if cell } (i,j) \text{ is known} \\ b_{ij} & \text{otherwise} \end{cases}$$

Tighter bounds on cells may be achievable via Frechet bounds [37]. However, these bounds are implicit in the row and column sum constraints described below. Let $k \in [1, N]$ index an individual. Then we need to sample data sets $X = \{X_1, \ldots X_N\}$ according to D such that

$$\sum_{k:k\in row\ j} X_k = r_j,\ \forall j \in [m] \quad \sum_{k:k\in col\ j} X_k = c_j,\ \forall j \in [n] \tag{1}$$

$$l_{ij} \leq \sum_{k:k\in cell(i,j)} X_k \leq u_{ij},\quad \forall i,j \in [m]\times[n] \tag{2}$$

We now describe two different sampling algorithms.

5.2.1 Dynamic Programming

Let us first consider the case where every $p_k = p$, i.e., all individuals have the same prior probability of having $X_k = 1$. In Section 5.2.2 we shall tackle the more general case of distinct p_ks. In this case it suffices to sample data sets satisfying Equations 1 and 2 uniformly at random. This is because every consistent data set will have the same number of 0s, say j, and the same number of 1s, $N-j$, and therefore probability mass proportional to $p^j(1-p)^{N-j}$. Thus each consistent data set is equally likely.

In order to sample consistent data sets uniformly at random, it suffices to (1) sample a contingency table, $M \in \mathbb{N}^{m+1\times n+1}$, consistent with cell bounds and known cell values with probability proportional to $\prod_{(i,j)\in[m]\times[n]}\binom{b_{ij}}{M_{ij}}$ and then (2) sample an underlying data set for this table uniformly at random.

(2) is done easily enough by setting any M_{ij} individuals in each cell (i,j) to 1 and the others to 0. For (1), prior work in this area [17] focuses on sampling consistent contingency tables uniformly at random and we cannot use these results directly. Instead, we give initial solutions to this new problem of sampling tables from a nonuniform distribution. We illustrate a dynamic programming approach that runs in polynomial time when the number of rows in the contingency table is a constant. The algorithm is similar to known dynamic programming algorithms in the literature [32], but modified to enable sampling from the desired distribution.

We wish to sample a table $M \in \mathbb{N}^{m\times n}$ with probability proportional to $\prod_{(i,j)\in[m]\times[n]}\binom{b_{ij}}{M_{ij}}$ such that

$$\sum_{j=1}^{n} M_{ij} = r_i,\ \forall i \in [m] \quad \sum_{i=1}^{m} M_{ij} = c_j,\ \forall j \in [n]$$

$$\text{and } l_{ij} \leq M_{ij} \leq u_{ij},\ \forall i,j \in [m]\times[n]$$

Let $\mathcal{M}_j = \{M_j \in \mathbb{N}^m : \sum_{i=1}^m M_{ij} = c_j, l_{ij} \leq M_{ij} \leq u_{ij}\}$, i.e., $\mathcal{M}_j$ is the set of all possible assignments of the column sum c_j to cells in the jth column. $|\mathcal{M}_j|$ is at most $\min\{\binom{c_j+m-1}{m-1}, \prod_{i=1}^m b_{ij}\}$. Let $N_j = \sum_{k=1}^j c_j$, i.e., the total number of 1s in the first j columns.

Now the dynamic programming based sampling algorithm works by building a table of counts F for every possible set of partial row sums for every column. Here a set of partial row sums for column j is a vector $\vec{pr} \in \mathbb{N}^m$ specifying a possible sum for the first j columns of each row. Note that for column j there could be at most N_j^{m-1} possible partial row sum vectors. The count stored with each table entry for a partial row sum vector $\vec{pr}$ and a column j is essentially the number of possible assignments of 0s and 1s to the x_ks that fall in the first j columns that could have resulted in the partial row sums $\vec{pr}$. This count is computed recursively:

$$F(\vec{pr}, j) = \sum_{M_j \in \mathcal{M}_j} F(\vec{pr} - M_j, j-1) \times \prod_{i=1}^{m} \binom{b_{ij}}{M_{ij}}$$

$$F(\vec{pr}, 1) = \prod_{i=1}^{m} \binom{b_{i1}}{pr_i}$$

Once the table is built, $F(\vec{r}, n)$ thus contains a count of the total number of data sets consistent with the cell values. Given the table F, we can now use a backtracking approach to sample a contingency table according to the distribution D conditioned on the cell values. In particular, working backwards, we sample a possible $M_n \in \mathcal{M}_n$ with probability $\prod_{i=1}^{m} \binom{b_{in}}{M_{in}} F(\vec{r} - M_n, n-1)/F(\vec{r}, n)$ and so on, sampling each subsequent M_j with probability

$$\prod_{i=1}^{m} \binom{b_{ij}}{M_{ij}} F(\vec{r} - \sum_{k=j}^{n} M_k, j-1)/F(\vec{r} - \sum_{k=j+1}^{n} M_k, j).$$

Thus a table $M = \{M_1 \ldots M_n\}$ will be sampled with probability $\prod_{i,j} \binom{b_{ij}}{M_{ij}}/F(\vec{r}, n)$ exactly as required.

The time taken to build F is $O(nN^{2m})$ and then the sampling also takes $O(nN^m)$ time. The algorithm is thus polynomial in N, which could be quite large. It may be possible to use techniques from [32] to further reduce the running time to being polynomial in the number of columns, thus greatly improving the practicality of our algorithms, and we highlight this as a very interesting avenue for future research.

Note also, that there isn't a simple way to extend this algorithm when the p_ks could be different. In this case, computing the weighted count of data sets that satisfy partial row sum constraints (that needs to be stored with each table entry in F) could itself take an exponential time. Instead for this case we use a different approach, described next. But first we summarize the results of this section.

Theorem 3 *A contingency table $M \in \mathbb{N}^{m+1 \times n+1}$ consistent with a set of row/column sums and cell bounds can be sampled with probability proportional to $\prod_{(i,j)\in[m]\times[n]} \binom{b_{ij}}{M_{ij}}$ in $O(nN^{2m})$ time.*

5.2.2 Sampling Integral Flows

In the case where the X_ks are independent Bernoulli variables, but not identically distributed, we directly try to sample a consistent data set X according to the distribution D, instead of going through the intermediate stage of sampling a consistent contingency table first. Recall that we wish to sample X satisfying Equations 1 and 2 with probability proportional to $\prod_{k:X_k=1} p_k \prod_{k:X_k=0}(1-p_k)$.

We first reduce the problem to one of sampling an integral max flow in a graph $G = (V, E)$ and from there to sampling perfect matchings in a bipartite graph $\hat{G} = (\hat{V}, \hat{E})$.

Constructing G: For every row in the contingency table, create a node u_i, $i \in [m]$. For every column in the contingency table, create a node w_j, $j \in [n]$. Now for each individual in the table, create a node x_k. If individual k comes from cell (i, j) whose value is not known, then direct an edge with capacity 1 from u_i to x_k and an edge with capacity 1 from x_k to w_j. On the other hand, if the value for cell (i, j) is known to be $V(i, j)$, then create a node y_{ij} and direct an edge with capacity 1 from each x_k that belongs to the cell to y_{ij}, followed by $V(i, j)$ edges with capacity 1 from y_{ij} to w_j. The y_{ij} node thus ensures that at most $V(i, j)$ flow can pass through the individuals who come from cell (i, j) if its value is already known. Now create a source node, s and a sink node t. And direct r_i edges with capacity 1 from s to each u_i; c_j edges with capacity 1 from each w_j to t. Then the following is clear

Lemma 1 *A maximum 0,1 flow f in G corresponds to a data set that satisfies the row and column sum constraints of the contingency table, and that respects the upper bounds on the cell values of the contingency table.*

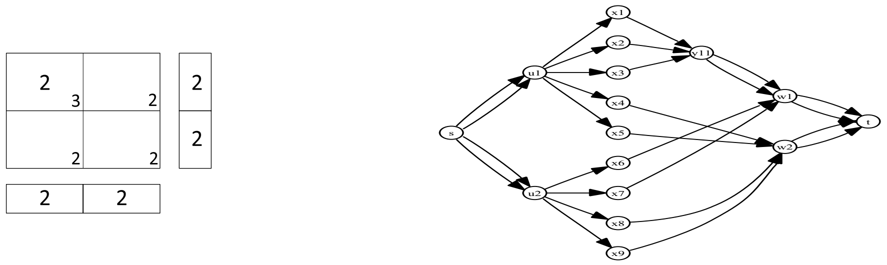
Figure 2 shows an example contingency table and corresponding graph. The number of individuals in each cell of the contingency table is indicated at the bottom right hand corner of the cell. The value for the first cell is known to be 2, and all the row and column sums are 2 each.

We would therefore like to sample a maximum flow 0,1 flow in G with probability proportional to

$$\prod_{(u_i,x_k):f(u_i,x_k)=1} p_k \prod_{(u_i,x_k):f(u_i,x_k)=0} 1 - p_k.$$

There is a reduction from the problem of sampling integral flows in a graph to sampling perfect matchings in a bipartite graph that can be found in [48]. Our reduction deviates from this to ensure that (1) we sample only maximal integral flows, (2) for a cell whose value is already known, *exactly* that much flow passes through individuals belonging to the cell and (3) we sample flows from the required probability distribution.

Constructing $\hat{G}$: We first direct $\sum_{i=1}^{m} r_i$ edges with capacity 1 from t to s and then create an undirected bipartite graph $\hat{G}$ as follows: For the ith edge between



(a) Contingency Table (b) Graph G

Figure 2: An example contingency table and corresponding G

nodes $v_j, v_k \in V$, we create nodes h^i_{jk}, m^i_{jk} and t^i_{jk} together with edges (h^i_{jk}, m^i_{jk}) and (m^i_{jk}, t^i_{jk}). Additionally for each $v_j \in V$ with out-degree $d_G(v_j)$, we create nodes $a^1_j, \ldots a^{d_G(v_j)}_j$. And then edges $\{(a^l_j, h^i_{jk}), (t^i_{jk}, a^{l'}_k)\}_{l,i,l'}$.

Now a 0,1 flow f from s to t in G corresponds to a set of perfect matchings $\mathcal{M}$ in $\hat{G}$ in the following way: If the ith edge between nodes v_j and v_k has a flow of 1 in f, add the edge (h^i_{jk}, m^i_{jk}) to $\mathcal{M}$, or else add the edge (m^i_{jk}, t^i_{jk}). Now for each $v_j \in V$, note that the set of vertices $\{h^i_{jk}\}_{k,i} \cup \{t^{i'}_{k'j}\}_{k',i'}$ consists of exactly $d_G(v_j)$ unmatched vertices which can be paired in $d_G(v_j)!$ ways with the unmatched set of vertices $\{a^l_j\}_l$. Thus the flow f in G corresponds to $\prod_{v \in V} d_G(v)!$ perfect matchings in $\hat{G}$ and every perfect matching in $\hat{G}$ corresponds exactly to a 0,1 flow in G.

Now to ensure that we only sample matchings corresponding to maximum 0,1 flows, we make the following modification to the graph: Let $\{a^l_s\}_l$ be the nodes corresponding to the source node s in G. We remove all edges of the form $\{(a^l_s, h^{l'}_{sj})\}_{l,l'}$ from $\hat{G}$. This restricts us to the set of matchings whose corresponding flows have a flow of 1 on all the edges from s to the row nodes in the graph G. Also to ensure that exactly $V(i,j)$ flow passes through the individuals in a cell (i,j) whose cell value is already known, we remove all edges of the form $\{(a^l_{y_{ij}}, h^{l'}_{y_{ij}})\}_{l,l'}$ from $\hat{G}$.

To ensure that we sample a $0,1$ flow with the appropriate probability, we associate a weight of p_k with each (h_{ik}, m_{ik}) and a weight of $1 - p_k$ with each (m_{ik}, t_{ik}) edge in $\hat{G}$ where i and k correspond to nodes u_i and x_k in G. With every other edge in $\hat{G}$, we associate a weight of 1. Our task is then to sample a perfect matching on $\hat{G}$ with probability proportional to the product of the weights of the edges in the matching. [48] provides a *fully polynomial almost W-generator* for this purpose, i.e., a randomized algorithm that given as inputs a bipartite graph, $\hat{G}$, a probability distribution over matchings, W, and a bias parameter $\tau \in (0,1]$ outputs a random perfect matching on $\hat{G}$ from a distribution W' that is at most τ away from W in total variation distance.

The algorithm runs in time polynomial in N and $\log \tau^{-1}$ and it works only if the weight associated with each matching in W can be computed as the product of weights on its edges, which is true in our reduction. Thus:

Theorem 4 *There exists a fully polynomial almost D-generator for sampling Boolean data sets X satisfying Equations 1 and 2.*

6 Utility

So far, we have only discussed privacy, but it is also important to consider utility. Without a careful study of utility, it is possible that suppressors could be overzealous, suppressing every cell, interior or otherwise.

In this section, we ask the important question: what kinds of contingency tables require few suppressions? This question is of independent interest, providing insights in to how individuals should be grouped together to minimize suppressions. A secondary question is, in tables that require few suppressions, does our algorithm actually suppress a small number? We provide preliminary answers to both questions.

Once again we consider the case of independent Bernoulli X_ks. For the first question we prove the following theorem. At a high level it states that if each cell in the table contains "sufficiently many" individuals and if the p_ks for these individuals are bounded away from 0 or 1, the table will be safe to release as such without any suppressions with high probability.

Theorem 5 *Consider an $(m+1) \times (n+1)$ contingency table, where each interior cell in the table contains at least N_0 people. Suppose every individual in the table has p_k in the range $(\Delta, 1-\Delta)$ where $0 < \Delta \leq 0.5$. Let $\rho = \Delta \ln(1+\epsilon)$. Then releasing every cell value in the table preserves ϵ-semantic-privacy of every individual with probability at least $(1 - (\frac{e^{\rho}}{(1+\rho)^{1+\rho}})^{N_0\Delta} - e^{-\rho^2 N_0 \Delta/2})^{nm}$.*

In general, the greater the number of people in each cell, and the higher the variance of each X_k, the more likely it is that revealing cell values is safe. The variance of an X_k is given by $p_k(1-p_k)$ and is maximized when $p_k = \frac{1}{2}$. Thus a table with 100 interior cells, each cell containing 10,000 people, each with $p_k = \frac{1}{2}$ can be released as such, without any suppressions. With at least 99% probability the table will be 0.2-semantically-private.

To answer the second utility question about the suppressor that we designed, we begin by considering the case of a table with just one cell. This is an admittedly trivial question — if there is only one sum, it ought to be easy to establish when our algorithm will release the sum. We conjecture that the result extends to multicell tables, and provide experimental evidence below.

Theorem 6 *Consider a (ϵ, δ)-suppressor that seeks to release a table with a single cell containing N people and the cell value for that cell. Suppose further that each person* *in the cell has p_k between $(\Delta, 1-\Delta)$, $0 < \Delta \leq 0.5$. Let $\rho = \Delta \ln(1+\epsilon)$. Then if $(\frac{e^\rho}{(1+\rho)^{1+\rho}})^{N\Delta} + e^{-\rho^2 N\Delta/2} < \frac{\delta}{\ell}$, the (ϵ, δ)-suppressor will release the cell value with probability at least $1 - (\frac{e^{\ell/2-1}}{(\frac{\ell}{2})^{\ell/2}})^{1/\ell \ln 1/\delta}$.*

To interpret this theorem, consider a table containing 100,000 people, each of whom has p_k in the range $(0.25, 0.75)$. Then a $(0.2, 0.1)$-suppressor will release the table sum almost surely. If there are 10,000 people in the cell and each p_k is in the range $(0.4, 0.6)$, then a $(0.2, 0.1)$-suppressor will release the table sum almost surely as well.

In general, the greater the variance of the private attributes of individuals in the table, and the greater the number of people in the table, the more likely the suppressor is to release the table sum. As we show in the Appendix, the ideas of the analysis can be applied to tables with multiple cells. Overall, both results suggest a binning strategy that groups together many high variance individuals in order to minimize suppressions, exactly as intuition would suggest. Our experiments below confirmed this intuition.

To validate the theoretical results on utility, we experimented with a generated data set of people with different hair and eye colors. In this data set, blonde-haired people had a probability $p_{\text{blonde}} = 0.5$ of possessing a particular gene, black-haired people had a probability $p_{\text{black}} = 0.4$ of possessing the gene and brown-haired people had a probability $p_{\text{brown}} = 0.3$. The table in Figure 3 was the table released by our cell suppression algorithm over this data set for $\epsilon = 0.5$ and $\delta = 0.2$.

	blonde	black	brown	
blue	101	23	22	146
black	32	95	24	151
brown	45	38	70	153
	178	156	116	450

Figure 3: Table released by cell suppression algorithm. Rows correspond to eye color, columns to hair color, and entries to the number of people who satisfy the row and column heading who possess a certain gene. None of the cells are suppressed.

As expected, none of the cells in the table were suppressed since the binning grouped together large numbers of individuals whose p_ks were bounded away from 0 or 1.[1] The experiment thus confirmed our intuition developed from Theorems 5 and 6.

This example also illustrates the kind of case where suppression-based approaches may be preferable to perturbation. Figure 4 shows the result of running the perturbation algorithm of [6] on the hair/eye-color data set for $\epsilon = 0.5$. While the perturbation preserved the accuracy of the row and column sums for the most part, some of the interior cell values are inaccurate. For example, looking at the perturbed table, one might be tempted to conclude that brown-haired, black-eyed individuals are extremely

[1]The number of individuals in the interior cells of the table from left to right and top to bottom were 200, 80, 100, 80, 200, 100, 100, 100, 200.

	blonde	black	brown	
blue	103	20	38	161
black	33	91	1	125
brown	44	42	77	163
	180	153	116	449

Figure 4: Table released by output perturbation algorithm.

unlikely to possess the gene, whereas in reality, this is not true. Drawing the right conclusions might be extremely critical in certain situations.

We further experimented with an even larger similarly generated data-set over 10,000 individuals with p_ks in the range $[0.3, 0.5]$. Our suppression algorithm released a 10×10 table with each interior cell containing between 80 and 200 individuals and once again there were no suppressions. In contrast, the perturbation algorithm when run on the 10×10 table, resulted in 10% of the cells getting distorted by more than 25% of their original values. The results are summarized in Figure 5

	Cell Suppression	Perturbation
Number of Suppressions	0	0
Number of Released Cells	100	100
Fraction distorted by >25%	0	10%

Figure 5: Comparison of Cell Suppression vs Output Perturbation on 10,000 individuals with $p_k s \in [0.3, 0.5]$

What this shows is that cell suppression can provide greater utility than perturbation if many high variance individuals are binned together in the contingency table. A current drawback of the suppression approach, however, is its complexity; the indistinguishability-based privacy definition from the perturbation literature does away with hassles of estimating the posterior and prior beliefs of attackers and allows for elegant and simple algorithms and analyses. Coming up with a similar privacy definition for suppression-based approaches where exact cell values (or query responses, in the auditing case) are provided if at all, is an interesting open problem.

7 Conclusions and Future Work

We uncovered the fundamental issue that cell suppressions can leak information and designed simulatable cell suppressors for private Boolean data with provable privacy guarantees. We also made key contributions to the related problem of query auditing by examining a heretofore unexamined assumption about the prior knowledge of attackers.

Many directions for future work remain. We introduced a new contingency table sampling problem that merits further study. Our results indicate that polynomial time solutions exist, however the sampling techniques that we invoke are not practical, though they have been steadily improving over the years. Improving the running time of our suppression algorithms by faster sampling is a promising direction for future work. Alternatively, in Section 5.2, we also suggested the use of heuristics for faster sampling. A thorough theoretical and experimental evaluation of the efficiency-privacy tradeoff of such heuristic approaches was beyond the scope of this paper, but remains an interesting avenue for future work. For multidimensional tables, the general suppression framework from Section 5.1 continues to work. However, no algorithms are currently known for sampling multidimensional tables.

Our framework for suppression also assumes that the suppressor knows the distribution D from which the data was drawn. While this may be a reasonable assumption in some cases, as argued in Section 4.3, it need not always be true. The data to be published may follow a distribution that is vastly different from the distribution of past data, or there may not even be any past data releases for the suppressor to learn from.

Perhaps most useful, therefore, would be a privacy definition similar to indistinguishability that does away with such assumptions about D and the need for sampling altogether. Does such a definition even exist for privacy-preserving mechanisms that do not add noise?

Besides this, tackling the cell suppression problem in greater generality, e.g., for multiple table releases or dynamic data, is a natural next step.

Acknowledgements

We thank Kobbi Nissim, Bhaskara Marthi and Kamalika Chaudhuri for many discussions and Mary Cryan for guidance through the sampling contingency table literature.

8 Appendix

An Equivalent Definition of Privacy: In our proofs, we often also use the following alternative definition of privacy from [35, 72].

Definition 3 *Let M be a contingency table released by a randomized suppressor. We say that M is* ϵ-private *if for every individual k, with corresponding private value X_k, and for all pairs of distinct values v, v' in the domain of private values:*

$$|1 - \frac{P_D(M|X_k = v)}{P_D(M|X_k = v')}| \leq \epsilon$$

.

The two privacy definitions are related in that ϵ-privacy implies 3ϵ-semantic-privacy and ϵ-semantic-privacy implies ϵ-privacy (folklore). Some of our proofs are more easily expressed with one definition, and we will switch back and forth between the two as if they were the same.

Impossibility Result from Section 4.3: We use the equivalent ϵ-privacy definition defined above. In the example from Section 4.3, the effect of releasing the table sum for someone who knows the data distribution D is:

$$\begin{aligned}\frac{P_D(\sum X_k = N/2|X_i = 1)}{P_D(\sum X_k = N/2|X_i = 0)} &= \frac{\binom{N-1}{\frac{N}{2}-1} p^{\frac{N}{2}-1}(1-p)^{N-1-(\frac{N}{2}-1)}}{\binom{N-1}{\frac{N}{2}} p^{\frac{N}{2}}(1-p)^{N-1-\frac{N}{2}}} \\ &= \frac{1-p_D}{p_D} = 1\end{aligned}$$

Since $p = 1/2$, releasing the sum preserves everyone's privacy according to the distribution D. However, since the attacker's prior belief is far from the actual distribution, the attacker gains knowledge from the table sum — indeed, there is a massive privacy breach for every X_i:

$$\frac{P_{D_A}(\sum X_k = N/2|X_i = 1)}{P_{D_A}(\sum X_k = N/2|X_i = 0)} = \frac{1-p_{D_A}}{p_{D_A}} = 99$$

Proof of Theorem 1: In the following, we will use the notation $\bar{k}$ to indicate the set of indices in $\{1, \ldots, N\}$ that are not k. Let $\mathcal{E}_k^v$ denote the event that element $X_k = v_k$. Let $\mathcal{E}_{\bar{k}}^v$ denote the event that $X_1 = v_1, \ldots, X_{k-1} = v_{k-1}, X_{k+1} = v_{k+1}, \ldots, X_N = v_N$. Then

$$\begin{aligned}
E[R] &= \sum_{v,M} P_D(X=v,M) \sum_k \log P_{D_A}(\mathcal{E}_k^v|M) \\
&= \sum_{k,M,v} P_D(X=v,M) \log P_{D_A}(\mathcal{E}_k^v|M) \\
&= \sum_{k,M,v_k,v_{\bar{k}}} P_D(\mathcal{E}_k^v,M) P_D(\mathcal{E}_{\bar{k}}^v|\mathcal{E}_k^v,M) \log P_{D_A}(\mathcal{E}_k^v|M) \\
&= \sum_{k,M,v_k} P_D(\mathcal{E}_k^v,M) \log P_{D_A}(\mathcal{E}_k^v|M) \sum_{v_{\bar{k}}} P_D(\mathcal{E}_{\bar{k}}^v|\mathcal{E}_k^v,M)
\end{aligned}$$

But $\sum_{v_{\bar{k}}} P_D(\mathcal{E}_{\bar{k}}^v|\mathcal{E}_k^v,M) = 1$. Therefore,

$$\begin{aligned}
E[R] &= \sum_{k,M,v_k} P_D(\mathcal{E}_k^v,M) \log P_{D_A}(\mathcal{E}_k^v|M) \\
&= \sum_k \sum_M P_D(M) \sum_{v_k} P_D(\mathcal{E}_k^v|M) \log P_{D_A}(\mathcal{E}_k^v|M)
\end{aligned}$$

This is maximized when $D_A = D$, for the same reason that the KL divergence of two distributions is minimized when the two distributions are equal.

Proof of Theorem 2: The contingency table is not ϵ-semantically private if at any point the suppressor made a decision (considered a particular split point in the binning stage to be safe or chose to reveal an interior cell value) that violated ϵ-semantic privacy of some individual. Consider the decision made by the suppressor at a particular interior cell (i,j) (the case for row/column splits is similar). Let $p(i,j)$ denote the probability that revealing cell value $V(i,j)$ in conjunction with previously released cell values violates the ϵ-semantic privacy of some individual. The cell suppressor algorithm essentially estimates $p(i,j)$ via multiple draws of consistent cell values according to the distribution D conditioned on previously released cell values. When $p(i,j) > \delta/T$, then by the Chernoff bound, the fraction of sampled cell values that will be considered unsafe is larger than $\delta/2T$ with probability at least $1-\delta/T$. Hence if $p(i,j) > \delta/T$, the cell suppressor would have made the wrong choice and revealed the cell value with probability at most δ/T. If $p(i,j) < \delta/T$, ϵ-semantic privacy of some individual is breached, only if the suppressor chooses to reveal the value for that cell, and even then only with probability at most δ/T. Thus the probability that the suppressor makes a wrong choice at any of the T decision points is less than δ by the union bound.

Proof of Theorem 5: If all interior cell values of the table are released, then the posterior probability of an individual is only affected by the cell value $V(i,j)$ of the cell (i,j) that he belongs to since the private values of individuals in other cells are not correlated. The release of the cell value is safe for an individual k in the cell if $|1 - \frac{P(V(i,j)|X_k=1)}{P(V(i,j)|X_k=0)}| \leq \epsilon$.

Let's consider the case of X_1 in cell $(1,1)$ containing $N_1 > N_0$ individuals. Then

$$|1 - \frac{P(V(1,1)|X_1 = 1)}{P(V(1,1)|X_1 = 0)}| = |1 - \frac{P(\sum_{k=2}^{N_1} X_k = V(1,1) - 1)}{P(\sum_{k=2}^{N_1} X_k = V(1,1))}|$$

If N_1 is large enough, then the sum $X_2 + \ldots + X_{N_1}$ is normally distributed about $\mu = \sum_{k=2}^{N_1} p_k$ with variance $\sigma^2 = \sum_{k=2}^{N_1} p_k(1-p_k)$. So $V(1,1)$ is safe for X_1 if

$$|1 - \frac{e^{-(V(1,1)-1-\mu)^2/\sigma^2}}{e^{-(V(1,1)-\mu)^2/\sigma^2}}| \leq \epsilon \tag{3}$$

Now if $V(1,1)$ lies in the range $|\sum_{k=1}^{N_1} p_k(1-\rho), \sum_{k=1}^{N_1} p_k(1+\rho)|$, where $\rho = \Delta \ln(1+\epsilon)$ then Inequality 3 will be satisfied and the cell value will be safe for all individuals in the cell. By the Chernoff bounds, the probability that $V(1,1)$ lies in this range is at least $1 - (\frac{e^\rho}{(1+\rho)^{1+\rho}})^{N_1\Delta} - e^{-\rho^2 N_1 \Delta/2}$. And similarly, the probability that all interior $V(i,j)$s lie within safe ranges of their expectations is at least $(1-(\frac{e^\rho}{(1+\rho)^{1+\rho}})^{N_0\Delta} - e^{-\rho^2 N_0\Delta/2})^{nm}$.

Proof of Theorem 6: The (ϵ, δ)-suppressor releases the table sum if fewer than $1/2 \ln(1/\delta)$ of the sampled values for the table sum are unsafe. Let S be a sampled table sum. Now S is safe if $\forall k$, $|1 - \frac{P(S|X_k=1)}{P(S|X_k=0)}| \leq \epsilon$.

Let's just consider the case for $k = 1$. Then

$$|1 - \frac{P(S|X_1 = 1)}{P(S|X_1 = 0)}| = |1 - \frac{P(X_2 + \ldots + X_N = S - 1)}{P(X_2 + \ldots + X_N = S)}|$$

If N is large enough, then the sum $X_2 + \ldots + X_N$ is normally distributed about $\mu = \sum_{k=2}^{N} p_k$ with variance $\sigma^2 = \sum_{k=2}^{N} p_k(1-p_k)$. So a sample table sum is safe if

$$|1 - \frac{e^{-(S-1-\mu)^2/\sigma^2}}{e^{-(S-\mu)^2/\sigma^2}}| \leq \epsilon \tag{4}$$

Now if S lies in the range $|\sum_{k=1}^{N} p_k(1-\rho), \sum_{k=1}^{N} p_k(1+\rho)|$, where $\rho = \Delta \ln(1+\epsilon)$ then Inequality 4 will be satisfied and the sample count will be safe. By the Chernoff bounds, the probability that S lies in this range is at least $1-(\frac{e^\rho}{(1+\rho)^{1+\rho}})^{N\Delta} - e^{-\rho^2 N\Delta/2} > 1-\delta/\ell$, and the probability that S is actually unsafe is less than δ/ℓ.

Therefore by the Chernoff bounds, the probability that more than $1/2\ln(1/\delta)$ of the sampled data sets will be unsafe is less than $(\frac{e^{\ell/2-1}}{(\frac{\ell}{2})^{\ell/2}})^{1/\ell\ln 1/\delta}$. And therefore the aggregate count for the table will be released with probability at least $1-(\frac{e^{\ell/2-1}}{(\frac{\ell}{2})^{\ell/2}})^{1/\ell\ln 1/\delta}$.

Extending the analysis to more than one cell: Now if a table has more than one cell, we believe that the above analysis can be extended. The idea is that the vector of cell values (interior as well as row and column cell values) is a linear transformation of the interior cell values. For example in a 3×3 table the vector of cell values is

$$\vec{S} = (V(1,1), V(1,2), V(1,3), V(2,1), V(2,2), V(2,3), V(3,1), V(3,2), V(3,3))$$

where $V(1,3)$ and $V(2,3)$ are the row sums, $V(3,1)$ and $V(3,2)$ are the column sums, $V(3,3)$ is the table sum and the remaining are the interior cell values.

If $\vec{V} = (V(1,1), V(1,2), V(2,1), V(2,2))$ is the vector of interior cell values, then $\vec{S}$ can be written as $A\vec{V}^T$, where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ & \cdots & & \end{bmatrix}$$

$\vec{S}$ is thus a linear transformation of $\vec{V}$. If the number of individuals in each interior cell is large, then $\vec{V}$ is itself normally distributed about

$$\vec{\mu} = (\sum_{k\in(1,1)} p_k, \sum_{k\in(1,2)} p_k, \sum_{k\in(2,1)} p_k, \sum_{k\in(2,2)} p_k)$$

with diagonal covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_{1,1} & 0 & 0 & 0 \\ 0 & \sigma_{1,2} & 0 & 0 \\ 0 & 0 & \sigma_{2,1} & 0 \\ 0 & 0 & 0 & \sigma_{2,2} \end{bmatrix}$$

Here $\sigma_(i,j) = \sum_{k\in(i,j)} p_k(1-p_k)$.

$\vec{S}$ will therefore be normally distributed about $A\vec{\mu}^T$ with covariance $A\Sigma A^T$. In the same way as in Theorem 6, one can then evaluate the range that a sampled cell value, in conjunction with previously released cell values would need to lie in order to be safe, and then evaluate the probability that a sampled cell value will actually lie in that range.

Intuitively it would seem that if all cells have a large number of people with high variance, sampled cell values will lie within safe range of expectations and once actual cell values have been released, these too will lie within safe range of expectations, and so on.

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] G. Aggarwal, T. Feder, K.Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and An Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory*, 2005.

[3] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory*, 2005.

[4] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2005.

[5] D. Applegate and R. Kannan. Sampling and integration of near log-concave functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 156–163, 1991.

[6] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2007.

[7] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2005.

[8] K. Chaudhuri and N. Mishra. When random sampling preserves privacy. In *Advances in Cryptology (CRYPTO)*, 2006.

[9] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography Conference*, 2005.

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

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

[12] F. Chin and G. Ozsoyoglu. Statistical database design. *ACM Trans. Database Syst.*, 6(1):113–139, 1981.

[13] L. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75:377–385, June 1980.

[14] L. Cox. Network models for complementary cell suppression. *Journal of the American Statistical Association*, 90(432):1453–1462, 1995.

[15] 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 34th Annual ACM Symposium on Theory of Computing*. ACM Press, 2002.

[16] M. Cryan and M. Dyer. A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. *Journal of Computer and System Sciences*, 67(2):291–310, 2003.

[17] M. Cryan, M. Dyer, and D. Randall. Approximately counting integral flows and cell-bounded contingency tables. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, 2005.

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

[19] D. Denning. Secure statistical databases with random sample queries. *ACM Trans. Database Syst.*, 5(3):291–315, 1980.

[20] P. Diaconis and L. Saloff-Coste. Random walk on contingency tables with fixed row and column sums. Technical report, Harvard University, 1995.

[21] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Annals of Statistics*, 26:363–397, 1998.

[22] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2003.

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

[24] A. Dobra and S. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals. *Statistical Journal of the United Nations ECE*, 18:363–371, 2001.

[25] A. Dobra, S. Fienberg, A. Rinaldo, A. Slavkovic, and Y. Zhou. Algebraic statistics and contingency table problems: Log-linear models, likelihood estimation, and disclosure limitation. In *Emerging Applications of Algebraic Geometry, IMA Series in Applied Mathematics*, pages 63–88. Springer, 2008.

[26] A. Dobra, S. Fienberg, and M. Trottini. Assessing the risk of disclosure of confidential categorical data (with discussion). In *Bayesian Statistics 7*, pages 125–144. Clarendon: Oxford University Press, 2003.

[27] G. Duncan and S. Fienberg. Obtaining information while preserving privacy: a markov perturbation method for tabular data. In *Statistical Data Protection, EUROSTAT*, 1998.

[28] G. Duncan, S. Fienberg, R. Krishnan, R. Padman, and S. Roehrig. Disclosure limitation methods and information loss for tabular data. In *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, pages 135–166. Elsevier, 2001.

[29] C. Dwork and S. Fienberg. Cs-statistics workshop on privacy and confidentiality. 2005.

[30] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

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

[32] M. Dyer. Approximate counting by dynamic programming. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 2003.

[33] 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.

[34] M. Dyer, R. Kannan, and J. Mount. Sampling contingency tables. *Random Structures and Algorithms*, 10(4):487–506, 1997.

[35] 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.

[36] A. Evfimievski, R. Srikant, R. Agarwal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[37] S. Fienberg. Frechet and bonferroni bounds for multi-way tables of counts with applications to disclosure limitation. In *Statistical Data Protection, EUROSTAT*, 1999.

[38] S. Fienberg and U. Makov. Confidentiality, uniqueness and disclosure limitation for categorical data. *Jounral of Official Statistics*, 14:385–397, 1998.

[39] S. Fienberg, U. Makov, and A. Sanil. A bayesian approach to data disclosure. *Journal of Official Statistics*, 13:75–89, 1997.

[40] S. Fienberg, U. Makov, and R. Steele. Disclosure limitation using perturbation and related methods for categorical data (with discussion). *Journal of Official Statistics*, 14:485–511, 1998.

[41] S. Fienberg and A. Slavkovic. Making the release of confidential data from multi-way tables count. *Chance*, 17(3):5–10, 2004.

[42] S. Fienberg and A. Slavkovic. Preserving the confidentiality of categorical statistical data bases when releasing information for association rules. *Data Mining and Knowledge Discovery*, 11:155–180, 2005.

[43] S. Fienberg and A. Slavkovic. A survey of statistical approaches to preserving confidentiality of contingency table entries. In *Privacy Preserving Data Mining: Models and Algorithms*, pages 289–310. Springer, 2008.

[44] M. Fischetti and J. Gonzalez. Models and algorithms for optimizing cell suppression in tabular data with linear constraints. *Journal of the American Statistical Association*, 95(451):916–928, 2000.

[45] M. Fischetti and J. Gonzalez. Partial cell suppression: A new methodology for statistical disclosure control. *Statistics and Computing*, 13(1):13–21, 2003.

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

[47] D. Gusfield. A graph theoretic approach to statistical data security. *SIAM Journal on Computing*, 17(3):552–571, June 1988.

[48] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[49] P. Jonsson and A. Krokhin. Computational complexity of auditing discrete attributes in statistical databases. In *Manuscript*, 2003.

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

[51] R. Kannan, L. Lovasz, and M. Simonovits. Random walks and an $O^*(n5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11, 1997.

[52] M. Kao. Data security equals graph connectivity. *SIAM Journal on Discrete Mathematics*, 9(1):87–100, February 1996.

[53] S. Kasiviswanathan, M. Rudelson, A. Smith, and J. Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, 2010.

[54] J. Kelly, B. Golden, and A. Assad. Cell suppression: Disclosure protection for sensitive tabular data. *Networks*, 22:397–417, 1992.

[55] K. Kenthapadi. *Models and algorithms for data privacy*. Ph.d. thesis, Stanford University, 2006.

[56] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2005.

[57] N. Kirkendall and G. Sande. Comparison of systems implementing automated cell suppression for economic statistics. *Journal of Official Statistics*, 14(4):513–535, 1998.

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

[59] Kristen Riedt Lefevre. *Anonymity in data publishing and distribution*. Ph.d. thesis, University of Wisconsin at Madison, 2007.

[60] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: privacy beyond k-anonymity and l-diversity. In *Proceedings of the IEEE International Conference on Data Engineering*, 2007.

[61] 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*, 2002.

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

[63] L. Lovasz and S. Vempala. Logconcave functions: Geometry and efficient sampling algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.

[64] L. Lovasz and S. Vempala. Simulated annealing in convex bodies and an $O^*(n4)$ volume algorithm. In *Proceedings 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 650–659, 2003.

[65] A. Machanavajjhala and J. Gehrke. On the Efficiency of Checking Perfect Privacy. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2006.

[66] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the IEEE International Conference on Data Engineering*, 2006.

[67] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3, 2007.

[68] B. Malin and L. Sweeney. How (not) to protect genomic data privacy in a distributed network: Using trail re-identification to evaluate and design anonymity protection systems. *Journal of Biomedical Informatics*, 37(3):179–192, 2004.

[69] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004.

[70] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004.

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

[72] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2006.

[73] S. Nabar, K. Kenthapadi, N. Mishra, and R. Motwani. A survey of query auditing techniques for data privacy. In *Privacy Preserving Data Mining: Models and Algorithms*, pages 415–431. Springer, 2008.

[74] S. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards robustness in query auditing. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006.

[75] Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2007.

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

[77] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1998.

[78] A. Sinclair. *Algorithms for Random Generation and Counting, a Markov Chain Approach.* Birkhaüser, 1992.

[79] A. Slavkovic. Dimacs/dydan workshop on data privacy. 2008.

[80] L. Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. In *Proceedings AMIA Annual Fall Symposium*, 1997.

[81] L. Sweeney. Information explosion. In *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*, 2001.

[82] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[83] M. Trottini. A decision-theoretic approach to data disclosure problems. *Research in Official Statistics*, 4:7–22, 2001.

[84] M. Trottini and S. Fienberg. Modeling user uncertainty for disclosure risk and data utility. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 10(5):511–527, 2002.