MODELS AND ALGORITHMS FOR DATA PRIVACY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Krishnaram Kenthapadi
September 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Rajeev Motwani Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Nina Mishra

Approved for the University Committee on Graduate Studies.

Abstract

Over the last twenty years, there has been a tremendous growth in the amount of private data collected about individuals. With the rapid growth in database, networking, and computing technologies, such data can be integrated and analyzed digitally. On the one hand, this has led to the development of data mining tools that aim to infer useful trends from this data. But, on the other hand, easy access to personal data poses a threat to individual privacy. In this thesis, we provide models and algorithms for protecting the privacy of individuals in such large data sets while still allowing users to mine useful trends and statistics.

We focus on the problem of *statistical disclosure control* – revealing aggregate statistics about a population while preserving the privacy of individuals. A *statistical database* can be viewed as a table containing personal records, where the rows correspond to individuals and the columns correspond to different attributes. For example, a medical database may contain attributes such as name, social security number, address, age, gender, ethnicity, and medical history for each patient. We would like the medical researchers to have some form of access to this database so as to learn trends such as correlation between age and heart disease, while maintaining individual privacy. There are broadly two frameworks for protecting privacy in statistical databases. In the interactive framework, the user (researcher) queries the database through a privacy mechanism, which may deny the query or alter the answer to the query in order to ensure privacy. In the non-interactive framework, the original database is first sanitized so as to preserve privacy and then the modified version is released. We study methods under both these frameworks as each method is useful in different contexts.

The first part of the thesis focuses on the interactive framework and provides models and algorithms for two methods used in this framework. We first consider the online query

auditing problem: given a sequence of queries that have already been posed about the data, their corresponding answers and given a new query, deny the answer if privacy can be breached or give the true answer otherwise. We uncover the fundamental problem that query denials leak information. As this problem was overlooked in previous work, some of the previously suggested auditors can be used by an attacker to compromise the privacy of a large fraction of the individuals in the data. To overcome this problem, we introduce a new model called simulatable auditing where query denials provably do not leak information. We also describe a probabilistic notion of (partial) compromise, in order to overcome the known limitations of the existing privacy definition. We then present simulatable auditing algorithms under both these definitions. The second problem we consider is output perturbation, in which the database administrator computes exact answer to the query and then outputs a perturbed answer (by adding random noise) as the response to the query. Inspired by the desire to enable individuals to retain control over their information, we provide a fault-tolerant distributed implementation of output perturbation schemes, thereby eliminating the need for a trusted database administrator. In the process, we provide protocols for the cooperative generation of shares of random noise according to different distributions.

The second part of the thesis focuses on the non-interactive framework and considers two anonymization methods for publishing data for analysis from a table containing personal records. We consider the k-Anonymity model proposed by Samarati and Sweeney, and present approximation algorithms for anonymizing databases. Then we propose a new method for anonymizing data records, where the data records are clustered and then cluster centers are published. To ensure privacy of the data records, we impose the constraint that each cluster must contain no fewer than a pre-specified number of data records. We provide approximation algorithms to come up with such a clustering.

Acknowledgements

I owe my foremost gratitude to my advisor, Rajeev Motwani for providing tremendous guidance and support over the last five years. He has always been a source of inspiration and motivation for me. He has excellent insight at suggesting research problems that are well-motivated and also theoretically challenging. My interactions with him have greatly shaped my perspectives about research. I am especially thankful for his support and encouragement during the initial stage of my PhD program.

I would like to thank my professors at the Indian Institute of Technology, Madras for providing me an excellent undergraduate education. In particular, I thank my undergraduate advisor, C. Pandu Rangan for motivating me to pursue research in the area of algorithms.

I am grateful to Nina Mishra for being a wonderful mentor and collaborator over these years. I thank Hector Garcia-Molina for providing valuable inputs on various topics such as choosing research problems, giving effective talks, and job search. I would like to thank my reading committee members: Rajeev Motwani, Dan Boneh, and Nina Mishra and other members on my orals committee: Hector Garcia-Molina and Amin Saberi. I also wish to thank my other mentors: Kobbi Nissim and Cynthia Dwork.

I thank my officemates over the years, Dilys, Gurmeet, and Rina, for the lively discussions we had over a variety of topics. I would also like to thank Shubha, Gagan, Ying, Sergei, An, Brian, David, Mayur, Utkarsh, Adam, Anupam, Sriram, Ilya, Satish, Vijay, Prasanna, Mayank, Arvind, and all other colleagues who provided a wonderful academic environment at Stanford. I would like to thank Kathi DiTommaso for all the help and feedback she provided from time to time on the progress of the PhD program. I am thankful to the theory admins, Lynda, Maggie, and Wendy for helping me on numerous occasions and for making the theory wing feel like a community.

I thank all my friends, especially my roommate for the last four years, Bhaskaran, for making Stanford life such a great experience. Irrespective of whether they are nearby or far away, they have made my life so colorful.

Above all, I thank my parents and brother Venkatesh who have always been very understanding and encouraging, showering unconditional love and affection upon me all the time. My gratitude to them is beyond expression.

Contents

List of Figures

Chapter 1

Introduction

Over the last twenty years, there has been a tremendous growth in the amount of private data collected about individuals. This data comes from a variety of sources including medical, financial, library, telephone, and shopping records. With the rapid growth in database, networking, and computing technologies, such data can be integrated and analyzed digitally. On the one hand, this has led to the development of data mining tools that aim to infer useful trends from this data. But, on the other hand, easy access to personal data poses a threat to individual privacy. In this thesis, we provide models and algorithms for protecting the privacy of individuals in such large data sets while still allowing users to mine useful trends and statistics.

We focus on the problem of *statistical disclosure control* – revealing aggregate statistics about a population while preserving the privacy of individuals. A *statistical database* can be viewed as a table containing personal records, where the rows correspond to individuals and the columns correspond to different attributes. For example, a medical database may contain attributes such as name, social security number, address, age, gender, ethnicity, and medical history for each patient. It is desirable to provide aggregate knowledge about such databases. For example, if medical researchers have some form of access to this database, they can learn correlations between age (or ethnicity) and the risk of different diseases. Similarly by discovering the occurrence of communicable diseases in a certain area, they can detect the outbreak of an epidemic and thereby prevent it from spreading to other areas. However the use of data containing personal information has to be restricted

in order to protect individual privacy. For instance, a health insurance company can use the above data to increase the insurance premiums of individuals falling under certain profile. Thus there are two fundamentally conflicting goals: *privacy* for individuals and *utility* for data mining purposes. The tension between these goals is evident: we can achieve perfect privacy (but no utility) by refusing to publish any form of the data or answer any queries about the data; we can achieve perfect utility (but no privacy) by publishing the exact data or answering all queries about the data exactly.

1.1 Frameworks for Protecting Privacy

There are broadly two frameworks for protecting privacy in statistical databases. In the interactive framework, the user (researcher) queries the database through a privacy mechanism, which may deny the query or alter the answer to the query in order to ensure privacy. In the non-interactive framework, the original database is first sanitized so as to preserve privacy and then the modified version is released. Interactive framework mainly comprises of two methods: query auditing and output perturbation. In query auditing [KPR03, DN03, KMN05], a query is denied if the response could reveal sensitive information and answered exactly otherwise. In output perturbation [DN03, DN04, BDMN05, DMNS06, DKM$^+$06], the privacy mechanism computes the exact answer to the query and then outputs a perturbed version (say, by adding noise) as the response to the query. The methods under non-interactive framework generally involve perturbation of the data (input perturbation [AS00, AA01, EGS03, AST05]), anonymization (k-Anonymity [Sam01, SS98, Swe02, MW04, AFK$^+$05b]), computing summaries (such as histograms [CDM$^+$05], sketches [MS06], or clusters), or a combination of these [AFK$^+$06]. In input perturbation, the original database is perturbed (say, by adding noise) into a transformed database, which is then released to the users. In k-Anonymity, the identifying fields are removed first and then some of the remaining entries are suppressed or generalized (see Chapter 4) so that for each tuple in the modified table, there are at least $k-1$ other tuples identical to it. Yet another approach, that is somewhat orthogonal to the above classification, is to use techniques from *secure multi-party computation* [Yao86, GMW87, LP02, AMP04, FNP04]. Using these techniques, several parties with private inputs can compute a function of their inputs such

that an adversarial party (or a coalition of parties) cannot learn any information that cannot be deduced from the output of the function and the input of the adversary. For example, two hospitals may want to learn the correlation between age and heart disease across their patient databases without revealing any other information. An overview of some of the privacy methods can be found in [AW89].

1.1.1 Relative Merits of Different Privacy Techniques

We next emphasize that both the frameworks are relevant in different contexts. The results obtained in the interactive framework are expected to be of better quality since only queries of interest to the user are answered, whereas in the non-interactive framework, the user has access to the entire sanitized database and hence can compute answer to any query. For example, under many circumstances, the results obtained using output perturbation are of provably better quality than is possible for non-interactive solutions [DMNS06]. On the other hand, non-interactive methods are preferable whenever the underlying data mining task is inherently ad hoc and the researchers have to examine the data in order to discover data aggregation queries of interest. Moreover the sanitization can be done offline as no interaction with the user is needed. We can also avoid the risk of accidental or intentional disclosure of the sensitive data by deleting the original data or locking it in a secure vault. Further, for all interactive methods, collusion and denial of service are problems of concern. There is an implicit assumption that all users can collude with each other and hence queries from all users are treated as coming from a single user. Consequently any one user has reduced utility. In particular, a malicious user may pose queries in such a way that many innocuous queries are either denied or answered with excessive noise (as the case may be) in the future.

Similarly each of the above methods has its own advantages and disadvantages and depending on the application some method may be better than others. While query auditing and non-interactive methods maintain consistency (*i.e.*, if the same query is posed again, we get the same answer), output perturbation does not. The query auditing method is useful in settings where exact answers to queries are necessary. For example, doctors often require exact answers to queries when designing new drugs. Similarly, among the non-interactive

methods, k-Anonymity method is desirable when we want to draw inferences with 100% confidence. Secure function evaluation is useful when the aggregate function (such as set intersection) is known a priori and there are efficient protocols for computing this function. However, privacy is preserved only to the extent that the output of the function itself does not reveal any private information. The difficult part is to determine the functions that are privacy-preserving in the context of statistical databases.

The first part of the thesis focuses on the interactive framework and provides models and algorithms for query auditing and output perturbation methods. The second part focuses on the non-interactive framework and considers anonymization methods based on k-Anonymity and clustering.

1.2 Privacy Protection in the Interactive Framework

1.2.1 Query Auditing

Consider a data set consisting of private information about individuals. The *online query auditing problem* is: given a sequence of queries that have already been posed about the data, their corresponding answers and given a new query, deny the answer if privacy can be breached or give the true answer otherwise. We uncover the fundamental problem that query denials leak information. This problem was overlooked in previous work. Because of this oversight, some of the previously suggested auditors [Chi86, KPR03] can be used by an attacker to compromise the privacy of a large fraction of the individuals in the data. To overcome this problem, we introduce a new model called *simulatable auditing* where query denials provably do not leak information. We present a simulatable auditing algorithm for `max` queries under the classical definition of privacy where a breach occurs if a sensitive value is fully compromised. Because of the known limitations of the classical definition of compromise, we describe a probabilistic notion of (partial) compromise, closely related to the notion of semantic security. We demonstrate that `sum` queries can be audited in a simulatable fashion under probabilistic compromise, making some distributional assumptions. We describe the above model and algorithmic results (joint work with Nina Mishra and Kobbi Nissim and an earlier version published in [KMN05]) in Chapter 2.

1.2.2 Output Perturbation

Another dimension along which the privacy techniques can be classified is the amount of trust required on the database administrator. The positive results in the privacy literature fall into three broad categories: non-interactive with trusted server, non-interactive with untrusted server – specifically, via *randomized response*, in which a data holder alters her data with some probability before sending it to the server – and interactive with trusted server. In particular, the privacy methods for the interactive framework assume that the database administrator is trusted by the individuals whose private information is contained in the database. Inspired by the desire to enable individuals to retain control over their information (as we contend in [ABG+04]), we provide a *distributed* implementation of the output perturbation schemes described in [DN04, BDMN05, DMNS06], thereby removing the assumption of a trusted collector of data. Such an approach is desirable even from the perspective of an organization such as census bureau: the organization does not have to protect against insider attacks or worry about the high liability costs associated with a privacy breach.

Our implementation replaces the trusted server with the assumption that strictly fewer than one third of the participants are faulty (we handle Byzantine faults). In the above output perturbation schemes, privacy is obtained by perturbing the true answer to a database query by the addition of a small amount of Gaussian or exponentially distributed random noise. Under many circumstances the results obtained are of provably better quality (accuracy and conciseness, *i.e.*, the number of samples needed for correct statistics to be computed) than is possible for randomized response or other non-interactive solutions [DMNS06]. Our principal technical contribution is in the cooperative generation of shares of noise sampled from in one case the Binomial distribution (as an approximation for the Gaussian) and in the second case the Poisson distribution (as an approximation for the exponential). Our model and results (joint work with Cynthia Dwork, Frank McSherry, Ilya Mironov, and Moni Naor) were originally published in [DKM+06] and are described in Chapter 3.

1.3 Privacy Protection in the Non-interactive Framework

1.3.1 K-Anonymity

Next we consider the problem of releasing a table containing personal records, while ensuring individual privacy and maintaining data integrity to the extent possible. As discussed earlier, when the aggregate queries of interest are not known a priori, techniques such as query auditing, output perturbation, and secure function evaluation do not provide an adequate solution, and we need to release an anonymized view of the database that enables the computation of non-sensitive query aggregates, perhaps with some error or uncertainty. Moreover, techniques under non-interactive framework such as input perturbation, sketches, or clustering may not be suitable if one wants to draw inferences with 100% confidence. Another approach is to *suppress* some of the data values, while releasing the remaining data values exactly. We note that suppressing just the identifying attributes, such as name and social security number, is not sufficient to protect privacy. This is because we can still join the table with public databases (such as voter list) and identify individuals using non-identifying attributes, such as age, race, gender, and zip code (also called quasi-identifying attributes). In order to protect privacy, we adopt the k-Anonymity model which was proposed by Samarati and Sweeney [Sam01, SS98, Swe02]. Suppose we have a table with each tuple having only quasi-identifying attributes. In the k-Anonymity model, we suppress or generalize some of the entries in the table so as to ensure that for each tuple in the modified table, there are at least $k-1$ other tuples in the modified table that are identical to it. Consequently, even with the knowledge of an individual's quasi-identifying attributes, an adversary cannot track down an individual's record further than a set of at least k records. In other words, releasing a table after k-anonymization keeps each individual hidden in a crowd of $k-1$ other people. We study the problem of k-Anonymizing a table, with minimum amount of suppression/generalization and provide approximation algorithms for it. We present the algorithms and hardness results (joint work with Gagan Aggarwal, Tomas Feder, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu and originally published in [AFK+05a, AFK+05b]) in Chapter 4.

1.3.2 Anonymity via Clustering

We again consider the problem of publishing data for analysis from a table containing personal records, while maintaining individual privacy. We propose a new method for anonymizing data records, where quasi-identifiers of data records are first clustered and then cluster centers are published. To ensure privacy of the data records, we impose the constraint that each cluster must contain no fewer than a pre-specified number of data records. This technique is more general since we have a much larger choice for cluster centers than k-Anonymity. In many cases, it lets us release a lot more information without compromising privacy. We also provide constant-factor approximation algorithms to come up with such a clustering. We further observe that a few outlier points can significantly increase the cost of anonymization. Hence, we extend our algorithms to allow an ϵ fraction of points to remain unclustered, *i.e.*, deleted from the anonymized publication. Thus, by not releasing a small fraction of the database records, we can ensure that the data published for analysis has less distortion and hence is more useful. Our approximation algorithms for new clustering objectives are of independent interest and could be applicable in other clustering scenarios as well. Our model and approximation algorithms (joint work with Gagan Aggarwal, Tomas Feder, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu) were originally published in [AFK$^+$06] and are described in Chapter 5.

Part I

Privacy Protection in the Interactive Framework

Chapter 2

Privacy-preserving Auditing Algorithms

Consider a data set consisting of private information about individuals. We consider the *online query auditing problem*: given a sequence of queries that have already been posed about the data, their corresponding answers and given a new query, deny the answer if privacy can be breached or give the true answer otherwise. Formally, let $X = \{x_1, \ldots, x_n\}$ be a set of n private values from n individuals, where each x_i is some real value. Suppose that the queries $q_1, \ldots, q_{t-1}$ have already been posed about X and the answers $a_1, \ldots, a_{t-1}$ have already been given, where each a_j is either the true answer to the query or "denied". Given a new query q_t, deny the answer if privacy may be breached, and provide the true answer otherwise. The *classical* definition of compromise is that there exists an index i such that x_i is uniquely determined. In other words, in all data sets consistent with the queries and answers, there is only one possible value of x_i. The kinds of queries considered in this chapter are `sum` and `max` queries. Given a collection of indices $S \subseteq [n]$, $\mathtt{sum}(S) = \sum_{i \in S} x_i$ and $\mathtt{max}(S) = \max_{i \in S} x_i$.

One of the the first auditing results dates back almost 30 years ago to the work of Dobkin, Jones, and Lipton [DJL79]. That work restricts the class of acceptable queries by their size and overlap, and then demonstrates that a data set cannot be compromised provided that the number of queries is appropriately upper bounded. Following that work, many others considered the auditing problem including [Rei79, CO81a, CO81b, Chi86, KPR03] (see also [AW89] for a survey).

In the work of Chin [Chi86], the online max auditing problem is considered. Given an online sequence of max queries, that paper gives a method of determining whether a value has been uniquely determined using only the queries that were exactly answered. Another example of an online auditing algorithm is due to Kleinberg, Papadimitriou, and Raghavan [KPR03]. In the case that the underlying data set is Boolean valued, a method is given for determining whether a value has been uniquely determined – again using only the queries that were exactly answered.

These online auditing algorithms ignore the queries that were denied and this turns out to be quite problematic since denials can leak information. A simple example illustrates the phenomena. Suppose that the underlying data set is real-valued and that a query is denied only if some value is fully compromised. Suppose that the attacker poses the first query $\mathtt{sum}(x_1, x_2, x_3)$ and the auditor answers 15. Suppose also that the attacker then poses the second query $\mathtt{max}(x_1, x_2, x_3)$ 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 $\mathtt{max}(x_1, x_2, x_3) \not< 5$ since then the sum could not be 15. Further, if $\mathtt{max}(x_1, x_2, x_3) > 5$ then the query would not have been denied since no value could be uniquely determined. Consequently, $\mathtt{max}(x_1, x_2, x_3) = 5$ and the attacker learns that $x_1 = x_2 = x_3 = 5$ — a privacy breach of all three entries. The issue here is that query denials reduce the space of possible consistent solutions, and this reduction is not explicitly accounted for in existing online auditing algorithms.

This oversight of previous work is critical: privacy-preserving auditing algorithms that ignore denials [Chi86, KPR03] can be used to launch massive privacy attacks.

2.1 Contributions

Our first contribution is to illustrate how such major privacy violations can ensue from ignoring query denials. In the case of max queries, we illustrate how 1/8 of the data set can be compromised in expectation by allowing the denials of [Chi86] to guide us towards a breach. In the case where the underlying data set is Boolean valued and each query requests the sum of a subset of private values, we demonstrate how the "conservative" approximate online auditing algorithm of [KPR03] (that denies more often than it should) can be used

to compromise about 1/2 of the data set in expectation – again using the denials to guide us towards a breach.

How can we overcome the problem of denials leaking information? The simulation paradigm from cryptography offers an elegant solution that we apply to auditing. We say that an auditor is *simulatable* if an attacker, knowing the query-answer history, could make the same decision as to whether or not a newly posed query will be answered. Since the auditor only uses information already known to the attacker when deciding whether to deny, the attacker can mimic or simulate that decision. Hence, the decision to deny a query provably does not leak any information.

We next give a general method for designing simulatable auditors. The idea is that if an auditor has received queries $q_1, \ldots, q_t$ and given answers $a_1, \ldots, a_{t-1}$, it simply considers many possible answers to the query q_t (obliviously of the actual data) and determines how often privacy would be compromised. If privacy is breached one or many times in these answers, then the query is denied, otherwise the query is answered.

We then give an algorithm for simulatable auditing of `max` queries under classical compromise, where a compromise occurs if a value is uniquely determined. The algorithm runs in time logarithmic in the number of previous queries and linear in the sum of the sizes of the previous queries. Simulatable auditing of `sum` queries follows from previous work.

Next we revisit the definition of compromise. The classical definition of compromise has been extensively studied in prior work. This definition is conceptually simple, has an appealing combinatorial structure, and serves as a starting point for evaluating solutions for privacy. However, as has been noted by many others, e.g., [Bec80, KU77, Dal77, LWWJ02], this definition is inadequate in many real contexts. For example, if an attacker can deduce that a private data element x_i falls in a tiny interval, then the classical definition of privacy is not violated unless x_i can be uniquely determined. While some have proposed a privacy definition where each x_i can only be deduced to lie in a sufficiently large interval [LWWJ02], note that the distribution of the values in the interval matters. For example, ensuring that age lies in an interval of length 50 when the user can deduce that age is between $[-50, 0]$ does not preserve privacy.

In order to extend the discussion on auditors to more realistic partial compromise notions of privacy, we describe an auditing privacy definition that is similar to the notion of

semantic security. Our privacy definition assumes that there exists an underlying probability distribution from which the data is drawn. This is a fairly natural assumption since most attributes such as age, salary, etc. have a known probability distribution. The essence of the definition is that for each data element x_i and every interval J with not too small apriori probability mass, the auditor ensures that the prior probability that x_i falls in the interval J is about the same as the posterior probability that x_i falls in J given the queries and answers. This definition overcomes some of the aforementioned problems with classical compromise.

With this notion of privacy, we describe a simulatable auditor for `sum` queries. The new auditing algorithm computes posterior probabilities by utilizing existing randomized algorithms for sampling from a logconcave distribution, e.g., [LV03]. To guarantee simulatability, we make sure that the auditing algorithm does not access the data set while deciding whether to allow the newly posed query q_t (in particular, it does not compute the true answer to q_t). Instead, the auditor draws many data sets according to the underlying distribution, conditioned on the previous queries and answers. For each of the randomly generated data sets, the auditor computes the answer a'_t to the current query and checks whether revealing this answer would breach privacy. If for most answers the data set is not compromised then the query is answered, and otherwise the query is denied.

The rest of this chapter is organized as follows. In Section 2.2 we discuss related work on auditing. In Section 2.3 we illustrate how denials leak information and give counterexamples to theorems proved in previous auditing work [Chi86, KPR03]. We then introduce simulatable auditing in Section 2.4 and prove that `max` queries can be audited under this definition of auditing and the classical definition of privacy in Section 2.5. Then in Section 2.6 we describe a probabilistic definition of privacy. Finally in Section 2.7 we prove that `sum` queries can be audited under this definition of privacy in a simulatable fashion.

2.2 Related Work

We partition related work into online and offline auditing. In the *offline* auditing problem, one is given a sequence of queries and exact answers and the goal is to determine if a privacy breach has occurred ex post facto. As the initial motivation for work on auditing involves the online auditing problem, we begin with known online auditing results.

2.2.1 Online Auditing

The earliest work is due to Dobkin, Jones, and Lipton [DJL79] and Reiss [Rei79] for the online `sum` auditing problem, where the answer to a query q is $\sum_{i \in q} x_i$. With queries of size at least k elements, each pair overlapping in at most r elements, they showed that any data set can be compromised in $(2k-(\ell+1))/r$ queries by an attacker that knows ℓ values a priori. For fixed k, r and ℓ, if the auditor denies answers to query $(2k-(\ell+1))/r$ and on, then the data set is definitely not compromised. Here the monitor logs all the queries and disallows q_i if $|q_i| < k$, or for some query $t < i$, $|q_i \cap q_t| > r$, or if $i \geq (2k-(\ell+1))/r$.[1] These results completely ignore the answers to the queries. On the one hand, we will see later that this is desirable in that the auditor is simulatable – the decision itself cannot leak any information about the data set. On the other hand, we will see that because answers are ignored, sometimes only short query sequences are permitted (that could be longer if previous answers were used).

The online `max` auditing problem was first considered in [Chi86]. Both the online and offline Boolean `sum` auditing were considered in [KPR03]. We describe these online results in more detail in Section 2.3 and the offline Boolean `sum` auditing work in Section 2.2.2.

Following the publication of [KMN05], the paper [NMK+06] solves the online `max` and `min` simulatable auditing problem under both classical and probabilistic compromise (where both `max` and `min` queries are allowed). An initial study of the utility of auditing is also undertaken.

2.2.2 Offline Auditing

In the *offline auditing* problem, the auditor is given an offline set of queries $q_1, \ldots, q_t$ and true answers $a_1, \ldots, a_t$ and must determine if a breach of privacy has occurred. In most related work, a privacy breach is defined to occur whenever some element in the data set can be uniquely determined. If only `sum` or only `max` queries are posed, then polynomial-time auditing algorithms are known to exist [CO81a]. However, when `sum` and `max` queries

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

are intermingled, then determining whether a specific value can be uniquely determined is known to be NP-hard [Chi86].

Kam and Ullman [KU77] consider auditing subcube queries which take the form of a sequence of 0s, 1s, and *s where the *s represent "don't cares". For example, the query 10**1* matches all entries with a 1 in the first position, 0 in the second, 1 in the fifth and anything else in the remaining positions. Assuming `sum` queries over the subcubes, they demonstrate when compromise can occur depending on the number of *s in the queries and also depending on the range of input data values.

Kleinberg, Papadimitriou, and Raghavan [KPR03] investigate the offline `sum` auditing problem of Boolean data. They begin by proving that the offline `sum` auditing problem is coNP-hard. Then they give an efficient offline `sum` auditing algorithm in the case that the queries are "one-dimensional", *i.e.*, for some ordering of the elements say $x_1, \ldots, x_n$, each query involves a consecutive sequence of values $x_i, x_{i+1}, x_{i+2}, \ldots, x_j$. An offline `max` auditing algorithm is also given. Note that an offline algorithm cannot be used to solve the online problem as illustrated in [KMN05].

In the *offline maximum auditing* problem, the auditor is given a set of queries $q_1, \ldots, q_t$, and must identify a maximum-sized subset of queries such that all can be answered simultaneously without breaching privacy. Chin [Chi86] proved that the offline maximum `sum` query auditing problem is NP-hard, as is the offline maximum `max` query auditing problem.

2.3 Examples where Denials Leak Information

We first demonstrate how the `max` auditor of [Chi86] can be used by an attacker to breach the privacy of 1/8 of the data set. The same attack works also for `sum`/`max` auditors. Then we demonstrate how an attacker can use the approximate online auditing problem of [KPR03] to breach the privacy of 1/2 of the data set. Finally, while all the examples assume that the attacker has the ability to pose queries involving arbitrary subsets of the data, we demonstrate that an attacker who only poses SQL queries can still compromise the data.

Neither [Chi86] nor [KPR03] explicitly state what their algorithms do in the event a query is denied. One interpretation is that once a query is denied, every query thereafter

will also be denied. Under this interpretation, auditors have almost no utility since a denial of service attack can be mounted with a first singleton query, e.g., $q_1 = \texttt{max}(x_3)$ or $q_1 = \texttt{sum}(x_3)$ – such a query will be denied, and so will every other one henceforward. Another interpretation is that once a query is denied, it is treated as if it was never posed. We use this latter interpretation, although some other interpretation may have been intended.

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

2.3.1 max Auditing Breach

Prior to describing the breach, we describe at a high level the max auditing algorithm of [Chi86]. Given a sequence of max queries and corresponding answers, a method is given for compressing the sequence into a short representation that has $O(n)$ size, assuming the private data set consists of distinct values. Each element of the synopsis is a predicate of the form $\texttt{max}(S_i) < M$ or $\texttt{max}(S_i) = M$ where each S_i is a subset of indices. Since there are no duplicates in the data set, the method ensures that query sets of predicates in the synopsis are pairwise disjoint, so that the size of the synopsis is $O(n)$. It suffices to consider just these predicates while checking for privacy breach instead of the entire sequence of past queries. When a new query S_t is posed to the auditor, the real answer to the query M_t is computed and then the synopsis is efficiently updated so that the sets are pairwise disjoint. A breach is shown to occur if and only if there is an S_i such that $|S_i| = 1$ and the synopsis contains a predicate of the form $\texttt{max}(S_i) = M$. In such a case, the query is denied and we assume that the synopsis goes back to its previous state. Otherwise the query is answered and the synopsis remains the same.

In order to breach privacy, the attacker partitions the data set $X = \{x_1, \ldots, x_n\}$ into $n/4$ disjoint 4-tuples and considers each 4-tuple independently. In each of these 4-tuples, the attacker will use the exact answers and the denials to learn one of the four entries, with success probability 1/2. Hence, on average the attacker will learn 1/8 of the entries. Let x_1, x_2, x_3, x_4 be the entries in one 4-tuple. The attacker will first issue the query

$\mathsf{max}(x_1, x_2, x_3, x_4)$. This query is never denied since it is disjoint from all previous queries and knowing the answer will not help to uniquely determine one of these 4 values. Let a be the answer to the query. For the second query, the attacker drops at random one of the four entries, and queries the maximum of the other three. If this query is denied, then the dropped entry equals a. Otherwise, let a' ($= a$) be the answer and drop another random element and query the maximum of the remaining two. If this query is denied, then the second dropped entry equals a'. If the query is allowed, we continue with the next 4-tuple.

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

2.3.2 Boolean Auditing Breach

Prior to describing the Boolean auditing attack, we describe the "conservative" approximate auditor of [KPR03] that denies more often than it should. For each element x_i of the data set, the *trace* of x_i is defined to be the set of queries in which x_i participates. The claim in this paper is that if for each variable x_i there is a corresponding variable x_j such that $x_i \neq x_j$ and x_i and x_j share the same trace, then no value is uniquely determined. The trace only seems to be updated when a query is exactly answered, and not when a query is denied.

Next we demonstrate how an attacker can compromise 1/2 of the data in expectation with the conservative approximate auditor of [KPR03]. In this example, we assume that the data is 1/2 0s and 1/2 1s[2]. The attacker randomly permutes the entries and then partitions the data set $X = \{x_1, \ldots, x_n\}$ into $n/2$ disjoint 2-tuples. The attacker then poses the queries $\mathsf{sum}(x_i, x_{i+1})$ for odd i. If the query is answered then $x_i \neq x_{i+1}$, but the pair is forever ignored. And, if the query is denied, then the attacker can deduce that $x_i = x_j$ and furthermore since the trace is not updated on a denied query, future queries can be posed

[2]The attack can be modified to work for $0 < p < 1$ when there is a p fraction of 0s and $1 - p$ fraction of 1s – and p is not known to the attacker. In such a case, with high probability, the attacker can breach even more data: a $1 - 2p(1 - p)$ ($\geq 1/2$) fraction, in expectation.

about these values. At the end of this process, the attacker has pairs that are equal, but does not know if they are both 0s or both 1s.

Assume without loss of generality that the queries that were denied involved the values $x_1, \ldots, x_m$ where $m = n/2$ in expectation. The attacker now asks queries $\texttt{sum}(x_i, x_{i+1})$ for even i. If such a query is denied, then $x_{i-1} = x_i = x_{i+1} = x_{i+2}$. Otherwise, if the query is answered, then $x_{i-1} = x_i$ are different from $x_{i+1} = x_{i+2}$. At the end of this process, the private values can be partitioned into two sets, one set having all 0s and the other all 1s, but the attacker will not know which side has which value.

To determine this final bit of information, note that in expectation about 1/2 of the previous queries were denied. Let x_j be a value that is different from x_1 that was in a denied query. We ask the final query $x_1 + x_j + x_m$. Note that this query will be answered since $x_1 \neq x_j$ and they have the same trace, and similarly for x_m. Suppose the answer to the query is a. If $x_m = x_1$ then we can determine x_1 by solving for x_1 in the equation $x_1 + (1 - x_1) + x_1 = a$. Otherwise, if $x_m \neq x_1$, then we can solve for x_1 in the equation $x_1 + (1 - x_1) + (1 - x_1) = a$. Once the attacker knows the value of x_1, all the other values (1/2 of the data set in expectation) can now also be uniquely determined.

2.3.3 SQL Queries

Whereas the previous examples assume that the attacker could pose queries about arbitrary subsets of the data, we next show that even if we weakened the attacker by only allowing queries over attributes of the data, data can still be compromised. For example, consider the three queries, `sum`, `count` and `max` (in that order) on the 'salary' attribute, all conditioned on the same predicate involving other attributes. Since the selection condition is the same, all the three queries act on the same subset of tuples. Suppose that the first two queries are answered. Then the `max` query is denied whenever its value is exactly equal to the ratio of the `sum` and the `count` values (which happens when all the selected tuples have the same salary value). Hence the attacker learns the salary values of all the selected tuples whenever the third query is denied. Even though the attacker may not know the identities of these selected tuples, learning the private values of many individuals can still be considered a significant breach.

2.4 Simulatable Auditing

Intuitively, denials leak because users can ask *why* a query was denied, and the reason is in the data. If the decision to allow or deny a query depends on the actual data, it reduces the set of possible consistent solutions for the underlying data.

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

To work around the leakage problem, we make use of the simulation paradigm which is used vastly in cryptography (starting with the definition of semantic security [GM82]). The idea is the following: The reason that denials leak information is because the auditor uses information that is not available to the attacker (the answer to the newly posed query). In particular, this results in a computation the attacker could not perform by himself. A successful attacker capitalizes on this leakage to gain information. We introduce a notion of auditing where the attacker *provably* cannot gain any new information from the auditor's decision. This is formalized by requiring that the attacker is able to simulate or mimic the auditor's decisions. In such a case, because the attacker can equivalently decide if a query would be denied, denials do not leak information.

2.4.1 A Formal Definition of Simulatable Auditing

We begin by formally defining an auditor. We will give two definitions: one for an auditor and another for a randomized auditor that knows the underlying distribution $\mathcal{D}$ from which the data is drawn.

Definition 2.1 *1. An* auditor *is a function of $q_1, \ldots, q_t$ and the data set X that either gives an exact answer to the query q_t or denies the answer.*

2. *A* randomized auditor *is a randomized function of $q_1, \ldots, q_t$, the data set X, and the probability distribution $\mathcal{D}$ that either gives an exact answer to the query q_t or denies the answer.*

Next we define a simulatable auditor. Again, we give two variants depending on whether the auditor is deterministic or randomized.

Definition 2.2 *Let $\mathcal{Q}_t = \langle q_1, \ldots, q_t \rangle$ be any sequence of queries and let $\mathcal{A}_t = \langle a_1, \ldots, a_t \rangle$ be the corresponding answers according to data set X.*

1. *An* auditor B is simulatable *if there exists another auditor B^* that is a function of $\mathcal{Q}_t$ and $\mathcal{A}_{t-1}$, and the outcome of B on $\mathcal{Q}_t, \mathcal{A}_t$ and X is equal to that of B^* on $\mathcal{Q}_t, \mathcal{A}_{t-1}$.*

2. *A* randomized auditor B is simulatable *if there exists another auditor B^* that is a probabilistic function of $\mathcal{Q}_t, \mathcal{A}_{t-1}, \mathcal{D}$, and the outcome of B on $\mathcal{Q}_t, \mathcal{A}_t, \mathcal{D}$ and X is computationally indistinguishable from that of B^* on $\mathcal{Q}_t, \mathcal{A}_{t-1}, \mathcal{D}$.*

All the auditors we design are trivially simulatable since the only information they use is $\mathcal{Q}_t$ and $\mathcal{A}_{t-1}$ (and possibly the underlying distribution $\mathcal{D}$).

A nice 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 the auditor's decision. We thus assume without loss of generality that $q_1, \ldots, q_{t-1}$ were all answered.

Simulatable auditors improve upon methods that completely ignore all previous query answers [DJL79] in that longer query sequences can now be answered (an example is given in Section 2.5.4) and improve upon the use of offline algorithms to solve the online problem since denials do not leak information as shown in [KMN05].

2.4.2 A Perspective on Auditing

We cast related work on auditing based on two important dimensions: utility and privacy (see Figure 2.1). It is interesting to note the relationship between the information an auditor uses and its utility – the more information used, the longer query sequences the auditor can

answer. That is because an informed auditor need not deny queries that do not actually put privacy at risk. On the other hand, as we saw in Section 2.3, if the auditor uses too much information, some of this information may be leaked, and privacy may be adversely affected.

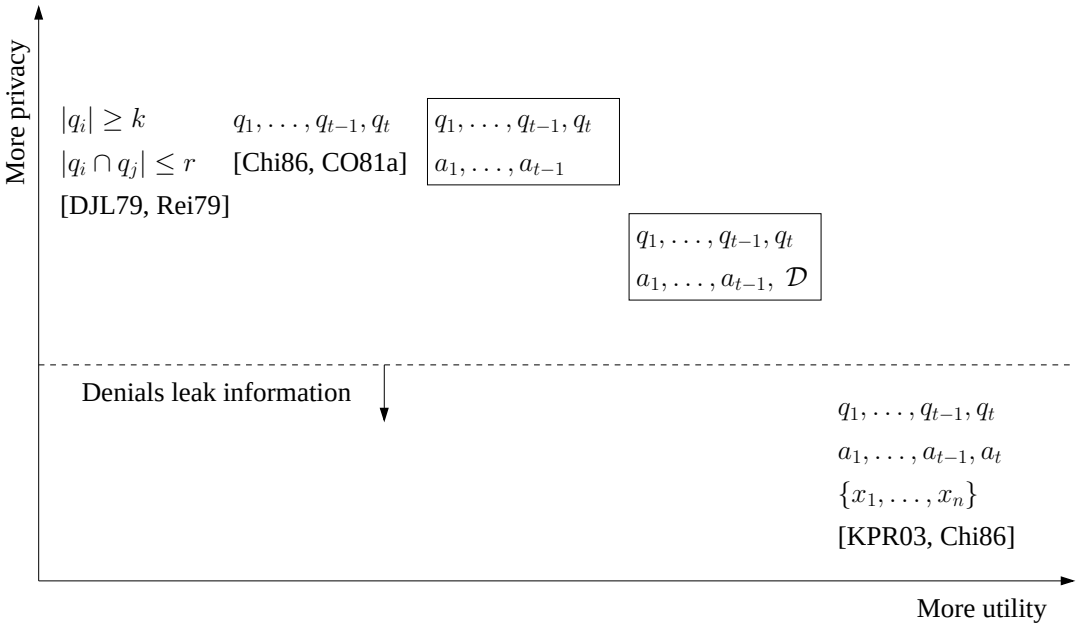


Figure 2.1: Online query auditing approaches

The oldest work on auditing includes methods that simply consider the size of queries and the size of the intersection between pairs of queries [DJL79, Rei79] (upper left hand corner of Figure 2.1). Subsequently, the contents of queries were considered (such as the elementary row and column matrix operations suggested in [Chi86, CO81a]). We call these *monitoring* methods. Query monitoring only makes requirements about the queries, and is oblivious of the actual data entries. In other words, to decide whether a query q_t is allowed, the monitoring algorithm takes as input the query q_t and the previously allowed queries $q_1, \ldots, q_{t-1}$, but ignores the answers to all these queries. This obliviousness of the query answers immediately implies the safety of the auditing algorithm in the sense that query denials cannot leak information. In fact, a user need not even communicate with the data set to check which queries would be allowed, and hence these auditors are simulatable.

Other work on online auditing uses the queries $q_1, \ldots, q_t$ and all of their answers

$a_1, \ldots, a_t$ [Chi86, KPR03] (bottom right corner of Figure 2.1). While this approach yields more utility, we saw in Section 2.3 that denials leak private information.

Our work on simulatable auditing can be viewed as a 'middle point' (denoted in rectangular boxes in Figure 2.1). Simulatable auditors use all the queries $q_1, \ldots, q_t$ and the answers to only the previous queries $a_1, \ldots, a_{t-1}$ to decide whether to answer or deny the newly posed query q_t. We will construct simulatable auditors that guarantee 'classical' privacy. We will also consider a variant of this 'middle point', where the auditing algorithm (as well as the attacker) has access to the underlying probability distribution[3]. With respect to this variant, we construct simulatable auditors that preserve privacy with high probability.

2.4.3 A General Approach for Constructing Simulatable Auditors

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

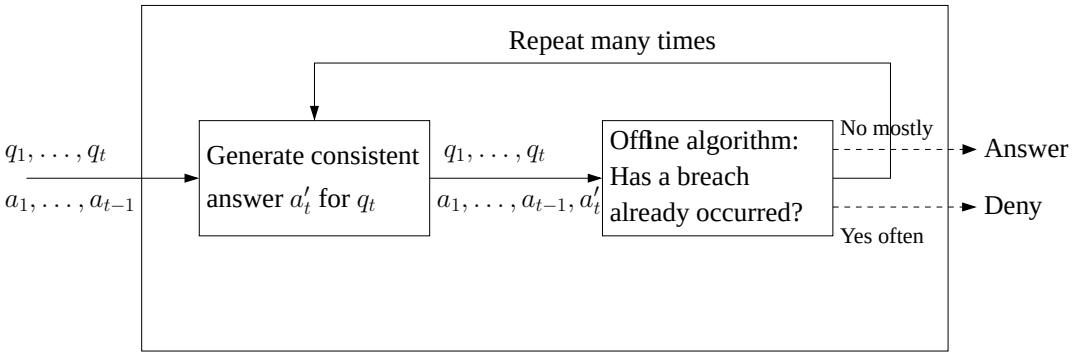


Figure 2.2: General approach for designing simulatable auditors

The general approach (shown in Figure 2.2) works as follows: Choose a set of consistent answers to the last query q_t. For each of these answers, check if privacy is compromised. If compromise occurs for too many of the consistent answers, the query is denied. Otherwise, it is allowed. In the case of classical compromise for `max` simulatable auditing, we deterministically construct a small set of answers to the last query q_t so that if any one leads to compromise, then we deny the answer and otherwise we give the true answer. In

[3]This model was hinted at informally in [DN03] and the following work [DN04].

the case of probabilistic compromise for sum queries, we randomly generate many consistent answers and if sufficiently many lead to compromise, then we deny the query and otherwise we answer the query.

Following the publication of [KMN05], this general approach was also successfully used in [NMK+06].

2.5 Simulatable Auditing Algorithms, Classical Compromise

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

2.5.1 Simulatable Auditing of sum Queries

Observe that existing sum auditing algorithms are already simulatable [Chi86]. In these algorithms each query is expressed as a row in a matrix with a 1 wherever there is an index in the query and a 0 otherwise. If the matrix can be reduced to a form where there is a row with one 1 and the rest 0s then some value has been compromised. Such a transformation of the original matrix can be performed via elementary row and column operations. The reason this auditor is simulatable is that the answers to the queries are ignored when the matrix is transformed.

2.5.2 Simulatable Auditing of max Queries

We provide a simulatable auditor for the problem of auditing max queries over real-valued data. The data consists of a set of n values, $X = \{x_1, x_2, \ldots, x_n\}$ and the queries $q_1, q_2, \ldots$ are subsets of $\{1, 2, \ldots, n\}$. The answer corresponding to the query q_i is $a_i = \texttt{max}\{x_j | j \in q_i\}$. Given a set of queries $q_1, \ldots, q_{t-1}$ and the corresponding answers $a_1, \ldots, a_{t-1}$ and the current query q_t, the simulatable auditor denies q_t if and only if there exists an answer a_t, consistent with $a_1, \ldots, a_{t-1}$, such that the answer helps to uniquely determine some

element x_j. Since the decision to deny or answer the current query is independent of the real answer a_t, we should decide to answer q_t only if compromise is not possible for all consistent answers to q_t (as the real answer could be any of these). Conversely, if compromise is not possible for all consistent answers to q_t, it is safe to answer q_t.

Revisiting the max Auditing Breach of Section 2.3.1

We now return to the max auditing breach example of Section 2.3.1 and describe how a simulatable auditor would work. The first query $\texttt{max}(x_1, x_2, x_3, x_4)$ is always answered since there is no answer, a_1 for which a value is uniquely determined. Suppose the second query is $\texttt{max}(x_1, x_2, x_4)$. This query will always be denied since $x_3 = a_1$ whenever $a_2 < a_1$. In general, under the classical privacy definition, the simulatable auditor has to deny the current query even if only one consistent answer to q_t compromises privacy. Thus, many queries may be denied. This issue is addressed by our probabilistic definition of privacy in Section 2.6.

max Simulatable Algorithm

We now discuss how we obtain an algorithm for max simulatable auditing. A naive solution is to determine if for all possible answers a_t in $(-\infty, +\infty)$ whether (a) a_t is consistent and (b) whether some private element would be uniquely determined if a_t were the answer. Of course, such a naive algorithm is computationally expensive. Instead, we show that it is sufficient to test only a finite number of points. Let $q'_1, \ldots, q'_l$ be the previous queries that intersect with the current query q_t, ordered according to the corresponding answers, $a'_1 \leq \ldots \leq a'_l$. Let $a'_{lb} = a'_1 - 1$ and $a'_{ub} = a'_l + 1$ be the bounding values. Our algorithm checks only $2l + 1$ values: the bounding values, the above l answers, and the mid-points of the intervals determined by them.

We now prove that the algorithm works as desired.

Theorem 2.5.1 *Algorithm 1 is a* max *simulatable auditor that runs in time* $O(t \sum_{i=1}^{t} |q_i|)$ *where* t *is the number of queries.*

We begin by describing how to determine if a private element is uniquely determined (from [KPR03]). Given a set of queries and answers, the upper bound, μ_j for an element

Algorithm 1 `max` SIMULATABLE AUDITOR

Input: (allowed) queries and answers q_i and a_i for $i = 1, \ldots, t-1$, a new query q_t.
Let $q'_1, \ldots, q'_l$ be the previous queries that intersect with q_t, ordered according to the corresponding answers, $a'_1 \leq \ldots \leq a'_l$. Let $a'_{lb} = a'_1 - 1$ and $a'_{ub} = a'_l + 1$.
for $a_t \in \{a'_{lb}, a'_1, \frac{a'_1+a'_2}{2}, a'_2, \frac{a'_2+a'_3}{2}, a'_3, \ldots, a'_{l-1}, \frac{a'_{l-1}+a'_l}{2}, a'_l, a'_{ub}\}$ **do**
    if (a_t is consistent with the previous answers $a_1, \ldots, a_{t-1}$) AND ($\exists 1 \leq j \leq n$ such that x_j is uniquely determined {using [KPR03]}) **then**
        Output "Deny" and **return**
    end if
end for
Output "Answer" and **return**

x_j is defined to be the minimum over the answers to the queries containing x_j, *i.e.*, $\mu_j = \min\{a_k | j \in q_k\}$. In other words, μ_j is the best possible upper bound for x_j that can be obtained from the answers to the queries. We say that j is an *extreme* element for the query set q_k if $j \in q_k$ and $\mu_j = a_k$. This means that the upper bound for x_j is realized by the query set q_k, *i.e.*, the answer to every other query containing x_j is greater than or equal to a_k. The upper bounds of all elements as well as the extreme elements of all the query sets can be computed in $O(\sum_{i=1}^{t} |q_i|)$ time. Since the input includes both the data set and the queries, the time for the above computation is linear in the input size. In [KPR03], it is shown that a value x_j is uniquely determined if and only if there exists a query set q_k for which j is the only extreme element. Hence, for a given value of a_t, we can check if $\exists 1 \leq j \leq n$ such that x_j is uniquely determined.

Next we explain how to determine if an answer to the current query is consistent with the previous queries and answers.

Lemma 2.5.2 *An answer a_t to query q_t is consistent if and only if every query set has at least one extreme element.*

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

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

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

Lemma 2.5.3 *For $1 \leq j \leq n$, x_j is uniquely determined for some value of a_t in (a'_s, a'_{s+1}) if and only if x_j is uniquely determined when $a_t = \frac{a'_s + a'_{s+1}}{2}$.*

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

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

- $q_t = \{j\}$: x_j is breached irrespective of the value of a_t.
- j is the only extreme element of q_t and $|q_t| > 1$: Suppose that x_j is uniquely determined for some value of a_t in (a'_s, a'_{s+1}). This means that each element indexed in $q_t \setminus \{j\}$ had an upper bound $< a_t$ and hence $\leq a'_s$ (since an upper bound can only be one of the answers given so far). Since this holds even for $a_t = \frac{a'_s + a'_{s+1}}{2}$, j is still the only extreme element of q_t and hence x_j is still uniquely determined. A similar argument applies for the converse.
- j is the only extreme element of q'_k for some $k \in [1, l]$: Suppose that x_j is uniquely determined for some value of a_t in (a'_s, a'_{s+1}). This means that $a_t < a'_k$ (and hence $a'_{s+1} \leq a'_k$) and revealing a_t reduced the upper bound of some element indexed in $q'_k \setminus \{j\}$. This would be the case even when $a_t = \frac{a'_s + a'_{s+1}}{2}$. The converse can be argued similarly.

To complete the proof, we will show that values of a_t in (a'_s, a'_{s+1}) are either all consistent or all inconsistent (so that our algorithm preserves consistency when considering only the mid-point value). Suppose that some value of a_t in (a'_s, a'_{s+1}) is inconsistent. Then, by Lemma 2.5.2, some query set q_α has no extreme element. We consider two cases:

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

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

□

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

As noted earlier, the upper bounds of all elements as well as the extreme elements of all the query sets and hence each iteration of the for loop in Algorithm 2.5.2 can be computed in $O(\sum_{i=1}^{t} |q_i|)$ time (which is linear in the input size). As the number of iterations is $2l + 1 \leq 2t$, the running time of the algorithm is $O(t \sum_{i=1}^{t} |q_i|)$, proving Theorem 2.5.1.

2.5.3 Improving the Running Time of the `max` Simulatable Auditor

We have shown that it is sufficient to test a boundable number of answers to the query q_t. A next natural question is can we speed up the algorithm by doing binary search through this set of answers. In particular, does an inconsistent answer to a query imply that all values smaller (or larger) are also inconsistent? It turns out that we can do a form of binary search, although not as simply as the previous question implies. In this section, we show how to improve the running time to $O((\log t) \sum_{i=1}^{t} |q_i|)$ where t is the number of queries[4].

We give four conditions each of which exhibits a monotonicity property. Two of the conditions pin down an interval of consistent answers. The other two conditions pin down an interval of answers where no value can be uniquely determined. If for every consistent answer, no value can be uniquely determined then the query is safe to answer. Consequently we can answer a query if the interval of consistent answers is contained in the interval of answers where no value is uniquely determined.

The four conditions are:

- $\mathcal{A} \equiv$ "$a_t = \theta$ is inconsistent because some query set q_k (different from the current query q_t) has no extreme element"

[4]We thank Robert Tarjan for suggesting this improvement.

- $\mathcal{B} \equiv$ "$a_t = \theta$ is inconsistent because the current query set q_t has no extreme element"
- $\mathcal{C} \equiv$ "For $a_t = \theta$, x_j is uniquely determined for some $j \notin q_t$"
- $\mathcal{D} \equiv$ "For $a_t = \theta$, x_j is uniquely determined for some $j \in q_t$"

Suppose that the $2l + 1$ values to be checked for a_t in step 3 of Algorithm 2.5.2 are arranged in increasing order in an array. As we go from left to right in this array, we will show that condition $\mathcal{A}$ holds for all array elements below some index and does not hold for elements thereafter. Consequently, we can determine the above index by performing a binary search in the array and checking for condition $\mathcal{A}$ at each fork. Condition $\mathcal{C}$ also exhibits monotonicity in the same direction as $\mathcal{A}$. Thus, for a condition $P \in \{\mathcal{A}, \mathcal{C}\}$, let $binarySearch(Left, P)$ denote the index of the leftmost array element for which the condition P does not hold. Conditions $\mathcal{B}$ and $\mathcal{D}$ also exhibit monotonicity, but in the opposite direction. Again, for a condition $P \in \{\mathcal{B}, \mathcal{D}\}$, we use binary search to obtain $binarySearch(Right, P)$ which is the index of the rightmost array element for which the condition P does not hold.

Algorithm 2 FASTER `max` SIMULATABLE AUDITOR

1: $\alpha \leftarrow binarySearch(Left, \mathcal{A})$
2: $\beta \leftarrow binarySearch(Right, \mathcal{B})$
3: $\gamma \leftarrow binarySearch(Left, \mathcal{C})$
4: $\delta \leftarrow binarySearch(Right, \mathcal{D})$
5: **if** $[\alpha, \beta] \subseteq [\gamma, \delta]$ **then**
6: Output "Answer" and **return**
7: **else**
8: Output "Deny" and **return**
9: **end if**

Step 5 of Algorithm 2 checks whether for every consistent answer a_t no value x_j is uniquely determined. The array elements with indices in the range $[\alpha, \beta]$ correspond to consistent answers for the current query whereas those outside this range are inconsistent. Further, there is at least one value of a_t that is consistent, *i.e.*, $\alpha \leq \beta$. Similarly, the array elements with indices in the range $[\gamma, \delta]$ correspond to answers for which no data value is uniquely determined whereas some x_j is uniquely determined for those outside this range.

Hence it is safe to answer the current query if for every consistent answer, no data value is uniquely determined, *i.e.*, the interval $[\alpha, \beta]$ is fully contained in the interval $[\gamma, \delta]$. The running time of the algorithm is $O((\log t) \sum_{i=1}^{t} |q_i|)$, since this is the time taken by each binary search step.

We now prove the main lemma.

Lemma 2.5.4 *1. If $\mathcal{A}$ then any value of $a_t < \theta$ is also inconsistent.*

2. If $\mathcal{B}$ then any value of $a_t > \theta$ is also inconsistent.

3. If $\mathcal{C}$ then x_j is uniquely determined for any value of $a_t < \theta$.

4. If $\mathcal{D}$ then x_j is uniquely determined for any value of $a_t > \theta$.

Proof:

Part 1: Since the sequence of the first $t-1$ queries and answers was consistent and $a_t = \theta$ is inconsistent, it follows from Lemma 2.5.2 that some query set q_k, which had some extreme element(s) earlier, no longer has any extreme element. This means that the extreme element(s) of q_k is no longer extreme for q_k because $a_t = \theta$ reduces the upper bound. In such a case, any value $a_t < \theta$ is also inconsistent.

Part 2: If $a_t = \theta$ is inconsistent because of the current query q_t then the upper bound of every element in q_t is less than $a_t = \theta$. Thus if $a_t > \theta$, q_t still has no extreme element and the answer would still be inconsistent.

Part 3: From [KPR03], because x_j is uniquely determined, we know that there exists a query set q_k (different from q_t) for which j is the only extreme element. This means that $\theta = a_t < a_k$ and revealing a_t reduced the upper bound of some element indexed in $q_k \setminus \{j\}$. This would be the case even for any value of $a_t < \theta$, so that j is still the only extreme element for q_k and hence x_j is still uniquely determined.

Part 4: As in the proof of Lemma 2.5.3, $q_t = \{j\}$ is a trivial case. Hence assume that $|q_t| > 1$ and j is the only extreme element of q_t. This means that each element indexed in $q_t \setminus \{j\}$ has an upper bound less than $a_t = \theta$. This would be the case even for any value of $a_t > \theta$, so that j is still the only extreme element for q_t and hence x_j is still uniquely determined. □

Consequently, we have the following theorem.

Theorem 2.5.5 *Algorithm 2 is a* max *simulatable auditor that runs in time* $O((\log t)\sum_{i=1}^{t}|q_i|)$ *where* t *is the number of queries.*

2.5.4 Utility of max Simulatable Auditor vs. Monitoring

While both simulatable auditors and monitoring methods are safe, simulatable auditors potentially have greater utility, as shown by the following example (see Figure 2.3).

Consider the problem of auditing max queries on a data set containing 5 elements. We will consider three queries and two possible sets of answers to the queries. We will demonstrate that the simulatable auditor answers the third query in the first case and denies it in the second case while a query monitor (which makes the decisions based only on the query sets) has to deny the third query in both cases. Let the query sets be $q_1 = \{1,2,3,4,5\}$, $q_2 = \{1,2,3\}$, $q_3 = \{3,4\}$ in that order. Suppose that the first query is answered as $a_1 = 10$. We consider two scenarios based on a_2. (1) If $a_2 = 10$, then every query set has at least two extreme elements, irrespective of the value of a_3. Hence the simulatable auditor will answer the third query. (2) Suppose $a_2 = 8$. Whenever $a_3 < 10$, 5 is the only extreme element for S_1 so that $x_5 = 10$ is determined. Hence it is not safe to answer the third query.

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

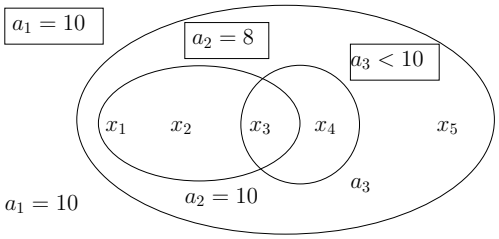


Figure 2.3: max simulatable auditor is more useful than max query restriction auditor. The values within the boxes correspond to the second scenario.

2.6 Probabilistic Compromise

We next describe a definition of privacy that arises from some of the previously noted limitations of classical compromise. On the one hand, classical compromise is a weak definition since if a private value can be deduced to lie in a tiny interval – or even a large interval where the distribution is heavily skewed towards a particular value – it is not considered a privacy breach. On the other hand, classical compromise is a strong definition since there are situations where no query would ever be answered. This problem has been previously noted [KPR03]. For example, if the data set contains items known to fall in a bounded range, e.g., Age, then no `sum` query would ever be answered. For instance, the query $\texttt{sum}(x_1, \ldots, x_n)$, would not be answered since there exists a data set, e.g., $x_i = 0$ for all i where a value, in fact all values, can be uniquely determined.

To work around these issues, we propose a definition of privacy that bounds the change in the ratio of the posterior probability that a value x_i lies in an interval I given the queries and answers to the prior probability that $x_i \in I$. This definition is related to the notion of semantic security [GM82] and to definitions suggested in the perturbation literature including [EGS03, DN03, DN04].

2.6.1 Privacy Definition

Consider an arbitrary data set $X = \{x_1, \ldots, x_n\}$, in which each x_i is chosen independently according to the same distribution $\mathcal{H}$ on $(-\infty, \infty)$. Let $\mathcal{D} = \mathcal{H}^n$ denote the joint distribution. Let the queries be denoted as $q_j = (Q_j, f_j)$, for $j = 1, \ldots, t$ where $Q_j \subseteq [n]$ specifies a subset of the data set entries and f_j specifies a function (such as `sum` or `max`). The answer $a_j = f_j(Q_j)$ is f_j applied to the subset of entries $\{x_i | i \in Q_j\}$.

We next define the notion of λ-*safe*. We say that a sequence of queries and answers is λ-*safe* for an entry x_i and an interval I if the attacker's confidence that $x_i \in I$ does not change significantly upon seeing the queries and answers.

Definition 2.3 *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 entry* x_i *and an interval* $I \subseteq (-\infty, \infty)$ *if the following Boolean*

predicate evaluates to 1:

$$\mathrm{Safe}_{\lambda,i,I}(q_1,\ldots,q_t,a_1,\ldots,a_t) = \begin{cases} 1 & \textit{if } 1/(1+\lambda) \le \frac{\Pr_{\mathcal{D}}(x_i \in I | \wedge_{j=1}^{t}(f_j(Q_j)=a_j))}{\Pr_{\mathcal{D}}(x_i \in I)} \le (1+\lambda) \\ 0 & \textit{otherwise} \end{cases}$$

We say that an interval J is β-*significant* if for every $i \in [n]$ the (a-priori) probability that $x_i \in J$ is at least $1/\beta$. We will only care about probability changes with respect to significant intervals. The definition below defines privacy in terms of a 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 β-significant intervals:

$$\mathrm{AllSafe}_{\lambda,\beta}(q_1,\ldots,q_t,a_1,\ldots,a_t) = \begin{cases} 1 & \text{if } \mathrm{Safe}_{\lambda,i,J}(q_1,\ldots,q_t,a_1,\ldots,a_t)=1, \text{ for every } i \in [n] \text{ and} \\ & \text{every } \beta\text{-significant interval } J \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

We now turn to our privacy definition. Let $X = \{x_1,\ldots,x_n\}$ be the data set, in which each x_i is chosen independently according to the same distribution[5] $\mathcal{H}$ on $(-\infty,\infty)$. Let $\mathcal{D} = \mathcal{H}^n$ denote the joint distribution. Consider the following (λ,β,T)-*privacy game* between an attacker and an auditor, where in each round t (for up to T rounds):

1. The attacker (adaptively) poses a query $q_t = (Q_t, f_t)$.

2. The auditor decides whether to allow q_t or not. The auditor replies with $a_t = f_t(Q_t)$ if q_t is allowed and with $a_t =$ "denied" otherwise.

3. The attacker wins if $\mathrm{AllSafe}_{\lambda,\beta}(q_1,\ldots,q_t,a_1,\ldots,a_t) = 0$.[6]

[5]Our analysis works even if each x_i is chosen independently from a different distribution. However, to simply the notation, we assume that each value is drawn from the same distribution.

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

Definition 2.4 *We say that an auditor is* $(\lambda, \delta, \beta, T)$-private *if for any attacker* $\mathcal{A}$

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

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

Combining Definitions 2.2 and 2.4, we have our new model of simulatable auditing. In other words, we seek auditors that are simulatable and $(\lambda, \delta, \beta, T)$-private.

Note that, on the one hand, the definition of simulatable auditing prevents the auditor from using a_t in deciding whether to answer or deny the query. On the other hand, the privacy definition requires that regardless of what a_t was, with high probability, each data value x_i is still safe (as defined by $\mathrm{AllSafe}_{\lambda,\beta}$). Consequently, it is important that the current query q_t be used in deciding whether to deny or answer. Because we cannot use the answer a_t, but want to use the query q_t, our auditor ignores the real answer a_t and instead makes guesses about the value of a_t obtained by randomly sampling data sets according to the distribution $\mathcal{D}$ conditioned on previous queries and answers.

2.6.2 Evaluating the Predicate $\mathrm{AllSafe}_{\lambda,\beta}$

Equation 2.1 requires checking whether the sequence of queries and answers is λ-safe for infinitely many intervals (*i.e.*, for all β-significant intervals). We next show that all such intervals J can be guaranteed to be λ-safe, by making sure that a *finite* number of intervals are safe.

We assume that the distribution $\mathcal{H}$ is specified such that we can obtain the partition $\mathcal{I}$ of $(-\infty, \infty)$ into ℓ intervals each with equal probability mass of $\frac{1}{\ell}$, *i.e.*, $\Pr_{\mathcal{D}}(x_i \in I) = \frac{1}{\ell}$ for every interval $I \in \mathcal{I}$. For example, if $\mathcal{H}$ is specified as a cumulative distribution function, then we can perform binary search to obtain the points in $(-\infty, \infty)$ that define the above partition to the desired level of precision.

We show that if privacy is preserved in each regularly-weighted interval of probability mass $\frac{1}{\ell}$, then the privacy of any β-significant interval is also preserved. In other words, to guarantee that any β-significant interval J is λ-safe, we will ensure that every interval

$I \in \mathcal{I}$ is $\tilde{\lambda}$-safe where $\tilde{\lambda}$ is smaller than λ. We provide a smooth trade-off: the finer-grained the partition (*i.e.*, larger ℓ), the weaker the privacy requirements (*i.e.*, larger $\tilde{\lambda}$) for each of the ℓ intervals and vice versa. The intuition is the following. The privacy guarantees of the intervals fully contained in J can be used to imply the privacy of J whereas the guarantees of the two bounding intervals (which are partially contained in J) cannot be used. Hence, if the partition is very fine-grained, J contains more intervals from the partition and hence weaker privacy requirements suffice.

Given λ and β, we can choose the trade-off parameter, c to be any integer greater than $1 + 2/\lambda$. Choosing c determines the parameters, $\ell = \lceil c\beta \rceil$ and $\tilde{\lambda} = \frac{\lambda(c-1)-2}{c+1}$. We formally state the lemma below and provide the proof (based on the above intuition) in Section 2.9.1.

Lemma 2.6.1 *Suppose* $\mathrm{Safe}_{\tilde{\lambda},i,I} = 1$ *for every* $i \in [n]$ *and each of the* ℓ *intervals* I *in the partition* $\mathcal{I}$*. Then,* $\mathrm{Safe}_{\lambda,i,J} = 1$ *for every* $i \in [n]$ *and every* β*-significant interval* J *(i.e.,* $\mathrm{AllSafe}_{\lambda,\beta} = 1$*).*

2.7 Simulatable sum Auditing, Probabilistic Compromise

In this section we consider the problem of auditing sum queries (where each query is of the form $\mathtt{sum}(Q_j)$ for a subset of the dimensions, Q_j) under the newly defined probabilistic definition of compromise.

Prior to describing the solution, we give some intuition. Assume for simplicity that each individual can take a value uniformly between $[0, 1]$. Then over n individuals, the data set $\{x_1, \ldots, x_n\}$ can be any point in the unit cube $[0, 1]^n$ with equal probability. A sum query and its corresponding answer induces a hyperplane. The data sets consistent with one sum query/answer are then those points in $[0, 1]^n$ that fall on this hyperplane. Each successive query/answer reduces the space of possible consistent data sets to those in the intersection of the induced hyperplanes that also fall in $[0, 1]^n$, *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.

How can we audit sum queries? Following the general paradigm suggested in Figure 2.2, given a sequence of queries and answers and given a new query q_t, we generate a

consistent answer to q_t (without using the underlying data set). We do this by drawing a point uniformly at random from the convex polytope induced by the previous queries and answers. This point is a sample data set and we then compute a candidate answer to q_t based on this sample data set. Once we have an answer we can then determine whether a privacy breach has occurred: Suppose that P is the current convex polytope. To determine if a breach has occurred for a particular individual x_i and a particular interval I, consider the definition of privacy breach: $\frac{\Pr(x_i \in I | \vec{x} \in P)}{\Pr(x_i \in I)} = \frac{\Pr(x_i \in I | \vec{x} \in P)}{|I|}$. We can estimate the probability in the numerator by sampling from the convex polytope P and estimating what fraction of the sample points lie 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. To increase our certainty, we repeat the above process with many consistent datasets, *i.e.*, many consistent answers to the query q_t. If many consistent answers lead to a privacy breach, we deny the answer and otherwise we give the exact answer. In fact, our algorithm in [KMN05] for auditing `sum` queries under probabilistic compromise uses the above technique.

While this intuition was given in the context of the uniform distribution, there is nothing specific about the uniform distribution that this argument utilizes. Indeed, in the rest of this section, we will assume that the underlying data set is generated from a more general logconcave distribution. We use this distribution because there exist algorithms for approximately sampling from it. However, there is nothing about our general approach that limits its applicability to logconcave distributions.

2.7.1 Properties of Logconcave Distributions

The class of logconcave distributions forms a common generalization of uniform distributions on convex sets and Gaussian distributions. A distribution over a domain D is said to be logconcave if it has a density function f such that the logarithm of f is concave on its support. That is, the density function $f : D \to \mathcal{R}_+$ is *logconcave* if it satisfies $f(\alpha x + (1-\alpha)y) \geq f(x)^\alpha f(y)^{1-\alpha}$ for every $x, y \in D$ and $0 \leq \alpha \leq 1$. These distributions constitute a broad class and play an important role in stochastic optimization and economics of uncertainty and information [LV03, Pre95, An96].

We assume that each element x_i is independently drawn according to the same logconcave distribution H over $\mathcal{R}$. Let $\mathcal{D} = H^n$ denote the joint distribution. Since the product of two logconcave functions is logconcave, the joint distribution $\mathcal{D}$ is also logconcave. Moreover, as the queries and answers impose convex constraints and indicator functions of convex sets are logconcave, the posteriori distribution $\mathcal{D}_{Q,t}$ (which is $\mathcal{D}$ conditioned on $\wedge_{j=1}^{t}(sum(Q_j) = a_j)$) is also logconcave. This is because the density function for the posteriori distribution can be expressed as the product of the density function for the apriori distribution and the indicator function corresponding to the convex constraints (scaled by a constant)[7].

Our algorithms make use of randomized, polynomial-time algorithms for sampling (with a small error) from a logconcave distribution (see for example [LV03]). We will use the following theorem.

Theorem 2.7.1 *There exists an algorithm $Sample(\mathcal{G}, \epsilon)$ for sampling from an arbitrary logconcave distribution $\mathcal{G}$ with running time of $O^*(n^k)$ (for some constant k) such that the sampled output follows a distribution $\mathcal{G}'$ where the total variation distance between $\mathcal{G}$ and $\mathcal{G}'$ is at most ϵ.*

The asterisk in the $O^*()$ notation indicates that (polynomial) dependence on $\log n$, and the error parameter, ϵ are not shown (similar to the notation used in [LV03]). The *total variation distance* between two probability distributions $\mathcal{G}$ and $\mathcal{G}'$ is the largest possible difference between the probabilities assigned to the same event, *i.e.*, $\sup_E |\Pr_{\mathcal{G}}(E) - \Pr_{\mathcal{G}'}(E)|$. The current best known algorithm [LV03] for sampling from a logconcave distribution runs in time $O^*(n^5)$.

2.7.2 Estimating the Predicate $\mathrm{AllSafe}_{\lambda,\beta}$ using Sampling

We begin by considering the situation when we have the answer to the last query a_t. Our auditors cannot use a_t, and we will show how to get around this assumption. Given a_t and all previous queries and answers, Algorithm 3 checks whether privacy has been breached.

[7]The density function for the posteriori distribution, $f_{\mathcal{D}_{Q,t}}$ is proportional to the apriori density, $f_{\mathcal{D}}$ inside the convex region C and is zero outside the region. Denoting the indicator function for the region by δ_C, we get: $f_{\mathcal{D}_{Q,t}}(\cdot) = \frac{f_{\mathcal{D}}(\cdot) \times \delta_C(\cdot)}{\int_C f_{\mathcal{D}}(\vec{x})d\vec{x}}$

As described in Section 2.6.2, it is enough to make sure that each of the ℓ intervals in $\mathcal{I}$ is safe (for a suitable choice of c and hence ℓ in Lemma 2.6.1). For each x_i and each interval I in $\mathcal{I}$, the apriori probability that x_i lies in I is equal to $\frac{1}{\ell}$, from the definition of $\mathcal{I}$. To estimate the posteriori probability that x_i lies in I, we sample many data sets according to the posteriori distribution $\mathcal{D}_{Q,t}$ (which is logconcave) and compute the fraction of the data sets satisfying $x_i \in I$. This is done using Algorithm $Sample$ from Theorem 2.7.1.

Algorithm 3 takes as input, the sequence of queries and answers and the parameters, λ (from Definition 2.3), η (probability of error), c (trade-off parameter from Lemma 2.6.1), β (from the notion of β-significant interval), and n (the size of the data set). $\tilde{\lambda}$ is defined in terms of λ as for Lemma 2.6.1. However, we check the privacy requirement with respect to a smaller parameter, $\lambda' = \tilde{\lambda}/3$. N denotes the number of data sets sampled and N_e denotes the number of data sets satisfying $x_i \in I$. The ratio, $\frac{N_e}{N}$ is an estimate of the posteriori probability that $x_i \in I$. As the apriori probability is equal to $1/\ell$, we require that the ratio, $\frac{\ell \cdot N_e}{N}$ be very close to 1. In Algorithm 3, let $\ell = \lceil c\beta \rceil$, $\tilde{\lambda} = \frac{\lambda(c-1)-2}{c+1}$, $\lambda' = \frac{\tilde{\lambda}}{3}$, $N = \frac{9\ell^2 \ln(2/\eta)}{\tilde{\lambda}^2} \cdot (1+\tilde{\lambda}/3)^2 \cdot \max((1+\tilde{\lambda})^2, (3+\tilde{\lambda}/3)^2)$ and $\epsilon = \frac{\eta}{2N}$. The proof will illustrate why this choice of parameters works.

Algorithm 3 SAFE

Input: Queries and answers q_j and a_j for $j = 1, \ldots, t$, the apriori distribution $\mathcal{D}$, and parameters $\lambda, \eta, c, \beta, n$.
Let safe=*true*
for each x_i and for each interval I in $\mathcal{I}$ **do**
  Sample N data sets according to $\mathcal{D}_{Q,t}$, using Algorithm $Sample(\mathcal{D}_{Q,t}, \epsilon)$
  Let N_e be the number of data sets satisfying $x_i \in I$
  if $\left(\frac{\ell \cdot N_e}{N} \notin \left[\frac{1}{1+\lambda'}, 1+\lambda'\right]\right)$ **then**
    Let safe=*false*
  end if
end for
Return safe

We now prove that Algorithm 3 behaves as desired. Ideally, we would like to prove that if the query is not safe then we deny the query and if the query is safe then we answer the query. However our claims will not be that strong for a few reasons: (a) we do not check all (in fact, infinitely many) β-significant intervals for privacy and instead check only ℓ intervals in the partition (b) we estimate the posteriori probability using sampling and then

use Chernoff bounds and (c) we cannot sample exactly from the underlying logconcave distribution. Consequently, with probability close to 1, whenever $\mathrm{AllSafe}_{\lambda,\beta} = 0$ we deny the query and for a smaller privacy parameter, $\tilde{\lambda}/9$ and larger significance parameter, ℓ, whenever $\mathrm{AllSafe}_{\tilde{\lambda}/9,\ell} = 1$ we answer the query. For the region in between, we make no guarantees.

Lemma 2.7.2 1. *If* $\mathrm{AllSafe}_{\lambda,\beta} = 0$ *then Algorithm* SAFE *returns* false *with probability at least* $1 - \eta$.

2. *If* $\mathrm{AllSafe}_{\tilde{\lambda}/9,\ell} = 1$ *then Algorithm* SAFE *returns* true *with probability at least* $1 - 2n\ell\eta$.

Proof: To simplify the analysis, we will assume that Algorithm $Sample$ always returns a sample from the true distribution. We will first take into account the error due to this assumption. For $1 \leq j \leq N$, let G_j be the distribution followed by the output of Algorithm $Sample(\mathcal{D}_{Q,t}, \epsilon)$ in j^{th} iteration of the algorithm. Then, the variation distance between the distributions, $dist(G_j, \mathcal{D}_{Q,t}) \leq \epsilon$. Using a standard argument (simple hybrid argument), it follows that, the variation distance between the two product distributions, $dist(G_1 \times G_2 \times \ldots \times G_N, \mathcal{D}_{Q,t}^N) \leq N\epsilon = \eta/2$.

We will use the subscript *'real'* to refer to probabilities under the distribution sampled by $Sample$ and *'ideal'* to denote the probabilities under the assumption that $Sample$ returns a sample from the true distribution. Then, for any event E, $\Pr_{real}(E) \leq \Pr_{ideal}(E) + \eta/2$.

We now prove the two parts of the lemma.

Part 1: Using Lemma 2.6.1, it follows that $\mathrm{Safe}_{\tilde{\lambda},i,I} = 0$ for some $i \in [n]$ and $I \in \mathcal{I}$. Let $p_e = \Pr_{\mathcal{D}_{Q,t}}(x_i \in I)$ denote the posteriori probability. From the definition of $\mathcal{I}$, the apriori probability $\Pr_{\mathcal{D}}(x_i \in I) = 1/\ell$.

Suppose $1 + \tilde{\lambda} < \frac{\Pr_{\mathcal{D}}(x_i \in I | \wedge_{j=1}^{t}(sum(Q_j) = a_j))}{\Pr_{\mathcal{D}}(x_i \in I)} = p_e\ell$.

We use Chernoff bounds to show that $\frac{\ell \cdot N_e}{N} > 1 + \lambda'$ with probability at least $1 - \eta$. Let $\theta_1 = \frac{p_e\ell - (1+\lambda')}{p_e\ell} \geq \frac{\tilde{\lambda} - \lambda'}{p_e\ell} = \frac{2\tilde{\lambda}}{3p_e\ell} > 0$. Then,

$$\begin{aligned}\Pr_{ideal}\left(\frac{\ell \cdot N_e}{N} \leq 1+\lambda'\right) &= \Pr_{ideal}(N_e \leq Np_e(1-\theta_1)) \\ &\leq e^{-\frac{Np_e\theta_1^2}{4}} \\ &\leq \eta/2\end{aligned}$$

where the last step is obtained using $N \geq \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}$, so that $\frac{Np_e\theta_1^2}{4} \geq \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2} \cdot \frac{p_e}{4} \cdot \left(\frac{2\tilde{\lambda}}{3p_e\ell}\right)^2 \geq \ln(2/\eta)$. Hence,

$$\begin{aligned}\Pr_{real}\left(\frac{\ell \cdot N_e}{N} \leq 1+\lambda'\right) &\leq \Pr_{ideal}\left(\frac{\ell \cdot N_e}{N} \leq 1+\lambda'\right) + \eta/2 \\ &\leq \eta\end{aligned}$$

Thus the Algorithm SAFE returns *false* with probability at least $1-\eta$.

Now suppose $p_e\ell < \frac{1}{1+\tilde{\lambda}}$. Let $\theta_2 = \frac{1-p_e\ell(1+\lambda')}{p_e\ell(1+\lambda')} > 0$. Using a similar argument as above, we get:

$$\begin{aligned}\Pr_{real}\left(\frac{\ell \cdot N_e}{N} \geq \frac{1}{1+\lambda'}\right) &\leq \Pr_{ideal}\left(\frac{\ell \cdot N_e}{N} \geq \frac{1}{1+\lambda'}\right) + \eta/2 \\ &= \Pr_{ideal}(N_e \geq Np_e(1+\theta_2)) + \eta/2 \\ &\leq e^{-\frac{Np_e\theta_2^2}{4}} + \eta/2 \\ &\leq \eta\end{aligned}$$

where the last step is obtained using $N \geq \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2} \cdot (1+\tilde{\lambda}/3)^2(1+\tilde{\lambda})^2$ and $\theta_2 = \frac{1-p_e\ell(1+\lambda')}{p_e\ell(1+\lambda')} \geq \frac{1-\frac{1+\lambda'}{1+\tilde{\lambda}}}{p_e\ell(1+\lambda')} = \frac{\tilde{\lambda}-\lambda'}{p_e\ell(1+\lambda')(1+\tilde{\lambda})} = \frac{2\tilde{\lambda}}{3p_e\ell(1+\tilde{\lambda}/3)(1+\tilde{\lambda})}$, so that $\frac{Np_e\theta_2^2}{4} \geq \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2} \cdot (1+\tilde{\lambda}/3)^2(1+\tilde{\lambda})^2 \cdot \frac{p_e}{4} \cdot \left(\frac{2\tilde{\lambda}}{3p_e\ell(1+\tilde{\lambda}/3)(1+\tilde{\lambda})}\right)^2 \geq \ln(2/\eta)$.

Part 2: We actually prove a stronger claim than what is given in the statement of the lemma, by assuming a weaker condition. The stronger claim was not given as part of the lemma

so that the two parts of the lemma are symmetric and easier to understand. By definition, $\mathrm{AllSafe}_{\tilde{\lambda}/9,\ell} = 1$ means that the sequence of queries and answers is $\tilde{\lambda}/9$-safe for all entries and all ℓ-significant intervals. In particular, this holds for the ℓ intervals in the partition $\mathcal{I}$. Our stronger claim, which requires only this weaker assumption, is stated below and proved in Section 2.9.2.

Claim 2.7.3 *Whenever* $\mathrm{Safe}_{\tilde{\lambda}/9,i,I} = 1$ *for every* $i \in [n]$ *and each of the* ℓ *intervals* I *in the partition* $\mathcal{I}$*, the following statements hold: (i)* $\mathrm{AllSafe}_{(\frac{\lambda}{9}+\frac{16}{9(c-1)}),\beta} = 1$*. (ii) Algorithm* SAFE *returns* true *with probability at least* $1 - 2n\ell\eta$*.*

□

The constants used above have not been optimized. By picking c to be sufficiently large, we can choose $\tilde{\lambda}$ to be arbitrarily close to λ. Further, by choosing a large N so that $\theta_1, \theta_2, \theta_3$ and θ_4 could be sufficiently small, we can choose any value $\lambda' < \tilde{\lambda}$ and thereby prove the second part of the lemma with any value $\lambda'' < \tilde{\lambda}$ (instead of $\tilde{\lambda}/9$). Furthermore, taking $\eta < 1/6n\ell$, and using standard methods (repetition and majority vote), one can lower the error guarantees of Lemma 2.7.2 to be exponentially small. Neglecting this small error, we assume that Algorithm SAFE always returns *false* when $\mathrm{AllSafe}_{\lambda,\beta} = 0$ and always returns *true* when $\mathrm{AllSafe}_{\lambda'',\lceil c\beta \rceil} = 1$, for some $\lambda'' < \lambda$. The choice of c provides a trade-off in the utility guarantee of Algorithm SAFE (*i.e.*, part 2 of Lemma 2.7.2): for a large c, the privacy parameter λ'' can be made very close to λ, but the significance parameter $\lceil c\beta \rceil$ deviates more from β. From now on we also assume that Algorithm SAFE has an additional parameter λ''.

2.7.3 Constructing the Simulatable Auditor

Without loss of generality, we assume that the input $q_1, \ldots, q_{t-1}, a_1, \ldots, a_{t-1}$ contains only queries allowed by the auditor. As the auditor is simulatable, denials do not leak any information (and hence do not change the conditional probability on data sets) beyond what the previously allowed queries already leak. Each data set sampled follows a distribution which is within a total variation distance, ϵ from the desired conditional distribution, $\mathcal{D}_{Q,t-1}$. For a data set X and query Q, let $\mathrm{sum}_X(Q) = \sum_{i \in Q} X(i)$.

Algorithm 4 sum SIMULATABLE AUDITOR

1: Input: (allowed) queries and answers q_j and a_j for $j = 1, \ldots, t-1$, a new query q_t, the apriori distribution $\mathcal{D}$, and parameters $\lambda, \lambda'', \eta, c, \beta, n, \delta, T$.
2: Let $\epsilon = \frac{\delta}{10T}$
3: **for** $O(\frac{T}{\delta} \log \frac{T}{\delta})$ times **do**
4: Sample a data set X' according to $\mathcal{D}_{Q,t-1}$, using Algorithm $Sample(\mathcal{D}_{Q,t-1}, \epsilon)$
5: Let $a'_t = \text{sum}_{X'}(Q_t)$
6: Evaluate Algorithm SAFE on input $q_1, \ldots, q_t, a_1, \ldots, a_{t-1}, a'_t, \mathcal{D}$ and parameters $\lambda, \lambda'', \eta, c, \beta, n$
7: **end for**
8: **if** the fraction of sampled data sets for which Algorithm SAFE returned *false* is more than $\frac{\delta}{2T}$ **then**
9: Output "Deny" and **return**
10: **else**
11: Output "Answer" and **return**
12: **end if**

Theorem 2.7.4 *Algorithm 4 is a $(\lambda, \delta, \beta, T)$-private simulatable auditor.*

Proof: By Definition 2.4, the attacker wins the game in round t if he poses a query q_t for which $\text{AllSafe}_{\lambda,\beta}(q_1, \ldots, q_t, a_1, \ldots, a_t) = 0$ and the auditor does not deny q_t.

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

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

where X'' is a data set drawn truly according to $\mathcal{D}_{Q,t-1}$ (*i.e.,* $\mathcal{D}$ conditioned on $q_1, \ldots, q_{t-1}$, $a_1, \ldots, a_{t-1}$). Let $\tilde{p}_{t,j}$ denote the corresponding probability under the distribution sampled by Algorithm $Sample$ in the j^{th} iteration. From Theorem 2.7.1, $\tilde{p}_{t,j} \geq p_t - \epsilon$. Let $\tilde{p}_t = \min_j \tilde{p}_{t,j}$ so that $\tilde{p}_t \geq p_t - \epsilon$. Note that, since $a'_t = \text{sum}_{X'}(Q_t)$ is precisely what the algorithm computes, the algorithm essentially estimates $\tilde{p}_t$ via multiple draws of random data sets X' from $\mathcal{D}_{Q,t-1}$.

Our auditor, however, may deny answering q_t. First consider the case when $p_t > \frac{\delta}{T}$ so that $\tilde{p}_t > \frac{9\delta}{10T}$. Then, by the Chernoff bound, the fraction computed in step 8 is expected to be higher than $\frac{\delta}{2T}$, with probability at least $1 - \frac{\delta}{T}$. Hence, if $p_t > \frac{\delta}{T}$, the attacker wins with probability at most $\frac{\delta}{T}$. When $p_t \leq \delta$, the attacker wins only if the query is allowed, and even then only with probability p_t. We get that in both cases the attacker wins with probability at most $\frac{\delta}{T}$. By the union bound, the probability that the attacker wins any one of the T rounds is at most δ, as desired. □

2.7.4 Running Time

Denoting the running time of Algorithm $Sample$ by TIME($Sample, \epsilon$), the running time of Algorithm SAFE is $O(nc\beta N \cdot \text{TIME}(Sample, \epsilon))$ and hence the running time of Algorithm 4 is $O(nc\beta N \frac{T}{\delta} \log \frac{T}{\delta} \cdot \text{TIME}(Sample, \epsilon))$.

We observe that the above algorithm can be modified to handle the case when the number of rounds T is not known apriori or could be potentially unbounded. Suppose the number of rounds is estimated to be within a constant factor of T_0. We allot an error budget of $\delta/2$ for the first T_0 queries, $\delta/4$ for the next T_0 queries, and so on. In other words, we set $\delta/2$ as the error parameter for the first T_0 rounds, $\delta/4$ as the error parameter for the next T_0 rounds, and so on. Then, we get that the attacker wins the first T_0 rounds with probability at most $\frac{\delta}{2}$, the next T_0 rounds with probability at most $\frac{\delta}{4}$, and so on. By the union bound, the probability that the attacker wins any one of the T rounds is at most $\frac{\delta}{2} + \frac{\delta}{4} + \ldots < \delta$.

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

2.8 Summary and Future Work

We uncovered the fundamental issue that query denials leak information. While existing online auditing algorithms do not explicitly account for query denials, we believe that future

research must account for such leakage if privacy is to be ensured. We suggest one natural way to get around the leakage that is inspired by the simulation paradigm in cryptography – where the decision to deny can be equivalently decided by either the attacker or the auditor.

Next we introduced a new definition of privacy. While we believe that this definition overcomes some of the limitations discussed, there is certainly room for future work. The current definition does not ensure that the privacy of a group of individuals or any function of a group of individuals is kept private.

Our `sum` simulatable auditing algorithm demonstrates that a polynomial-time solution exists. But the sampling algorithms that we invoke are still not practical – although they have been steadily improving over the years. Simulatable `sum` queries over Boolean data is an interesting avenue for further work, as is the study of other classes of queries such as the k^{th} ranked element, variance, clusters and combinations of these.

Another direction for future work is to fully understand the notion of *utility* for auditing algorithms. A particular dimension of utility has been studied in [NMK$^+$06], based on the intuition that an auditor that denies more often has lesser utility. A related question is to analyze the *price of simulatability* — how many queries get denied when they could have been safely answered if we had looked at the true answers while making the decision. Collusion and denial of service are closely related issues that need to be investigated. As discussed in Section 1.1.1, for all interactive methods, there is an implicit assumption that all users can collude with each other and hence queries from all users are treated as coming from a single user. As a result, each user has reduced utility. Moreover, a malicious user may pose queries in such a way that innocuous queries in the future get denied. One solution could be to include certain important (aggregate) queries to the pool of answered queries so that these queries will always be answered in the future. Yet another issue is the privacy of the queries themselves – by treating all users as one entity, we assume that any user has access to the queries already posed by other users. While this access is needed to run the simulatable algorithm, providing it would violate the privacy of the queries.

2.9 Miscellaneous Technical Details

2.9.1 Proof of Lemma 2.6.1

Recall that λ and β were given and the trade-off parameter, c was chosen to be any integer greater than $1 + 2/\lambda$. The number of intervals in the partition, ℓ was set to $\lceil c\beta \rceil$. Further, the privacy parameter, $\tilde{\lambda}$ is chosen to satisfy $(\frac{c+1}{c-1})(1+\tilde{\lambda}) = (1+\lambda)$ and is positive (due to the choice of c).

Let J be any β-significant interval. Denote $\Pr_{\mathcal{D}}(x_i \in J)$ by p_J and let $d = \lfloor \ell p_J \rfloor$. Note that $d \geq c$ since $d = \lfloor \ell p_J \rfloor \geq \lfloor c\beta p_J \rfloor \geq \lfloor c\beta \cdot \frac{1}{\beta} \rfloor = c$. Hence, $\frac{d+1}{d-1} \leq \frac{c+1}{c-1}$. To prove the lemma, we need to show that the sequence of queries and answers is λ-safe for J, for all i. Instead, we will prove that the sequence is $((\frac{d+1}{d-1})(1+\tilde{\lambda})-1)$-safe for J. This is a stronger claim since $(\frac{d+1}{d-1})(1+\tilde{\lambda}) \leq (\frac{c+1}{c-1})(1+\tilde{\lambda}) = (1+\lambda)$.

Claim 2.9.1 *Let J be any β-significant interval and $d = \lfloor \ell p_J \rfloor$. Then,*

$$\left(\frac{1}{1+\tilde{\lambda}}\right)\left(\frac{d-1}{d+1}\right) \leq \frac{\Pr_{\mathcal{D}}(x_i \in J | \wedge_{j=1}^{t} (f_j(Q_j) = a_j))}{\Pr_{\mathcal{D}}(x_i \in J)} \leq (1+\tilde{\lambda})\left(\frac{d+1}{d-1}\right).$$

Proof: Since $\frac{d}{\ell} \leq \Pr_{\mathcal{D}}(x_i \in J) < \frac{d+1}{\ell}$ and since x_i is equally likely to occur in any of the ℓ intervals in $\mathcal{I}$, one of the following is true:

- *Case 1*: J is contained in the union of $d+1$ consecutive intervals in $\mathcal{I}$, say, $I_1, I_2, \ldots, I_{d+1}$, of which J contains the intervals, $I_2, I_3, \ldots, I_d$. Denote $\Pr_{\mathcal{D}}(x_i \in I_k)$ by p_k and $\Pr_{\mathcal{D}}(x_i \in I_k | \wedge_{j=1}^{t} f_j(Q_j) = a_j)$ by q_k. Note that $p_k = \frac{1}{\ell} \forall k$.

$$\begin{aligned}
\frac{d-1}{\ell(1+\tilde{\lambda})} = \frac{\sum_{k=2}^{d} p_k}{1+\tilde{\lambda}} &\leq \sum_{k=2}^{d} q_k \\
&\leq \Pr_{\mathcal{D}}(x_i \in J | \wedge_{j=1}^{t} f_j(Q_j) = a_j) \\
&\leq \sum_{k=1}^{d+1} q_k \leq (1+\tilde{\lambda}) \sum_{k=1}^{d+1} p_k = (1+\tilde{\lambda})\frac{d+1}{\ell}
\end{aligned}$$

$$\frac{d+1}{\ell} = \sum_{k=1}^{d+1} p_k \geq \Pr_{\mathcal{D}}(x_i \in J) \geq \sum_{k=2}^{d} p_k = \frac{d-1}{\ell}$$

Taking the ratio gives the result.

- *Case 2*: J is contained in the union of $d+2$ consecutive intervals in $\mathcal{I}$, say, $I_1, I_2, \ldots, I_{d+2}$, of which J contains the intervals, $I_2, I_3, \ldots, I_{d+1}$. By a similar argument as above, we get:

$$\left(\frac{1}{1+\tilde{\lambda}}\right)\left(\frac{d}{d+2}\right) \leq \frac{\Pr_{\mathcal{D}}(x_i \in J | \wedge_{j=1}^{t} f_j(Q_j) = a_j)}{\Pr_{\mathcal{D}}(x_i \in J)} \leq (1+\tilde{\lambda})\left(\frac{d+2}{d}\right)$$

The result follows from the fact that $\frac{d+2}{d} < \frac{d+1}{d-1}$.

□

2.9.2 Proof of Claim 2.7.3

The part (i) of the claim follows directly from the proof of Lemma 2.6.1, by replacing $\tilde{\lambda}$ with $\tilde{\lambda}/9$. This is because the sequence of queries and answers is $((\frac{d+1}{d-1})(1+\tilde{\lambda}/9)-1)$-safe for the interval J (as defined in that proof). In order to see why $\mathrm{AllSafe}_{(\frac{\lambda}{9}+\frac{16}{9(c-1)}),\beta} = 1$, note that:

$$\left(\frac{d+1}{d-1}\right)(1+\tilde{\lambda}/9) \leq \left(\frac{c+1}{c-1}\right)(1+\tilde{\lambda}/9) = 1 + \left(\frac{\lambda}{9} + \frac{16}{9(c-1)}\right)$$

We next prove part (ii) of the claim. By assumption, for any i and $I \in \mathcal{I}$,

$$\frac{1}{1+\tilde{\lambda}/9} \leq p_e \ell \leq 1 + \tilde{\lambda}/9$$

We will show that for each i and $I \in \mathcal{I}$, Algorithm SAFE returns *false* with probability at most 2η. Using the union bound on all $i \in [n]$ and $I \in \mathcal{I}$ yields the proof.

Let $\theta_3 = \frac{(1+\lambda')-p_e\ell}{p_e\ell} > 0$ and $\theta_4 = \frac{p_e\ell(1+\lambda')-1}{p_e\ell(1+\lambda')} > 0$.

$$\begin{aligned}\Pr_{real}\left(\frac{\ell\cdot N_e}{N}>1+\lambda'\right) &\le \Pr_{ideal}\left(\frac{\ell\cdot N_e}{N}>1+\lambda'\right)+\eta/2\\ &= \Pr_{ideal}(N_e>Np_e(1+\theta_3))+\eta/2\\ &\le e^{-\frac{Np_e\theta_3^2}{4}}+\eta/2\\ &\le \eta\end{aligned}$$

The last step is obtained using $N \ge \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}\cdot(1+\tilde{\lambda}/3)^2(3+\tilde{\lambda}/3)^2 \ge \frac{81\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}$ and $\theta_3 = \frac{(1+\lambda')-p_e\ell}{p_e\ell} \ge \frac{(1+\lambda')-(1+\tilde{\lambda}/9)}{p_e\ell} = \frac{2\tilde{\lambda}}{9p_e\ell}$, so that $\frac{Np_e\theta_3^2}{4} \ge \frac{81\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}\cdot\frac{p_e}{4}\cdot\left(\frac{2\tilde{\lambda}}{9p_e\ell}\right)^2 \ge \ln(2/\eta)$.

Similarly,

$$\begin{aligned}\Pr_{real}\left(\frac{\ell\cdot N_e}{N}<\frac{1}{1+\lambda'}\right) &\le \Pr_{ideal}\left(\frac{\ell\cdot N_e}{N}<\frac{1}{1+\lambda'}\right)+\eta/2\\ &= \Pr_{ideal}(N_e<Np_e(1-\theta_4))+\eta/2\\ &\le e^{-\frac{Np_e\theta_4^2}{4}}+\eta/2\\ &\le \eta\end{aligned}$$

The last step is obtained using $N \ge \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}\cdot(1+\tilde{\lambda}/3)^2(3+\tilde{\lambda}/3)^2$ and $\theta_4 = \frac{p_e\ell(1+\lambda')-1}{p_e\ell(1+\lambda')} \ge \frac{\frac{1+\lambda'}{1+\tilde{\lambda}/9}-1}{p_e\ell(1+\lambda')} = \frac{\lambda'-\tilde{\lambda}/9}{p_e\ell(1+\lambda')(1+\tilde{\lambda}/9)} = \frac{2\tilde{\lambda}}{3p_e\ell(1+\tilde{\lambda}/3)(3+\tilde{\lambda}/3)}$, so that $\frac{Np_e\theta_4^2}{4} \ge \frac{9\ell^2\ln(2/\eta)}{\tilde{\lambda}^2}\cdot(1+\tilde{\lambda}/3)^2(3+\tilde{\lambda}/3)^2\cdot\frac{p_e}{4}\cdot\left(\frac{2\tilde{\lambda}}{3p_e\ell(1+\tilde{\lambda}/3)(3+\tilde{\lambda}/3)}\right)^2 \ge \ln(2/\eta)$.

Chapter 3

Our Data, Ourselves: Privacy via Distributed Noise Generation

In this chapter, we consider another method for protecting privacy in the interactive framework, namely *output perturbation* [DN03, DN04, BDMN05, DMNS06]. In output perturbation, the database administrator computes exact answer to the query and then outputs a perturbed answer (say, by adding noise) as the response to the query. We focus on the amount of trust required on the database administrator and ask the following question: Is there a way to implement an output perturbation scheme without requiring a trusted collector of data?

We can classify the privacy techniques based on the amount of trust required on the database administrator. The positive results in the privacy literature fall into three broad categories: non-interactive with trusted server, non-interactive with untrusted server – specifically, via *randomized response*, in which a data holder alters her data with some probability before sending it to the server – and interactive with trusted server. In particular, previous privacy methods for the interactive framework assume that the database administrator is trusted by the individuals whose private information is contained in the database. Inspired by the desire to enable individuals to retain control over their information (as we contend in [ABG+04]), we provide a *distributed* implementation of the output perturbation schemes described in [DN04, BDMN05, DMNS06], thereby removing the assumption of a trusted

collector of data. Such an approach is desirable even from the perspective of an organization such as census bureau: the organization does not have to protect against insider attacks or worry about the high liability costs associated with a privacy breach.

Our implementation replaces the trusted server with the assumption that strictly fewer than one third of the participants are faulty (we handle Byzantine faults). In the above output perturbation schemes, privacy is obtained by perturbing the true answer to a database query by the addition of a small amount of Gaussian or exponentially distributed random noise. For many cases the results obtained are of provably better quality (accuracy and conciseness, *i.e.*, the number of samples needed for correct statistics to be computed) than is possible for randomized response or other non-interactive solutions [DMNS06]. Our main technical contribution is in the cooperative generation of shares of noise sampled from in one case the Binomial distribution (as an approximation for the Gaussian) and in the second case the Poisson distribution (as an approximation for the exponential).

Consider a database that is a collection of rows; for example, a row might be a hospital record for an individual. A query is a function f mapping rows to the interval $[0, 1]$. The *true answer* to the query is the value obtained by applying f to each row and summing the results. By responding with an appropriately perturbed version of the true answer, privacy can be guaranteed. The computational power of this provably private "noisy sums" primitive is demonstrated in [BDMN05], where it was shown how to carry out accurate and privacy-preserving variants of many standard data mining algorithms, such as k-means clustering, principal component analysis, singular value decomposition, the perceptron algorithm, and anything learnable in the statistical queries (STAT) learning model[1].

Although the powerful techniques of secure function evaluation [Yao82, GMW87] may be used to emulate any privacy mechanism, generic computations can be expensive. The current work is inspired by the combination of the simplicity of securely computing sums and the power of the noisy sums. We provide *efficient* methods allowing the parties holding their own data to act *autonomously* and without a central trusted center, while simultaneously preventing malicious parties from interfering with the utility of the data.

The approach to decentralization is conceptually very simple. For ease of exposition we

[1]This was extended in [DMNS06] to handle functions f that operate on the database as a whole, rather than on individual rows of the database.

describe the protocol assuming that every data holder participates in every query and that the functions f are predicates. We discuss relaxations of these assumptions in Section 3.4.

Structure of ODO (Our Data, Ourselves) Protocol

1. **Share Summands:** On query f, the holder of d_i, the data in row i of the database, computes $f(d_i)$ and shares out this value using a *non-malleable verifiable secret sharing scheme* (see Section 3.1), $i = 1, \ldots, n$. The bits are represented as $0/1$ values in $\mathrm{GF}(q)$, for a large prime q. We denote this set $\{0,1\}_{\mathrm{GF}(q)}$ to make the choice of field clear.

2. **Verify Values:** Cooperatively verify that the shared values are *legitimate* (that is, in $\{0,1\}_{\mathrm{GF}(q)}$, when f is a predicate).

3. **Generate Noise Shares:** Cooperatively generate shares of appropriately distributed random noise.

4. **Sum All Shares:** Each participant adds together all the shares that it holds, obtaining a share of the noisy sum $\sum_i f(d_i) + \mathbf{noise}$. All arithmetic is in $\mathrm{GF}(q)$.

5. **Reconstruct:** Cooperatively reconstruct the noisy sum using the reconstruction technique of the verifiable secret sharing scheme.

Our main technical work is in Step 3. We consider two types of noise, *Gaussian* and *scaled symmetric exponential*. In the latter distribution the probability of being at distance $|x|$ from the mean is proportional to $\exp(-|x|/R)$, the scale R determining how "flat" the distribution will be. In our case the mean will always be 0. Naturally, we must approximate these distributions using finite-precision arithmetic. The Gaussian and exponential distributions will be approximated, respectively, by the Binomial and Poisson distributions.

The rest of this chapter is organized as follows. In Section 3.1 we review those elements from the literature necessary for our work, including definitions of randomness extractors and of privacy. In Sections 3.2 and 3.3 we discuss implementations of Step 3 for Gaussian and Exponential noise, respectively. Finally, various generalizations of our results are mentioned in Section 3.4.

3.1 Cryptographic and Other Tools

Model of Computation

We assume the standard synchronous model of computation in which n processors communicate by sending messages via point-to-point channels and up to $t \leq \lfloor \frac{n-1}{3} \rfloor$ may fail in an arbitrary, Byzantine, adaptive fashion. If the channels are secure, then the adversary may be computationally unbounded. However, if the secure channels are obtained by encryption then we assume the adversary is restricted to probabilistic polynomial time computations.

Next we give a brief description of several well-known primitive building blocks for constructing distributed protocols: Byzantine Agreement [LSP82], Distributed Coin Flipping [Rab83], Verifiable Secret Sharing (VSS) [CGMA85], Non-Malleable VSS, and Secure Function Evaluation (SFE) [Gol04].

Byzantine Agreement [LSP82] is a fundamental problem in distributed computing. In one of several equivalent statements of the problem, a single distinguished processor, called the source, is to broadcast one bit v to n processors. We allow at most t Byzantine failures, *i.e.*, processors that can fail in an unpredictable, coordinated, and malicious way. A Byzantine agreement protocol achieves three properties:

1. Eventually all honest processors irreversibly *decide* on a some value (termination);
2. All honest processors decide on the same value v^* (agreement);
3. If the source is honest, $v = v^*$ (validity).

The maximum number of faulty processors for which the problem is solvable is $t < n/3$. For round-optimal Byzantine agreement in synchronous networks see [FM97, GM98]; for a survey of earlier literature see [CD89].

Distributed coin flipping is a powerful tool often used as a subroutine in randomized Byzantine agreement protocols. The primitive is also interesting in its own right, especially in the context of distributed noise generation. Formally, a protocol for distributed coin flipping guarantees the following:

1. Eventually all honest processors decide on the same value $b \in \{0, 1\}$;

2. The value of b is unbiased, *i.e.*, $\Pr[b = 0] = \Pr[b = 1] = 1/2$, where the probability space is the coin tosses of all honest processors.

The problem of achieving distributed coin flipping with small bias was motivated by the pioneering work of Rabin [Rab83], who showed that if a *global coin* were available to the participants then Byzantine agreement could be achieved in an expected constant number of rounds, the constant depending inversely on the bias, in contrast to the $t + 1$-round lower bound for deterministic solutions [FL82][2]. Cryptographic solutions to distributed coin flipping include [FM88, CKS05].

One technique used to solve the distributed coin flipping problem is *verifiable secret sharing* (VSS)[3]. A VSS scheme allows any processor to distribute shares of a secret, which can be verified for consistency. If the shares verify, the honest processors can always reconstruct the secret regardless of the adversary's behavior. Moreover, the faulty processors by themselves cannot learn any information about the secret. Introduced in [CGMA85], VSS, in particular, verifiable secret sharing of a polynomial, has become the staple of multi-party computation protocols, since it allows sharing and reconstruction of the secret despite efforts of corrupted parties that may even include the secret's owner, and since the shares of values represented by the polynomials can be efficiently manipulated with the effect of adding and multiplying the represented values [BOGW88, CCD88]. For a formal definition and survey of the topic we refer the reader to Gennaro's PhD thesis [Gen96].

For many cryptographic applications, including ours, where we do *not* assume private point-to-point channels between each pair of parties, the secrecy guarantee of the standard VSS definition is not sufficient. Indeed, one may naïvely assume that the XOR of two random bits secretly shared and later reconstructed is an unbiased random coin if at least one of the two bits was contributed by an honest processor. This scheme is insecure if the adversary is able to *copy* the shared secret (*i.e.*, the value of the XOR of two equal bits will always be 0). The notion of a *non-malleable VSS*, which is resistant to this and similar attacks, can be defined similarly to a non-malleable commitment scheme [DDN91]. A non-malleable VSS scheme ensures that the values shared by a non-faulty processor are

[2]The problem is also studied in other models, such as the full information model with non-adaptive adversaries. See [Fei99] for a summary of results in this model.

[3]See [CKS05] for an intriguing alternative.

completely independent of the values shared by the other processors; even exact copying is prevented.

Informally, a *non-malleable* public-key cryptosystem [DDN91, CS98] is a probabilistic public-key cryptosystem [GM84] with the property that seeing an encryption $\alpha \in E(m)$ does not help an adversary to construct an *encryption* of a related message $\beta \in E(m')$, where $R(m, m')$ for a nontrivial polynomial time relation R. In the public key setting, non-malleable VSS can be constructed from VSS: all messages from i to j are prefixed with the label "E_i to E_j" (these are the public encryption keys, respectively, of i and j), and encrypted under E_j. Without public keys one can use non-malleable commitments [DDN91, Bar02, PR05] provided the parties have unique names (a realistic assumption if they have IP or e-mail addresses).

Finally, we mention that any problem in multi-party computation, including those discussed in this chapter, can be studied in full generality in the *secure function evaluation* (SFE) framework. SFE, defined in [Yao82, GMW87, BOGW88, CCD88] (see also [Gol04, Chapter 7] for comprehensive treatment) is a method of computing any function on multiple inputs in a way that leaks no information about the inputs other than the function's output. Existing methods are often based on Yao's method of garbled circuits and are prohibitively expensive for most applications.

Throughout this chapter we will use the following terminology. Values that have been shared and verified, but not yet reconstructed, are said to be *in shares*. Values that are publicly known are said to be *public*.

A *randomness extractor* [NZ96] is a method of converting a non-uniform input distribution into a near-uniform distribution on a smaller set. In general, an extractor is a randomized algorithm, which additionally requires a perfect source of randomness, called the seed. Provided that the input distribution has sufficiently high min-entropy, a good extractor takes a short seed and outputs a distribution that is statistically close to the uniform. Formally,

Definition 3.1 *Letting the min-entropy of a distribution $\mathcal{D}$ on X be denoted $H_\infty(\mathcal{D}) = -\log \max_{x \in X} \mathcal{D}(x)$, a function $F\colon X \times Y \mapsto \{0,1\}^n$ is a (δ, ϵ, n)-*extractor*, if for any*

distribution $\mathcal{D}$ on X such that $H_\infty(\mathcal{D}) > \delta$,

$$|\{F(x,y)\colon x \in_{\mathcal{D}} X, y \in_U Y\} - U_n| < \epsilon,$$

where $|\cdot|$ is the statistical distance between two distributions, U_n is the uniform distribution on $\{0,1\}^n$, and $x \in_{\mathcal{D}} X$ stands for choosing $x \in X$ according to $\mathcal{D}$.

Optimal extractors can extract $n = \delta - 2\log(1/\epsilon) + O(1)$ nearly-random bits with the seed length $O(\log|X|)$ (see [Sha02] for many constructions matching the bound).

While in general the presence of a truly random seed cannot be avoided, there exist *deterministic* extractors (*i.e.*, without Y) for sources with a special structure [CGH$^+$85, CW89, TV00, KZ03, GRS04] where the randomness is concentrated on k bits and the rest are fixed. Namely,

Definition 3.2 *A distribution $\mathcal{D}$ over $\{0,1\}^N$ is an (N,k) oblivious bit-fixing source if there exists $S = \{i_1, \ldots, i_k\} \subset [N]$, such that $X_{i_1}, \ldots, X_{i_k}$ are uniformly distributed in $\{0,1\}^k$, and the bits outside S are constant.*

For any (N,k) bit-fixing source and any constant $0 < \gamma < 1/2$ Gabizon *et al.* [GRS04] give an explicit deterministic (k,ϵ)-extractor that extracts $m = k - N^{1/2+\gamma}$ bits of entropy with $\epsilon = 2^{-\Omega(n^\gamma)}$ provided that $k \gg \sqrt{N}$. In our case $N = 2n$ (n is the number of participants), and strictly more than $2/3$ of the input bits will be good. Thus, $k > 2N/3$, and so we extract more than $N/2 = n$ high quality bits by taking $\gamma < 1/2$.

A *privacy mechanism* is an interface between a user and data. It can be interactive or non-interactive.

Assume the database consists of a number n of rows, $d_1, \ldots, d_n$. In its simplest form, a query is a predicate $f : Rows \to \{0,1\}$. In this case, the true answer is simply $\sum_i f(d_i)$. Slightly more generally, f may map $[n] \times Rows \to [0,1]$, and the true answer is $\sum_i f(i, d_i)$. Note that we are completely agnostic about the domain $Rows$; rows can be Boolean, integers, reals, tuples thereof, or even strings or pictures.

A mechanism gives *ϵ-indistinguishability* [DMNS06] if for any two data sets that differ on only one row, the respective output random variables (query responses) τ and τ' satisfy

for all sets S of responses:

$$\Pr[\tau \in S] \quad \leq \quad \exp(\epsilon) \times \Pr[\tau' \in S] \; . \tag{3.1}$$

This definition ensures that seeing τ instead of τ' can only increase the probability of any event by at most a small factor. As a consequence, there is little incentive for any one participant to conceal or misrepresent her value, as so doing could not substantially change the probability of any event.

Similarly, we say a mechanism gives δ-*approximate* ϵ-*indistinguishability* if for outputs τ and τ' based, respectively, on data sets differing in at most one row,

$$\Pr[\tau \in S] \quad \leq \quad \exp(\epsilon) \times \Pr[\tau' \in S] + \delta \; .$$

The presence of a non-zero δ permits us to relax the strict relative shift in the case of events that are not especially likely. We note that it is inappropriate to add non-zero δ to the statement of ϵ-indistinguishability in [DMNS06], where the sets S are constrained to be singleton sets.

Historically, the first strong positive results for output perturbation added noise drawn from a Gaussian distribution, with density function $\Pr[x] \propto \exp(-x^2/2R)$. A slightly different definition of privacy was used in [DN04, BDMN05]. In order to recast those results in terms of indistinguishability, we show in Section 3.1.1 that the addition of Gaussian noise gives δ-approximate ϵ-indistinguishability for the noisy sums primitive when $\epsilon > [\log(1/\delta)/R]^{1/2}$. In a similar vein, Binomial noise, where n tosses of an unbiased ± 1 coin are tallied and divided by 2, also gives δ-approximate ϵ-indistinguishability so long as the number of tosses n is at least $64 \log(2/\delta)/\epsilon^2$.

Adding, instead, exponential noise results in a mechanism that can ensure ϵ-indistinguishability (that is, $\delta = 0$) [BDMN05, DMNS06]. If the noise is distributed as $\Pr[x] \propto \exp(-|x|/R)$, then the mechanism gives $1/R$-indistinguishability (cf. $\epsilon > [\log(1/\delta)/R]^{1/2}$ for Gaussian noise). Note that although the Gaussian noise is more tightly concentrated around zero, giving somewhat better accuracy for any given choice of ϵ, the exponential noise allows $\delta = 0$, giving a more robust solution.

3.1.1 Math for Gaussians and Binomials

We extend the results in [DMNS06] by determining the values of ϵ and δ for the Gaussian and Binomial distributions for which the noisy sums primitive yields δ-approximate ϵ-indistinguishability. Consider an output τ on a database D and query f. Let $\tau = \sum_i f(i, d_i) + \mathbf{noise}$, so replacing D with D' differing only in one row changes the summation by at most 1. Bounding the ratio of probabilities that τ occurs with inputs D and D' amounts to bounding the ratio of probabilities that $\mathbf{noise} = x$ and $\mathbf{noise} = x + 1$, for the different possible ranges of values for x. Thus, we first determine the largest value of x such that a relative bound of $\exp(\epsilon)$ holds, and then integrate the probability mass outside of this interval.

Recall the Gaussian density function: $p(x) \propto \exp(-x^2/2R)$. The ratio of densities at two adjacent integral points is

$$\frac{\exp(-x^2/2R)}{\exp(-(x+1)^2)/2R} \;=\; \exp(x/R + 1/2R).$$

This value remains at most $\exp(\epsilon)$ until $x = \epsilon R - 1/2$. Provided that $R \geq 2\log(2/\delta)/\epsilon^2$ and that $\epsilon \leq 1$, the integrated probability beyond this point will be at most

$$\Pr[x > \epsilon R - 1/2] \;\leq\; \frac{\exp(-(\epsilon R)^2/2R)}{(\epsilon R)\sqrt{\pi}} \;\leq\; \delta\ .$$

As a consequence, we get δ-approximate ϵ-indistinguishability when R is at least $2\log(2/\delta)/\epsilon^2$.

For the Binomial noise with bias $1/2$, whose density at $n/2 + x$ is

$$\Pr[n/2 + x] \;=\; \binom{n}{n/2+x} 1/2^n\ ,$$

we see that the relative probabilities are

$$\frac{\Pr[n/2 + x]}{\Pr[n/2 + x + 1]} \;=\; \frac{n/2 + x + 1}{n/2 - x}\ .$$

So long as x is no more than $\epsilon n/8$, this should be no more than $(1+\epsilon) < \exp(\epsilon)$. Of course,

a Chernoff bound tells us that for such x the probability that a sample exceeds it is

$$\begin{aligned}\Pr[y > n/2 + \epsilon n/8] &= \Pr[y > (1 + \epsilon/4)n/2] \\ &\leq \exp(-(\epsilon^2 n/64)).\end{aligned}$$

We get δ-approximate ϵ-indistinguishability so long as n is chosen to be at least $64\log(2/\delta)/\epsilon^2$. This exceeds the estimate of the Gaussian due to approximation error, and general slop in the analysis, though it is clear that the form of the bound is the same.

3.1.2 Adaptive Query Sequences

One concern might be that after multiple queries, the values of ϵ and δ degrade in an inelegant manner. We now argue that this is not the case.

Theorem 3.1.1 *A mechanism that permits T adaptive interactions with a δ-approximate ϵ-indistinguishable mechanism ensures δT-approximate ϵT-indistinguishability.*

Proof: We start by examining the probability that the transcript, written as an ordered T-tuple, lands in a set S.

$$\Pr[x \in S] = \prod_{i \leq T} \Pr[x_i \in S_i | x_1, \ldots, x_{i-1}].$$

As the noise is independent at each step, the conditioning on $x_1, \ldots, x_{i-1}$ only affects the predicate that is asked. As a consequence, we can substitute

$$\prod_{i \leq T} \Pr[x_i \in S_i | x_1, \ldots, x_{i-1}] \leq \prod_{i \leq T} \left(\exp(\epsilon) \times \Pr[x_i' \in S_i | x_1, \ldots, x_{i-1}] + \delta\right).$$

If we look at the additive contribution of each of the δ terms, of which there are T, we notice that they are only ever multiplied by probabilities, which are at most one. Therefore,

each contributes at most an additive δ.

$$\begin{aligned}\prod_{i\leq T}\Pr[x_i \in S_i|x_1,\ldots,x_{i-1}] &\leq \prod_{i\leq T}(\exp(\epsilon)\times\Pr[x'_i \in S_i|x_1,\ldots,x_{i-1}]) + \delta T \\ &= \exp(\epsilon T)\times\prod_{i\leq T}(\Pr[x'_i \in S_i|x_1,\ldots,x_{i-1}]) + \delta T \\ &= \exp(\epsilon T)\times\Pr[x' \in S] + \delta T\ .\end{aligned}$$

The proof is complete. □

3.2 Generating Gaussian Noise

Were we not concerned with malicious failures, a simple approach would be to have each participant i perturb $f(d_i)$ by sampling from a Gaussian with mean zero and variance $\frac{3}{2}\mathbf{var}/n$, where $\mathbf{var}$ is a lower bound on the variance needed for preserving privacy (see Section 3.1). The perturbed values would be shared out and the shares summed, yielding $\sum_i f(d_i) + \mathbf{noise}$ in shares. Since, as usual in the Byzantine literature, we assume that at least $2/3$ of the participants will survive, the total variance for the noise would be sufficient (but not excessive). However, a Byzantine processor might add an outrageous amount of noise to its share, completely destroying the integrity of the results. We now sketch the main ideas in our solution for the Byzantine case.

Recall that the goal is for the participants to obtain the noise in shares. As mentioned earlier, we will approximate the Gaussian with the Binomial distribution, so if the participants hold shares of sufficiently many unbiased coins they can sum these to obtain a share of (approximately) correctly generated noise. Coin flipping in shares (and otherwise) is well studied, and can be achieved by having each participant non-malleably verifiably share out a value in $\mathrm{GF}(2)$, and then locally summing (in $\mathrm{GF}(2)$) the shares from all n secret sharings.

This suggests a conceptually straightforward solution: Generate many coins in shares, convert the shares from $\mathrm{GF}(2)$ to shares of values in a large field $\mathrm{GF}(q)$ (or to shares of integers), and then sum the shares. In addition to the conversion costs, the coins themselves are expensive to generate, since they require $\Omega(n)$ executions of verifiable secret sharing

per coin, which translates into $\Omega(nc)$ secret sharings for c coins[4]. To our knowledge, the most efficient scheme for generating random bits is due to Damgård *et al.* [DFK+06], which requires n sharings and two multiplications per coin.

We next outline a related but less expensive solution which at no intermediate or final point uses the full power of coin-flipping. The solution is cost effective when c is sufficiently large, *i.e.*, $c \in \Omega(n)$. As a result, we will require only $\Omega(c)$ sharings of values in $\mathrm{GF}(2)$ when $c \in \Omega(n)$. Let n denote both the number of players and the desired number of coins[5].

1. Each player i shares a random bit by sharing out a value $b_i \in \{0,1\}_{\mathrm{GF}(q)}$, using a non-malleable verifiable secret sharing scheme, where q is sufficiently large, and engages in a simple protocol to prove that the shared value is indeed in the specified set. (The verification is accomplished by distributively checking that $x^2 = x$ for each value x that was shared, in parallel. This is a single secure function evaluation of a product, addition of two shares, and a reconstruction, for each of the n bits b_i.) This gives a sequence of low-quality bits in shares, as some of the shared values may have been chosen adversarially. (Of course, the faulty processors know the values of the bits they themselves have produced.)

2. Now, suppose for a moment that we have a public source of unbiased bits, c_1, c_2,..., c_n. By XORing together the corresponding b's and c's, we can transform the low quality bits b_i (in shares) into high-quality bits $b_i \oplus c_i$, in shares. (Again, the faulty processors know the values of the (now randomized) bits they themselves have produced.) The XORing is simple: if $c_i = 0$ then the shares of b_i remain unchanged. If $c_i = 1$ then each share of b_i is replaced by one minus the original share.

3. Replace each share s by $2s - 1$, all arithmetic in $\mathrm{GF}(q)$. This maps shares of 0 to shares of -1, and shares of 1 to (different) shares of 1.

[4]When a single player shares out many values (not the case for us), the techniques of Bellare, Garay, and Rabin [BGR96] can be used to reduce the cost of verifying the shared out values. The techniques in [BGR96] complement ours; see Section 3.4.

[5]If the desired number of coins is $o(n)$, we can generate $\Theta(n)$ coins and keep the unused ones in reserve for future executions of the protocol. If $m \gg n$ coins are needed, each processor can run the protocol m/n times.

4. Finally, each participant sums her shares to get a share of the Binomial noise.

We now turn to the generation of the c_i. Each participant randomly chooses and non-malleably verifiably shares out two bits, for a total of $2n$ low-quality bits in shares. This is done in $\mathrm{GF}(2)$, so there is no need to check legitimacy. Let the low-quality source be $b'_1, b'_2, \ldots, b'_{2n}$. The b'_i are then reconstructed, so that they become public. The sequence $b'_1 b'_2 \ldots b'_{2n}$ is a bit-fixing source: some of the bits are biased, but they are independent of the other bits (generated by the good participants) due to the non-malleability of the secret sharing. The main advantage of such a source is that it is possible to apply a *deterministic* extractor on those bits and have the output be very close to uniform. Since the bits $b'_1, \ldots, b'_{2n}$ are public, this extraction operation can be done by each party individually with no additional communication. In particular we may use, say, the currently best known deterministic extractor of [GRS04], which produces a number $m > n$ of nearly unbiased bits. The outputs of the extractor are our public coins $c_1, \ldots, c_m$.

The principal costs are the multiplications for verifying membership in $\{0,1\}_{\mathrm{GF}(q)}$ and the executions of verifiable secret sharing. Note that all the verifications of membership are performed simultaneously, so the messages from the different executions can be bundled together. The same is true for the verifications in the VSS. The total cost of the scheme is $\Theta(n)$ multiplications and additions in shares, which can be all done in a constant number of rounds.

3.3 Generating Exponential Noise

Recall that in the exponential distribution the probability of obtaining a value at distance $|x|$ from the mean is proportional to $\exp(-|x|/R)$, where R is a scaling factor. For the present discussion we take $R = 1/(\ln 2)$, so that $\exp(-|x|/R) = 2^{-|x|}$. We approximate the exponential distribution with the Poisson distribution. An intuitively simple approach is to generate a large number of unbiased[6] random bits in shares, and then find (in shares) the position ℓ of the first 1. The value returned by this noise generation procedure is $\pm\ell$ (we flip one additional bit to get the sign). If there is no 1, then the algorithm fails, so the

[6]For values of $R \neq 1/(\ln 2)$ we would need to use biased bits.

number of bits must be sufficiently large that this occurs with negligible probability. All the computation must be done in shares, and we cannot "quit" once a 1 has been found (this would be disclosive). This "unary" approach works well when $R = 1/(\ln 2)$ and the coins are unbiased. For much larger values of R, the case in high-privacy settings, the coins need to be heavily biased toward 0, flattening the curve. This would mean more expected flips before seeing a 1, potentially requiring an excessive number of random bits.

Instead, we take advantage of the special structure of the exponential distribution, and see that we can generate the *binary* representation of an exponential variable using a number of coins that is independent of the bias. Let us return to the question of the location ℓ of the first 1 in a sequence of randomly generated bits. We can describe ℓ one bit at a time by answering the following series of questions:

1. What is the parity of ℓ? That is, $\ell = 2^i$ for some $i \geq 0$? (We begin counting the positions at 0, so that ℓ will be the number of 0's preceding the first 1.)

2. Is ℓ in the left half or the right half of a block of 4 positions, *i.e.*, is it the case that $2^2 i \leq \ell < 2^2 i + 2$ for some $i \geq 0$?

3. Is ℓ in the left half or the right half of a block 8 positions, *i.e.*, is it the case that $2^3 i \leq \ell < 2^3 i + 2^2$ for some $i \geq 0$?

4. And so on.

We generate the distribution of ℓ "in binary" by generating the answers to the above questions. (For some fixed d we simply assume that $\ell < 2^d$, so only a finite number of questions need be answered.)

To answer the questions, we need to be able to generate biased coins. The probability that ℓ is even (recall that we begin counting positions with 0) is $(1/2)\sum_{i=0}^{\infty}(2^{-2i})$. Similarly, the probability that ℓ is odd is $(1/2)\sum_{i=0}^{\infty}(2^{-(2i+1)})$. Thus,

$$\Pr[\ell \text{ odd}] = (1/2)\Pr[\ell \text{ even}].$$

Since the two probabilities sum to 1, the probability that ℓ is even is $2/3$. Similar analyses yield the necessary biases for the remaining questions.

The heart of the technical argument is thus to compute coins of arbitrary bias in shares in a manner that consumes on average a constant number of unbiased, completely unknown, random bits held in shares. We will construct and analyze a shallow circuit for this. In addition, we will present two incomparable probabilistic constructions. In any distributed implementation these schemes would need to be implemented by general secure function evaluation techniques. The circuits, which only use Boolean and finite field arithmetic, allow efficient SFE implementation.

3.3.1 Poisson Noise: The Details

In this section we describe several circuits for generating Poisson noise. The circuits will take as input random bits (the exact number depends on the circuit in question). In the distributed setting, the input would be the result of a protocol that generates (many) unbiased bits in shares. The circuit computation would be carried out in a distributed fashion using secure function evaluation, and would result in *many* samples, in shares, of **noise** generated according to the Poisson distribution. This fits into the high-level ODO protocol in the natural way: shares of the noise are added to the shares of $\sum_i f(i, d_i)$ and the resulting noisy sum is reconstructed.

For the remainder of this section, we let n denote the number of coins to be generated. It is unrelated to the number of participants in the protocol.

Recall the discussion in the Introduction of the exponential distribution, where $\Pr[x] \propto \exp(-|x|/R)$. Recall that one interpretation is to flip a (possibly biased) coin until the first 1 is seen, and then to output the number ℓ of 0's seen before the 1 occurs. Recall also that instead of generating ℓ in unary, we will generate it in binary.

We argue that the bits in the binary representation of the random variable ℓ are independent, and moreover we can determine their biases analytically. To see the independence, consider the distribution of the ith bit of ℓ:

$$\ell_i \;=\; \begin{cases} 0 & \text{w.p. } \Pr[0 \times 2^i \le \ell < 1 \times 2^i] + \Pr[2 \times 2^i \le \ell < 3 \times 2^i] + \ldots \\ 1 & \text{w.p. } \Pr[1 \times 2^i \le \ell < 2 \times 2^i] + \Pr[3 \times 2^i \le \ell < 4 \times 2^i] + \ldots \end{cases}$$

Notice that corresponding terms in the two summations, eg $\Pr[0 \times 2^i \le \ell < 1 \times 2^i]$ and

$\Pr[1 \times 2^i \leq \ell < 2 \times 2^i]$, are directly comparable; the first is exactly $\exp(2^i/R)$ times the second. This holds for every corresponding pair in the sums, and as such the two sums share the same ratio. As the two sum must total to one, we have additionally that

$$1 - \Pr[\ell_i] \;=\; \exp(2^i/R) \times \Pr[\ell_i] \; .$$

Solving, we find that

$$\Pr[\ell_i] \;=\; 1/(1 + \exp(2^i/R)) \; .$$

Recall as well that the observed ratio applied equally well to each pair of intervals, indicating that the bias is independent of the more significant bits. The problem of producing an exponentially distributed ℓ is therefore simply a matter of flipping a biased coin for each bit of ℓ. The circuit we will construct will generate many ℓ's according to the desired distribution, at an expected low amortized cost (number of input bits) per bit position in the binary expansion of ℓ. The circuit is a collection of circuits, each for one bit position, with the associated bias hard-wired in. It suffices therefore to describe the circuitry for one of these smaller circuits (Section 3.3.3). We let p denote the hard-wired bias.

A well-known technique for flipping a single coin of arbitrary bias p is to write p in binary, examine random bits until one differs from the corresponding bit in p, and then emit the complement of the random bit. To achieve a high fidelity to the original bias p, a large number d of random bits must be available. However, independent of p, the expected number of random bits consumed is at most 2. This fact will be central to our constructions.

In the sequel we distinguish between unbiased *bits*, which are inputs to the algorithm, and the generated, biased, *coins*, which are the outputs of the algorithm.

3.3.2 Implementation Details: Finite Resources

With finite randomness we cannot perfectly emulate the bias of the coins. Moreover, the expectation of higher order bits in the binary representation of ℓ diminishes at a doubly exponential rate (because the probability that $\ell \geq 2^i$ is exponentially small in 2^i), quickly giving probabilities that simply can not be achieved with any fixed amount of randomness.

To address these concerns, we will focus on the statistical difference between our produced distribution and the intended one. The method described above for obtaining coins with arbitrary bias, truncated after d bits have been consumed, can emulate any biased coin within statistical difference at most 2^{-d}. Accordingly, we set all bits of sufficiently high order to zero, which will simplify our circuit. The remaining output bits – let us imagine there are k of them – will result in a distribution whose statistical difference is at most $k2^{-d}$ from the target distribution. We note that by trimming the distribution to values at most 2^d in magnitude, we are introducing an additional error, but one whose statistical difference is quite small. There is an $\exp(-2^d/R)$ probability mass outside the $[-2^d, 2^d]$ interval that is removed and redistributed inside the interval. This results in an additional $2\exp(-2^d/R)$ statistical difference that should be incorporated into δ. For clarity, we absorb this term into the value k.

Using our set of coins with statistical difference at most $k2^{-d}$ from the target distribution, we arrive at a result akin to (3.1), though with an important difference. For response variables τ and τ' as before (based on databases differing it at most one row),

$$\forall S \subseteq U: \quad \Pr[\tau \in S] \quad \leq \quad \Pr[\tau' \in S] \times \exp(1/R) + k2^{-d} \; .$$

As before, the probability of any event increases by at most a factor of $\exp(1/R)$, but now with an additional additive $k2^{-d}$ term. This term is controlled by the parameter d, and can easily be made sufficiently small to allay most concerns.

We might like to remove the additive $k2^{-d}$ term, which changes the nature of the privacy guarantee. While this seems complicated at first, notice that it is *possible* to decrease the relative probability associated with each output coin arbitrarily, by adding more bits (that is, increasing d). What additional bits can not fix is our assignment of zero probability to noise values outside the permitted range (*i.e.*, involving bits that we do not have circuitry for).

One pleasant resolution to this problem, due to Adam Smith, is to constrain the output range of the sum of noise plus signal. If the answer plus noise is constrained to be a k-bit number, and conditioned on it lying in that range the distribution looks exponential, the same privacy guarantees apply. Guaranteeing that the output will have only k bits can

be done by computing the sum of noise and signal using $k+1$ bits, and then if there is overflow, outputting the noise-free answer. This increases the probability that $\mathbf{noise} = 0$ by a relatively trivial amount, and ensures that the output space is exactly that of k-bit numbers.

3.3.3 A Circuit for Flipping Many Biased Coins

We are now ready to construct a circuit for flipping a large number of independent coins with common bias. By producing many ($\Omega(n)$) coins at once, we could hope to leverage the law of large numbers and consume, with near certainty, a number of input bits that is little more than $2n$ and depends very weakly on d. For example, we could produce the coins sequentially, consuming what randomness we need and passing unused random bits on to the next coin. The circuit we now describe emulates this process, but does so in a substantially more parallel manner.

The circuit we construct takes 2^i unbiased input bits and produces 2^i output coins, as well as a number indicating how many of the coins are actually the result of the appropriate biased flips. That is, it is unlikely that we will be able to produce fully 2^i coins, and we should indicate how many of the coins are in fact valid. The construction is hierarchical, in that the circuit that takes 2^i inputs will be based on two level $i-1$ circuits, attached to the first and second halves of its inputs.

To facilitate the hierarchical construction, we augment the outputs of each circuit with the number of bits at the end of the 2^i that were consumed by the coin production process, but did not diverge from the binary representation of p. Any process that wishes to pick up where this circuit has left off should start under the assumption that the first coin is in fact this many bits into its production. For example, if this number is r then the process should begin by comparing the next random bit to the $(r+1)$st bit in the expansion of p. Bearing this in mind, we "bundle" d copies of this circuit together, each with a different assumption about the initial progress of the production of their first coin.

For each value $1 \leq j \leq d$ we need to produce a vector of 2^i coins c_j, a number of coins n_j, and d_j, a measure of progress towards the last coin. We imagine that we have access to two circuits of one level lower, responsible for the left and right half of our 2^i input bits,

and whose corresponding outputs are superscripted by L and R. Intuitively, for each value of j we ask the left circuit for d_j^L, which we use to select from the right circuit. Using index j for the left circuit and d_j^L for the right circuit, we combine the output coins using a shift of n_j^L to align them, and add the output counts n_j^L and $n_{d_j^L}^R$. We simply pass $d_{d_j^L}^R$ out as the appropriate value for d_j.

$$\begin{aligned} c_j &= c_j^L \mid (c_{d_j^L}^R >> n_j^L) \\ n_j &= n_j^L + n_{d_j^L}^R \\ d_j &= d_{d_j^L}^R \end{aligned}$$

The operation of subscripting is carried out using a multiplexer, and shifts, bitwise ors, and addition are similarly easily carried out in logarithmic depth.

The depth of each block is bounded by $\Theta(\log(nd))$, with the size bounded by $\Theta(2^i d(\log(n) + d)$, as each of d outputs must multiplex d possible inputs (taking $\Theta(d)$ circuitry) and then operate on them (limited by $\Theta(\log(n)2^i)$ for the barrel shifter). All told, the entire circuit has depth $\Theta(\log(nd)^2)$, with size $\Theta(nd(\log(n) + d)\log(n))$.

3.3.4 Probabilistic Constructions with Better Bounds

We describe two probabilistic constructions of circuits that take as input unbiased bits and produce as output coins of arbitrary, *not necessarily identical*, bias. Our first solution is optimal in terms of depth ($\Theta(\log d)$) but expensive in the gate count. Our second solution dramatically decreases the number of gates, paying a modest price in depth ($O(\log(n+d))$) and a logarithmic increase in the number of input bits.

A module common to both constructions is the comparator – a circuit that takes two bit strings $b_1, \ldots, b_d$ and $p^{(1)} \ldots p^{(d)}$ and outputs 0 if and only if the first string precedes the second string in the lexicographic order. Equivalently, the comparator outputs $\bar{b}_i$, where i is the index of the earliest occurrence 1 in the sequence $b_1 \oplus p^{(1)}, \ldots, b_d \oplus p^{(d)}$, or 1 if the two strings are equal. Based on this observation, a circuit of depth $\Theta(\log d)$ and size $\Theta(d)$ can be designed easily. Notice that the result of comparison is independent of the values of the strings beyond the point of divergence.

Brute Force Approach

Assume that we have nd independent unbiased bits $b_i^{(j)}$, for $1 \leq i \leq n$ and $1 \leq j \leq d$. To flip n independent coins, each with its own bias p_i, whose binary representation is $0.p_i^{(1)} \ldots p_i^{(d)}$, we run n comparators in parallel on inputs $(b_1^{(1)}, \ldots, b_1^{(d)}, p_1^{(1)}, \ldots, p_1^{(d)})$, $\ldots$, $(b_n^{(1)}, \ldots, b_n^{(d)}, p_n^{(1)}, \ldots, p_n^{(d)})$.

Our goal is to get by with many fewer than nd unbiased input bits of the brute force approach, since each of these requires an unbiased bit in shares. Intuitively, we may hope to get away with this because, as mentioned previously, the average number of bits consumed per output coin is 2, independent of the bias of the coin. Let c_i for $1 \leq i \leq n$ be the smallest index where $b_i^{(c_i)} \neq p_i^{(c_i)}$, and $d+1$ if the two strings are equal. The number c_i corresponds to the number of bits "consumed" during computation of the ith coin. Let $C = \sum_{i=1}^{n} c_i$. On expectation $E[C] = 2n$, and except with a negligible probability $C < 4n$.

Rather than having the set $\{b_i^{(j)}\}_{i,j}$ be given as input (too many bits), we will compute the set $\{b_i^{(j)}\}_{i,j}$ from a much smaller set of input bits. The construction will ensure that the *consumed* bits are independent except with negligible probability. Let the number of input bits be D, to be chosen later.

We will construct the circuit probabilistically. Specifically, we begin by choosing nd binary vectors $\{r_i^{(j)}\}_{i,j}$, $1 \leq i \leq n$ and $1 \leq j \leq d$, uniformly from $\{0,1\}^D$ to be hard-wired into the circuit. Let $b \in_R \{0,1\}^D$ be the uniformly chosen random input to the circuit.

The circuit computes the inner products of each of the hard-wired vectors $r_i^{(j)}$ with the input b. Let $b_i^{(j)} = \langle r_i^{(j)}, b \rangle$ denote the resulting bits. These are the $\{b_i^{(j)}\}_{i,j}$ we will plug into the brute force approach described above. Note that although much randomness was used in defining the circuit, the input to the circuit requires only D random bits.

Although the nd vectors are not linearly independent, very few of them – $O(n)$ – are actually used in the computation of our coins, since with overwhelming probability only this many of the $b_i^{(j)}$ are actually consumed. A straightforward counting argument therefore shows that the set of vectors actually used in generating consumed bits will be linearly independent, and so the coins will be mutually independent.

We claim that if $D > 4C$, then the consumed bits are going to be independent with high probability. Conditional on the sequence $c_1, \ldots, c_n$, the vectors $r_i^{(j)}$ for $1 \leq i \leq n$ and

$1 \leq j \leq c_i$ are independent with probability at least $1 - C2^{C-D} < 1 - 2^{-2C}$, where the probability space is the choice of the r's. For fixed C the number of possible $c_1, \ldots, c_n$ is at most $\binom{C}{n} < 2^C$. Hence the probability that for some $C < 4n$ and some $c_1, \ldots, c_n$, such that $c_1 + \cdots + c_n = C$ the vectors $r_i^{(j)}$ are linearly independent is at least $1 - 4n2^{-C}$. Finally, we observe that if the vectors are linearly independent, the bits $b_i^{(j)}$ are independent as random variables. The depth of this circuit is $\Theta(\log D)$, which is the time it takes to compute the inner product of two D-bit vectors. Its gate count is $\Theta(ndD)$, which is clearly suboptimal.

Using Low Weight Independent Vectors

Our second solution dramatically decreases the number of gates by reducing the weight (the number of non-zero elements) of the vectors r from the expected value $D/2$ to $s^2\lceil\log(n+1)\rceil$, where s is a small constant. To this end we adopt the construction from [DLN96] that converts an expander-like graph into a set of linearly independent vectors.

The construction below requires a field with at least nd non-zero elements. Let $\nu = \lceil\log(nd+1)\rceil$. We use $\mathrm{GF}(2^\nu)$, representing its elements as ν-bit strings.

Consider a bipartite graph G of constant degree s connecting sets $L = \{u_1, \ldots, u_n\}$, where the u's are distinct field elements, and $R = \{1, \ldots, \Delta\}$. The degree s can be as small as 3. Define matrix M of size $n \times s\Delta$ as follows: if $(u_i, \tau) \in G$, the elements $M[i][s(\tau-1), s(\tau-1)+1, \ldots, s\tau-1] = u_i, u_i^2, \ldots, u_i^s$, and $(0, \ldots, 0)$ (s zeros) otherwise. Thus, each row of the matrix has exactly s^2 non-zero elements.

For any set $S \subseteq L$, let $\Gamma(S) \subseteq R$ be the set of neighbors of S in G. The following claim is easily obtained from the proof of Lemma 5.1 in [DLN96]. It says that if for a set of vertices $T \subseteq L$ all of T's subsets are sufficiently expanding, then the rows of M corresponding to vertices in T are linearly independent.

Theorem 3.3.1 *Let $T \subseteq L$ be any set for which $\forall S \subseteq T$, $|\Gamma(S)| > (1 - \frac{1}{s+1})|S|$. Then the set of vectors $\{M[u] \colon u \in T\}$ is linearly independent.*

Consider a random bipartite graph with nd/ν elements in one class and $2C$ elements in the other. Associate the elements from the first class with bits $b_i^{(j)}$'s, grouped in ν-tuples. Define the bits as the results of the inner product of the corresponding rows of the matrix M from above with the input vector of length $2s^2C$ that consists of random elements from

$\mathrm{GF}(2^\nu)$. Observe that the random graph G satisfies the condition of Theorem 3.3.1 for all sets of size less than C with high probability if $C > (nd/\nu)^{1/(s-1)}$.

The depth of the resulting circuit is $\Theta(\log(n+d))$, the gate count is $\Theta(nds^2 \log(n+d))$, and the size of the input is $2n \log(n+d)$.

3.4 Generalizations

In this section we briefly discuss several generalizations of the basic scheme.

3.4.1 Alternatives to Full Participation

The main idea is to use a set of facilitators, possibly a very small set, but one for which we are sufficiently confident that fewer than one third of the members are faulty. Let $\mathcal{F}$ denote the set of facilitators. To respond to a query f, participant i shares $f(i, d_i)$ among the facilitators, and takes no further part in the computation.

To generate the noise, each member of $\mathcal{F}$ essentially takes on the work of $n/|\mathcal{F}|$ participants. When $|\mathcal{F}|$ is small, the batch verification technique of [BGR96] may be employed to verify the secrets shared out by each of the players (that is, one batch verification per member of $\mathcal{F}$), although this technique requires that the faulty players form a smaller fraction of the total than we have been assuming up to this point.

3.4.2 When f is Not a Predicate

Suppose we are evaluating f to k bits of precision, that is, k bits beyond the binary point. Let q be sufficiently large, say, at least $q > n2^k$. We will work in $\mathrm{GF}(q)$. Participant i will share out $2^k f(i, d_i)$, *one bit at a time*. Each of these is checked for membership in $\{0, 1\}_{\mathrm{GF}(q)}$. Then the shares of the most significant bit are multiplied by 2^{k-1}, shares of the next most significant are multiplied by 2^{k-2} and so on, and the shares of the binary representation of $f(i, d_i)$ are then summed. The noise generation procedure is amplified as well.

3.4.3 Beyond Sums

We have avoided the case in which f is an arbitrary function mapping the entire database to a (tuple of) value(s), although the theory for this case has been developed in [DMNS06]. This is because without information about the structure of f we can only rely on general techniques for secure function evaluation of f, which may be prohibitively expensive.

One case in which we can do better is in the generation of *privacy-preserving histograms*. A histogram is specified by a partition of the domain Rows; the true response to the histogram query is the exact number of elements in the database residing in each of the cells of the histogram. Histograms are *low sensitivity* queries, in that changing a single row of the database changes the counts of at most two cells in the histogram, and each of these two counts changes by at most 1. Thus, as discussed in [DMNS06], ϵ-indistinguishable histograms may be obtained by adding exponential noise with $R = 1/2\epsilon$ to each cell of the histogram. A separate execution of ODO for each cell solves the problem. The executions can be run concurrently. All participants in the histogram query must participate in each of the concurrent executions.

3.4.4 Individualized Privacy Policies

Suppose Citizen C has decided she is comfortable with a *lifetime* privacy loss of, say $\epsilon = 1$. Privacy erosion is cumulative: any time C participates in the ODO protocol she incurs a privacy loss determined by R, the parameter used in noise generation. C has two options: if R is fixed, she can limit the number of queries in which she participates, *provided the decision whether or not to participate is independent of her data*. If R is not fixed in advance, but is chosen by consensus (in the social sense), she can propose large values of R, or to use large values of R for certain types of queries. Similarly, queries could be submitted with a stated value of R, and dataholders could choose to participate only if this value of R is acceptable to them for this type of query. However, the techniques will all fail if the set of participants is more than one-third faulty; so the assumption must be that this bound will always be satisfied. This implicitly restricts the adversary.

3.5 Summary

This work ties together two areas of research: the study of privacy-preserving statistical databases and that of cryptographic protocols. It was inspired by the combination of the computational power of the noisy sums primitive in the first area and the simplicity of secure evaluation of sums in the second area. The effect is to remove the assumption of a trusted collector of data, allowing individuals control over the handling of their own information.

In the course of this work we have developed distributed algorithms for generation of Binomial and Poisson noise in shares. The former makes novel use of extractors for bit-fixing sources in order to reduce the number of secret sharings needed in generating massive numbers of coins. The latter examined for the first time distributed coin-flipping of coins with arbitrary bias.

Part II

Privacy Protection in the Non-interactive Framework

Chapter 4

Approximation Algorithms for k-Anonymity

In the second part of the thesis, we consider two methods for protecting privacy in the non-interactive framework. Our goal is to publish data for analysis from a table containing personal records, while ensuring individual privacy and maintaining data integrity to the extent possible. As discussed in Chapter 1, when the aggregate queries of interest are not known ahead of time, techniques such as query auditing, output perturbation, and secure function evaluation do not provide an adequate solution, and we need to release an anonymized view of the database that enables the computation of non-sensitive query aggregates, perhaps with some error or uncertainty. The first method, which we discuss in this chapter, is combinatorial in nature and involves suppression of certain information in such a way that we can draw inferences with 100% confidence. The second method (discussed in Chapter 5) is based on clustering. Compared to the first method, the second method allows us to release more information about a table. However, with the second method, the inferences drawn may not have full confidence.

More generally, techniques under non-interactive framework such as input perturbation, sketches, or clustering may not be suitable if one wants to draw inferences with 100% confidence. Another approach to protect privacy is to *suppress* some of the data values, while releasing the remaining data values exactly. For example, consider the following table which is part of a medical database, with the identifying attributes such as name and

social security number removed.

Age	Race	Gender	Zip Code	Diseases
47	White	Male	21004	Common Cold
35	White	Female	21004	Flu
27	Hispanic	Female	92010	Flu
27	White	Female	92010	Hypertension

By joining this table with public databases (such as a voter list), non-identifying attributes, such as Age, Race, Gender, and Zip Code in the above table, can together be used to identify individuals. In fact, Sweeney [Swe00] observed that for 87% of the population in the United States, the combination of non-key fields like Date of Birth, Gender, and Zip Code corresponded to a unique person. Such non-key fields are called *quasi-identifiers*.

In order to ensure the protection of privacy, we adopt the k-Anonymity model which was proposed by Samarati and Sweeney [Sam01, SS98, Swe02]. Suppose we have a table consisting of n tuples each having m quasi-identifying attributes (Age, Race, Gender, and Zip Code in the above table), and let $k > 1$ be an integer. The k-Anonymity framework provides for generalization of entries (generalization entails replacing an entry value with a less specific but semantically consistent value; a more formal description can be found in Section 4.1) in addition to suppression. The idea is to suppress/generalize some of the entries in the table so as to ensure that *for each tuple in the modified table, there are at least* $k - 1$ *other tuples in the modified table that are identical to it along the quasi-identifying attributes.* The objective is to minimize the extent of suppression and generalization. Note that entries in the column corresponding to the sensitive attribute ("Diseases" in the above example) are not altered. The following is an example of a k-anonymized table for $k = 2$.

Age	Race	Gender	Zip Code	Diseases
*	White	*	21004	Common Cold
*	White	*	21004	Flu
27	*	Female	92010	Flu
27	*	Female	92010	Hypertension

A k-anonymized table protects individual privacy in the sense that, even with the knowledge of an individual's quasi-identifying attributes, an adversary would not be able to track

down an individual's record further than a set of at least k records. Thus, releasing a table after k-anonymization prevents definitive *record linkages* with publicly available databases, and keeps each individual hidden in a crowd of $k-1$ other people. The privacy parameter k must be chosen according to the application in order to ensure the required level of privacy.

4.1 Model and Results

We now formally define the problem of k-Anonymity and state our results. The input is a table having n rows each with m quasi-identifying attributes. We view the table as consisting of n m-dimensional vectors: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Sigma^m$.

We first define a special case of the problem called *k-Anonymity with Suppression*, where suppression is the only permitted operation. A *k-Anonymous suppression function* t maps each $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ by replacing some components of $\mathbf{x}_i$ by $*$ (which corresponds to hiding those components of $\mathbf{x}_i$), so that every $\tilde{\mathbf{x}}_i$ is identical to at least $k-1$ other $\tilde{\mathbf{x}}_j$s. This results in a partition of the n row vectors into *clusters* of size at least k each. The cost of the suppression, $c(t)$ is the total number of hidden entries, or equivalently, the total number of $*$s in all the $\tilde{\mathbf{x}}_i$s.

k-Anonymity with Suppression: *Given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$, and an Anonymity parameter k, obtain a k-Anonymous suppression function t so that $c(t)$ is minimized.*

Next, we define the problem of *k-Anonymity with Generalization*, where in addition to suppressing entry values, we are also allowed to replace them with less specific but semantically consistent values. For example, we can make a date less specific by omitting the day and revealing just the month and year. We assume that for each attribute, a generalization hierarchy is specified as part of the input [SS98, Sam01]. For an attribute, each level of generalization corresponds to a partition of the attribute domain. A partition corresponding to any given level of the generalization hierarchy is a refinement of the partition corresponding to the next higher level. Singleton sets correspond to absence of generalization, while the partition consisting of a single set containing the whole domain corresponds to the highest level of generalization. Consider the example shown in Figure 4.1. The attribute "Quality" has a domain consisting of values $A+$, A, $A-$, $B+$, B and

$B-$ and has two levels of generalization. In the absence of generalization, the value of this attribute is reported exactly. The first level of generalization corresponds to the partition $\{\{A+, A, A-\}, \{B+, B, B-\}\}$. In order to generalize an entry with value "A" to the first level of generalization, it is replaced with the set $\{A+, A, A-\}$. The next higher level of generalization (also the highest level in this case) corresponds to replacing the entry with the set containing the whole domain, which is equivalent to suppressing the entry.

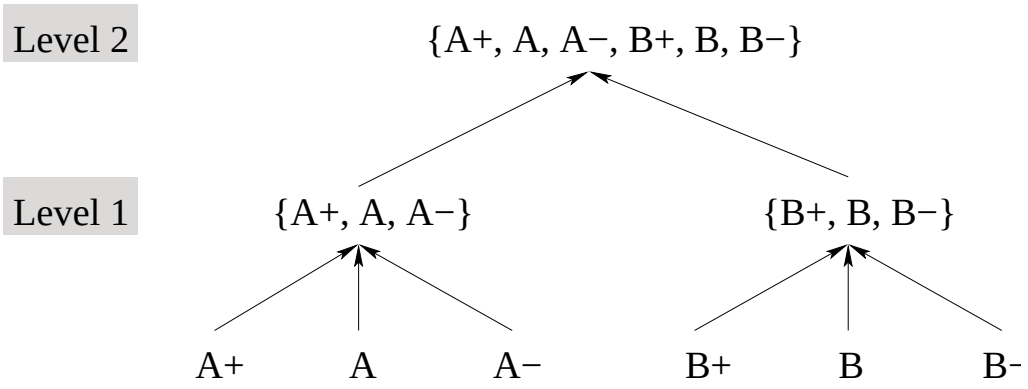


Figure 4.1: A possible generalization hierarchy for the attribute "Quality".

Let the j^{th} attribute have domain D^j and l_j levels of generalization. Let the partition corresponding to the h^{th} level of generalization be D_h^j for $1 \leq h \leq l_j$, with $D_0^j = D^j$. Let a value $y \in D^j$ when generalized to the h^{th} level be denoted by $g_h(y)$, e.g., $g_1(A) = \{A+, A, A-\}$. A *generalization function* h is a function that maps a pair (i, j), $i \leq n$, $j \leq m$ to a level of generalization $h(i, j) \leq l_j$. Semantically, $h(i, j)$ denotes the level to which j^{th} component of the i^{th} vector (or the $(i, j)^{th}$ entry in the table) is generalized. Let $h(\mathbf{x}_i)$ denote the *generalized* vector corresponding to $\mathbf{x}_i$, *i.e.*, $h(\mathbf{x}_i) = (g_{h(i,1)}(x_i[1]), g_{h(i,2)}(x_i[2]) \ldots, g_{h(i,m)}(x_i[m]))$. A generalization function is said to be k-Anonymous if for every i, $h(\mathbf{x}_i)$ is identical to $h(\mathbf{x}_j)$ for at least $k-1$ values of $j \neq i$.

Consider a k-Anonymous generalization function h. It incurs a cost of r/l_j whenever it generalizes a value for the j^{th} attribute to the r^{th} level. The total cost incurred by the generalization function h is defined as the sum of the costs incurred over all the entries of the table, *i.e.*, cost$(h) = \sum_i \sum_j h(i, j)/l_j$. Now we are ready to give a formal definition of the problem.

k-Anonymity with Generalization: *Given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$, and an Anonymity parameter k, obtain a k-Anonymous generalization function h such that cost(h) is minimized.*

Note that the problem of k-Anonymity with Suppression is a special case of the problem of k-Anonymity with Generalization, with only one level of generalization (corresponding to hiding the entry completely) for every attribute.

Clearly the decision version of both of these problems is in NP, since we can verify in polynomial time if the solution is k-Anonymous and the suppression cost less than a given value. We show that k-Anonymity with Suppression is NP-hard even when the alphabet size $|\Sigma| = 3$. Note that this automatically implies NP-hardness of k-Anonymity with Generalization. This improves upon the NP-hardness result of [MW04] which required an alphabet size of n. On the positive side, we provide an $O(k)$-approximation algorithm for k-Anonymity with Generalization for arbitrary k and arbitrary alphabet size, using a graph representation. This improves upon the previous best-known approximation guarantee of $O(k \log k)$ for k-Anonymity with Suppression [MW04]. We also show that it is not possible to achieve an approximation factor better than $\Theta(k)$ using the graph representation approach. For a binary alphabet, we provide improved approximation algorithms for $k = 2$ (an approximation factor of 1.5) and $k = 3$ (an approximation factor of 2).

The rest of this chapter is organized as follows. We establish the NP-hardness of k-Anonymity with Suppression in Section 4.2. We then present an $O(k)$-approximation algorithm for k-Anonymity with Generalization in Section 4.3. Next, in Sections 4.4 and 4.5, we provide a 1.5 approximation algorithm for the 2-Anonymity problem with binary alphabet, and a 2-approximation algorithm for 3-Anonymity with binary alphabet. Finally, we conclude with some future research directions in Section 4.6.

4.2 NP-hardness of k-Anonymity with Suppression

Theorem 4.2.1 *k-Anonymity with Suppression is NP-hard even for a ternary alphabet, i.e., $(\Sigma = \{0, 1, 2\})$.*

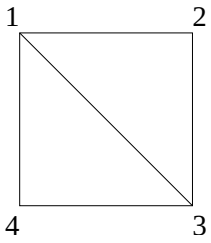
Proof: In this proof, k-Anonymity refers to the problem of k-Anonymity with Suppression. We give a reduction from the NP-hard problem of EDGE PARTITION INTO TRIANGLES [Kan94] which is defined as follows: *Given a graph $G = (V, E)$ with $|E| = 3m$ for some integer m, can the edges of G be partitioned into m edge-disjoint triangles?*

Given an instance of the above problem, $G = (V, E)$ with $3m$ edges (since the above problem is NP-hard even for simple graphs, we will assume that the graph G is simple), we create a preliminary table T with $3m$ rows — one row for each edge. For each of the n vertices of G, we create an attribute (column). The row corresponding to edge (a, b), referred to as r_{ab}, has ones in the positions corresponding to a and b and zeros everywhere else. Let a star with four vertices (having one vertex of degree 3) be referred to as a 4-star.

Equivalence to edge partition into triangles and 4-stars. We first show that the cost of the optimal 3-Anonymity solution for the table T is at most $9m$ if and only if E can be partitioned into a collection of m disjoint triangles and 4-stars. First suppose that such a partition of edges is given. Consider any triangle (with a, b, c as its vertices). By suppressing the positions a, b and c in the rows r_{ab}, r_{bc} and r_{ca}, we get a cluster containing three rows, with three $*$s in each modified row. Now consider a 4-star with vertices a, b, c, d, where d is the center vertex. By suppressing the positions a, b and c in the rows r_{ad}, r_{bd} and r_{cd}, we get a cluster containing three rows with three $*$s in each modified row. Thus we obtain a solution to 3-Anonymity of cost $9m$.

On the other hand, suppose that there is a 3-Anonymity solution of cost at most $9m$. Since G is simple, any three rows are distinct and differ in at least 3 positions. Hence there should be at least three $*$s in each modified row, so that the cost of the solution is at least $9m$. This implies that the solution cost is exactly $9m$ and each modified row has exactly three $*$s. Since any cluster of size more than three will have at least four $*$s in each modified row, it follows that each cluster has exactly three rows. There are exactly two possibilities: the corresponding edges form either a triangle or a 4-star, and each modified row in a triangle has three $*$s and zeros elsewhere while each modified row in a 4-star has three $*$s, single 1 and zeros elsewhere. Thus, the solution corresponds to a partition of the edges of the graph into triangles and 4-stars.

Equivalence to edge partition into triangles. Since we want a reduction from EDGE PARTITION INTO TRIANGLES, we create a table T' by "replicating" the columns of T so as to force the 4-stars to pay more $*$s. Let $t = \lceil \log_2(3m+1) \rceil$. In the new table T', every row has t *blocks*, each of which has n columns. Consider an arbitrary ordering of the edges in E and express the rank of an edge $e = (a, b)$, in this ordering, in binary notation as $e_1 e_2 \ldots e_t$. In the row corresponding to edge e, each block has zeros in all positions except a and b. A block can be in one of two configurations: $conf_0$ has a 1 in position a and a 2 in position b while $conf_1$ has a 2 in position a and a 1 in position b. The i^{th} block in the row corresponding to e has configuration $conf_{e_i}$. For example, consider the graph shown in Figure 4.2. Suppose the edges $(3,4), (1,4), (1,2), (1,3), (2,3)$ are ranked 1 (*i.e.*, $(001)_2$) through 5 (*i.e.*, $(101)_2$) respectively. Then, the table in Figure 4.2 represents the 3-Anonymity instance corresponding to the graph, with the i^{th} row in the table representing the vector corresponding to the edge ranked i.



(3,4)	0	0	1	2	0	0	1	2	0	0	2	1
(1,4)	1	0	0	2	2	0	0	1	1	0	0	2
(1,2)	1	2	0	0	2	1	0	0	2	1	0	0
(1,3)	2	0	1	0	1	0	2	0	1	0	2	0
(2,3)	0	2	1	0	0	1	2	0	0	2	1	0

Figure 4.2: The table shows the 3-anonymity instance corresponding to the graph on the left when the edges $(3,4), (1,4), (1,2), (1,3), (2,3)$ are ranked 1 through 5 respectively.

We will now show that the cost of the optimal 3-Anonymity solution on T' is at most $9mt$ if and only if E can be partitioned into m disjoint triangles.

Suppose that E can be partitioned into m disjoint triangles. As earlier, every triangle in such a partition corresponds to a cluster with $3t$ $*$s in each modified row. Thus we get a 3-Anonymity solution of cost $9mt$.

For the converse, suppose that we are given a 3-Anonymity solution of cost at most $9mt$. Again, any three rows differ in at least $3t$ positions so that the cost of any solution is at least $9mt$. Hence the solution cost is exactly $9mt$ and each modified row has exactly $3t$ $*$s. Thus, each cluster has exactly three rows. We claim that the corresponding edges should

form a triangle. We can see this as follows: suppose to the contrary the three rows form a 4-star. Let the common vertex be v. Consider the ternary digit $\in \{1, 2\}$ assigned by each of the three edges to v in $conf_0$ — two of the three edges must have assigned the same digit to v. Since these two edges differ in rank, they must have a different configuration (and therefore, a different digit in the column corresponding to v) in at least one of the blocks. Thus, the rows corresponding to the three edges contain an additional $*$ corresponding to vertex v in addition to the $3t$ $*$s corresponding to the remaining three vertices, contradicting the fact that each row has exactly $3t$ $*$s. □

The above proof shows that k-Anonymity is NP-hard even with a ternary alphabet for $k = 3$. By reduction from EDGE PARTITION INTO r-CLIQUES [Kan94], we can extend the above proof for $k = \binom{r}{2}$, for $r \geq 3$. By replicating the graph in the above reduction, we can further extend the proof for $k = \alpha\binom{r}{2}$ for any integer α and $r \geq 3$.

4.3 Algorithm for General k-Anonymity

In this section, we study the problem of k-Anonymity with Generalization for general k and arbitrary alphabet size, and give an $O(k)$-approximation algorithm for the problem. In this section, k-Anonymity refers to the problem of k-Anonymity with Generalization.

Construction of Graph. Given an instance of the k-Anonymity problem, we create an edge-weighted complete graph $G = (V, E)$. The vertex set V contains a vertex corresponding to each vector in the k-Anonymity problem. For two rows $\mathbf{a}$ and $\mathbf{b}$, let the unscaled generalization cost for the j^{th} component, $h_{a,b}(j)$, refer to the lowest level of generalization for attribute j for which the j^{th} components of both $\mathbf{a}$ and $\mathbf{b}$ are in the same partition, *i.e.*, the lowest level for which both have the same generalized value. The weight, $w(e)$, of an edge $e = (a, b)$ is the sum over all components j of the scaled generalization cost, *i.e.*, $w(e) = \sum_j h_{a,b}(j)/l_j$ (recall that the scaling factor l_j corresponds to the total number of levels of generalizations for the j^{th} attribute). The j^{th} attribute is said to contribute a weight of $h_{a,b}(j)/l_j$ to the edge e.

Limitations of the Graph Representation. As mentioned in Section 4.1, with this representation, we lose some information about the structure of the problem, and cannot achieve a better than $\Theta(k)$ approximation factor for the k-Anonymity problem. We show this by giving two instances (on binary alphabet) whose k-Anonymity cost differs by a factor of $\Theta(k)$, but the corresponding graphs for both the instances are identical. Let $l = 2^{k-2}$. For the first instance, take k vectors with kl-dimensions each. The bit positions $(i-1)l+1$ to il are referred to as the i^{th} block of a vector. The i^{th} vector has ones in the i^{th} block and zeros everywhere else. The k-Anonymity cost for this instance is k^2l. For the second instance, take k vectors with $4l = 2^k$ dimensions each. The i^{th} vector breaks up its 2^k dimensions into 2^i equal-sized blocks and has ones in the odd blocks and zeros in the even blocks. This instance incurs a k-Anonymity cost of $4kl$. Note that the graph corresponding to both the instances is a k-clique with all the pairwise distances being $2l = 2^{k-1}$.

Definition 4.1 (Charge of a vertex) *For any given k-Anonymity solution, define the* charge *of a vertex to be the total generalization cost of the vector it represents.*

Idea Behind the Algorithm. Let OPT denote the cost of an optimal k-Anonymity solution, *i.e.*, OPT is the sum of the charges of all the vertices in an optimal k-Anonymity solution. Let $F = \{T_1, T_2, \ldots, T_s\}$, a spanning forest (*i.e.*, a forest containing all the vertices) in which each tree T_i has at least k vertices, be a subgraph of G. This forest describes a feasible partition for the k-Anonymity problem. In the k-Anonymity solution as per this partition, the charge of each vertex is no more than the weight of the tree containing the vertex; recall that the weight of a tree T_i is given by $W(T_i) = \Sigma_{e \in E(T_i)} w(e)$, where $E(T_i)$ denotes the set of edges in tree T_i. We can see this as follows: if attribute j has to be generalized to level r for the vertices in tree T_i (note that an attribute is generalized to the same level for all rows in a cluster), there must exist a pair of vertices (a, b) in the cluster which have an unscaled generalization cost $h_{a,b}(j)$ equal to r. Thus, attribute j contributes a weight of at least r/l_j to the length of all paths (in G) between a and b. In particular, attribute j contributes a weight of at least r/l_j to the weight of tree T_i. Next, we sum the charges of all the vertices to get that the k-Anonymity cost of the partition corresponding to the forest F is at most $\Sigma_i |V(T_i)| W(T_i)$. We will refer to this as the k-Anonymity cost

of the forest. Note that the weight of a forest is simply the sum of the weights of its trees. Hence, the ratio of the k-Anonymity cost to the weight of a forest is at most the number of vertices in the largest tree in the forest. This implies that if we can find a forest with the size of the largest component at most L and weight at most OPT, then we have an L-approximation algorithm. Next, we present an algorithm that finds such a forest with $L \leq \max\{2k-1, 3k-5\}$.

The algorithm has the following overall structure, which is explained in more detail in the next two subsections.

Outline of the Algorithm:

1. Create a forest G with cost at most OPT. The number of vertices in each tree is at least k.

2. Compute a decomposition of this forest (deleting edges is allowed) such that each component has between k and $\max\{2k-1, 3k-5\}$ vertices. The decomposition is done in a way that does not increase the sum of the costs of the edges.

4.3.1 Algorithm for Producing a Forest with Trees of Size at least k

The key observation is that since each partition in a k-Anonymity solution groups a vertex with at least $k-1$ other vertices, the charge of a vertex is at least equal to its distance to its $(k-1)^{st}$ nearest neighbor. The idea is to construct a directed forest such that each vertex has at most one outgoing edge and $(\overrightarrow{u,v})$ is an edge only if v is one of the $k-1$ nearest neighbors of u.

Algorithm FOREST
Invariant:

- The chosen edges do not create any cycle.
- The out-degree of each vertex is at most one.

1. Start with an empty edge set so that each vertex is in its own connected component.

2. Repeat until all components are of size at least k:

Pick any component T having size smaller than k. Let u be a vertex in T without any outgoing edges. Since there are at most $k-2$ other vertices in T, one of the $k-1$ nearest neighbors of u, say v, must lie outside T. We add the edge $(\overrightarrow{u,v})$ to the forest. *Observe that this step does not violate any of the invariants.*

Lemma 4.3.1 *The forest produced by algorithm* FOREST *has minimum tree size at least k and has cost at most OPT.*

Proof: It is evident from the algorithm description that each component of the forest it produces has at least k vertices.

Let the cost of an edge $(\overrightarrow{u,v})$ be paid by vertex u. Note that each vertex u pays for at most one edge to one of its $k-1$ nearest neighbors. As noted earlier, this is less than the charge of this vertex in any k-Anonymity solution. Thus, the sum of costs of all edges in the forest is less than OPT, the total charge of all vertices in an optimal solution. □

In what follows we consider the underlying undirected graph on the edges.

4.3.2 Algorithm to Decompose Large Components into Smaller Ones

We next show how to break any component with size greater than $\max\{2k-1, 3k-5\}$ into two components each of size at least k. Let the size of the component we are breaking be $s > \max\{2k-1, 3k-5\}$.

Algorithm DECOMPOSE-COMPONENT

1. Pick any vertex u as the candidate vertex.
2. Root the tree at the candidate vertex u. Let U be the set of subtrees rooted at the children of u. Let the size of the largest subtree of u be ϕ, rooted at vertex v. If $s-\phi \geq k-1$, then we do one of the following partition and terminate (see Figure 4.3).
   A. If $\phi \geq k$ and $s-\phi \geq k$, then partition the tree into the largest subtree and the rest.
   B. If $s-\phi = k-1$, partition the tree into a component containing the subtrees rooted at the children of v and the rest. To connect the children of v create a

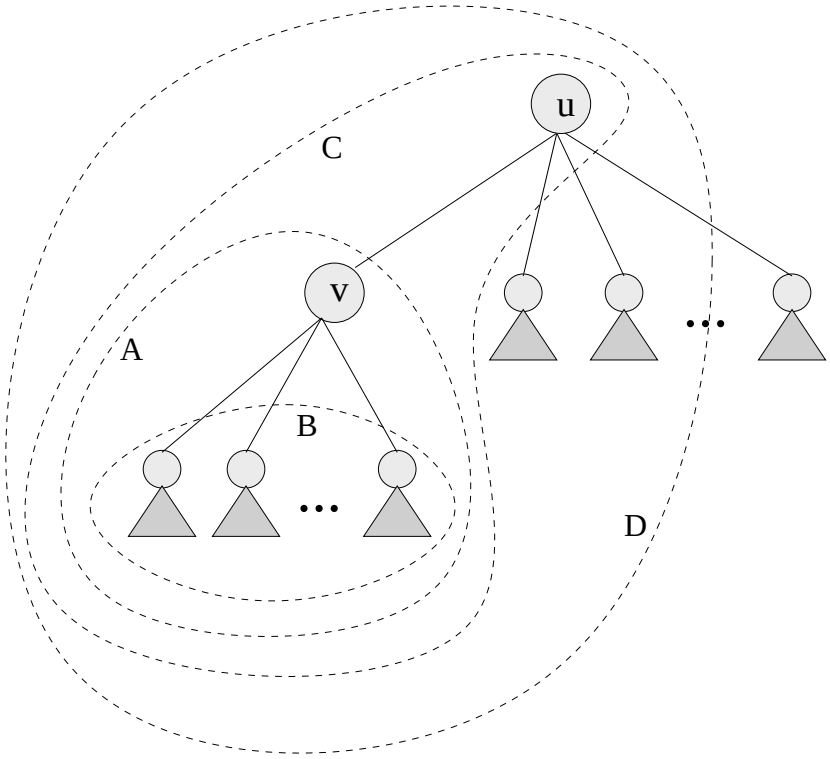


Figure 4.3: The decompositions corresponding to the sub-cases of the algorithm DECOMPOSE-COMPONENT.

dummy vertex v' to replace v. Note that v' is only a Steiner vertex (see Figure 4.4) and does not contribute to the size of the first component. Clearly, the sizes of both the components are at least k.

C. If $\phi = k - 1$, then partition into a component containing the subtree rooted at v along with the vertex u and the rest. In order to connect the children of u in the second component, we create a Steiner vertex u'.

D. Otherwise, all subtrees have size at most $k - 2$. In this case, we create an empty partition and keep adding subtrees of u to it until the first time its size becomes at least $k - 1$. Clearly, at this point, its size is at most $2k - 4$. Put the remaining subtrees (containing at least $k - 1$ vertices, since there are at least $3k - 4$ vertices in all) into the other partition. Observe that since $s \geq 2k$, at most one of the partitions has size equal to $k - 1$. If such a partition exists, add

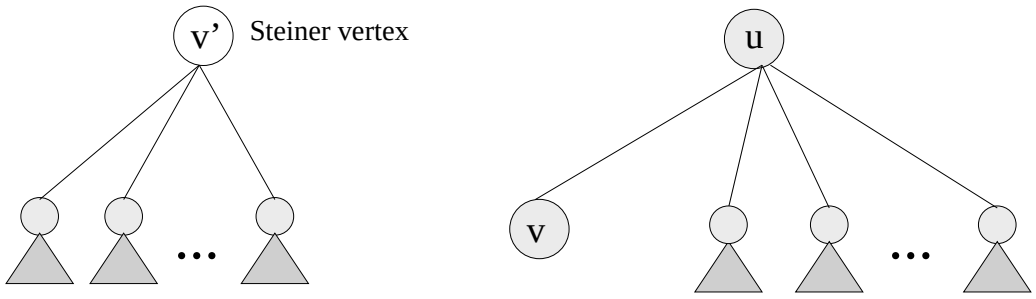


Figure 4.4: The decomposition corresponding to case B; the left partition contains a Steiner vertex v' that does not contribute to its size.

u to that partition, else add u to the first partition. In order to keep the partition not containing u connected, a Steiner vertex u' corresponding to u is placed in it.

3. Otherwise, pick the root of the largest subtree v as the new candidate vertex and go to Step 2.

Lemma 4.3.2 *The above algorithm terminates.*

Proof: We will prove this by showing that the size of the largest component ϕ (in Step 2) decreases in each iteration. Consider moving from candidate vertex u in one iteration to candidate vertex v in the next iteration. Since the algorithm did not terminate with u, if we root the tree at v, then the size of the subtree rooted at u is less than $k-1$. When we consider the largest subtree under v, either it is rooted at u, in which case, it is smaller than $k-1 < s-(k-1)$ and the algorithm terminates in this step; otherwise, the new largest subtree is a subtree of the previous largest subtree. □

Theorem 4.3.3 *There is a polynomial-time algorithm for the k-Anonymity problem, that achieves an approximation ratio of* $\max\{2k-1, 3k-5\}$.

Proof: First, use Algorithm FOREST to create a forest with cost at most OPT and minimum tree size at least k. Then repeatedly apply Algorithm DECOMPOSE-COMPONENT to any component that has size larger than $\max\{2k-1, 3k-5\}$. Note that both these algorithms terminate in $O(kn^2)$ time. □

The above algorithm can also be used when the attributes are assigned weights and the goal is to minimize the weighted generalization cost. In this case, the cost contributed by an attribute to an edge in the graph G is multiplied by its weight. The rest of the algorithm proceeds as before. It is also easy to extend the above analysis to the version of the problem where we allow an entire row to be deleted from the published database, instead of forcing it to pair with at least $k-1$ other rows. The deletion of an entire row is modeled as suppressing all the entries of that row (or generalizing all the entries of that row to the highest level). The objective function is the same as before: minimize the overall generalization cost. We first note that the distance between any two vertices is no more than the cost of deleting a vertex. Thus, if we run the same algorithm as above, the total cost of the forest F produced by Algorithm FOREST is no more than the optimal k-Anonymity cost (this is because the charge of any vertex in the optimal k-Anonymity solution is still no less than its distance to its $(k-1)^{st}$ nearest neighbor). The analysis for the rest of the algorithm remains the same.

4.4 Improved Algorithm for 2-Anonymity

In this section, we study the special case of $k = 2$. The algorithm of the previous section gives a 3-approximation algorithm for this case. We improve upon this result for binary alphabet, and provide a polynomial-time 1.5-approximation algorithm for 2-Anonymity (note that for binary alphabet, generalization is equivalent to suppression). This algorithm uses a technique that is completely different from the previous algorithm, and could potentially be extended to get an improved approximation factor for the general case. For this algorithm, we use the minimum-weight $[1,2]$-factor of a graph constructed from the 2-Anonymity instance. A $[1,2]$-factor of an edge-weighted graph G is defined to be a spanning (*i.e.*, containing all the vertices) subgraph F of G such that each vertex in F has degree 1 or 2. The weight of F is the sum of the weights of the edges in F. Cornuejols [Cor88] showed that a minimum-weight $[1,2]$-factor of a graph can be computed in polynomial time.

Given an instance of the 2-Anonymity problem on binary alphabet, we create an edge-weighted complete graph $G = (V, E)$ as follows. The vertex set V contains a vertex

corresponding to each vector in the 2-Anonymity problem. The weight of an edge (a, b) is the Hamming distance between the vectors represented by a and b (*i.e.*, the number of positions at which they differ). First we obtain a minimum-weight $[1, 2]$-factor F of G. By optimality, F is a vertex-disjoint collection of edges and pairs of adjacent edges (if a $[1, 2]$-factor has a component which is either a cycle or a path of length ≥ 3, we can obtain a $[1, 2]$-factor of smaller weight by removing edge(s)). We treat each component of F as a *cluster*, *i.e.*, retain the bits on which all the vectors in the cluster agree and replace all other bits by $*$s. Clearly, this results in a 2-anonymized table.

Theorem 4.4.1 *The number of $*$s introduced by the above algorithm is at most 1.5 times the number of $*$s in an optimal 2-Anonymity solution.*

Before we prove this theorem, consider three m-bit vectors x_1, x_2 and x_3 with pairwise Hamming distances α, β and γ as shown in Figure 4.5. Without loss of generality, let $\gamma \geq \alpha, \beta$. Let x_{med} denote the *median* vector whose i^{th} bit is the majority of the i^{th} bits of x_1, x_2 and x_3 and let p, q and r be the Hamming distances to x_{med} from x_1, x_2 and x_3 respectively. Let x_s be the *star* vector obtained by minimal suppression of x_1, x_2 and x_3, *i.e.*, it has the common bits where the three vectors agree and $*$s elsewhere. Observe that $\alpha = q + r$, $\beta = r + p$ and $\gamma = p + q$. The other relevant distances are shown in the figure.

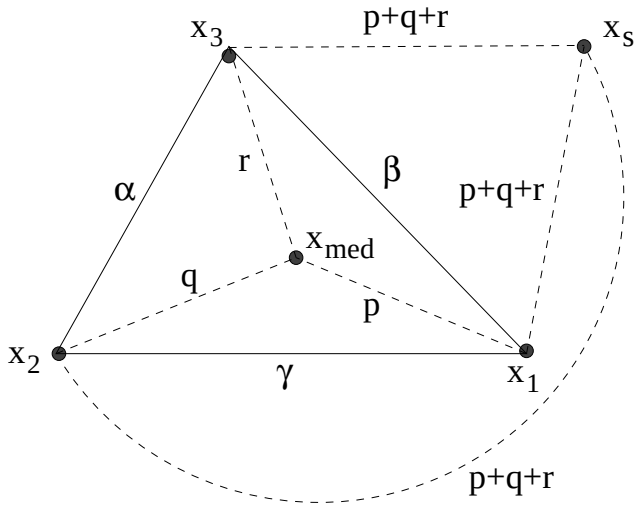


Figure 4.5: Three vectors and their corresponding "median" and "star" vectors

Observation 4.4.2 *If vertices x_1, x_2 and x_3 (as shown in Figure 4.5) form a cluster in a k-Anonymity solution, the number of $*$s in each modified vector is exactly equal to $p+q+r = \frac{1}{2}(\alpha+\beta+\gamma)$. If the cluster contains additional vertices, then the number of $*$s is at least $\frac{1}{2}(\alpha+\beta+\gamma)$.*

To see this, first note that since x_{med} is the median vertex, the attributes that contribute to p, q and r are distinct. Therefore, the number of $*$s in each modified vector is at least $p+q+r$. Moreover, when x_1, x_2 and x_3 are the only three vertices in the cluster, each attribute corresponding to a $*$ in the modified vector contributes to exactly one of p, q and r.

Let c_{OFAC} denote the weight of an optimal $[1,2]$-factor, let c_{ALG} be the cost of the 2-Anonymity solution obtained from it and let OPT denote the cost of the optimal 2-Anonymity solution respectively. The optimal 2-Anonymity solution can be assumed to consist only of disjoint clusters of size 2 or 3 (as bigger clusters can be broken into such clusters without increasing the cost). We can derive a $[1,2]$-factor from this solution as follows: for each cluster of size 2, include the edge between the two vertices; for a cluster of size 3, include the two lighter edges of the triangle formed by the three vertices. Denote the weight of this $[1,2]$-factor by c_{FAC}.

Lemma 4.4.3 $c_{ALG} \leq 3 \cdot c_{OFAC}$

Proof: Consider the optimal $[1,2]$-factor and the k-Anonymity solution corresponding to it. For a cluster of size 2, we have to suppress all the bits at which the two vectors differ so that the total number of $*$s in the two rows is twice the Hamming distance (which is equal to the edge weight). For a cluster of size 3, say the one in the figure, by Observation 4.4.2, the number of $*$s in each row is exactly $(\alpha+\beta+\gamma)/2$. So, the total number of stars is $\frac{3}{2}(\alpha+\beta+\gamma) \leq 3(\alpha+\beta)$ (using triangle inequality). The optimal $[1,2]$-factor would have contained the two lighter edges of the triangle, incurring a cost of $(\alpha+\beta)$ for this cluster. Summing over all the clusters formed by the optimal $[1,2]$-factor algorithm, we get $c_{ALG} \leq 3 \cdot c_{OFAC}$. □

Lemma 4.4.4 $c_{FAC} \leq \frac{1}{2}OPT$

Proof: Consider the optimal k-Anonymity solution and the $[1,2]$-factor corresponding to it. For a cluster of size 2, cost incurred by the $[1,2]$-factor FAC is equal to half the cost incurred in OPT. For a cluster of size 3, say the one in Figure 4.5, cost incurred in FAC is equal to $\alpha+\beta \leq \frac{2}{3}(\alpha+\beta+\gamma) = \frac{4}{3}(p+q+r)$, where the inequality is obtained by using the fact $\gamma \geq \alpha, \beta$. Since the cost incurred in OPT is $3(p+q+r)$, cost incurred in FAC is at most half the cost incurred in OPT. By summing over all the clusters, we get $c_{FAC} \leq OPT/2$. □

Since $c_{OFAC} \leq c_{FAC}$, it follows from the above lemmas that $c_{ALG} \leq \frac{3}{2}OPT$, which proves Theorem 4.4.1. For an arbitrary alphabet size, x_{med} is no longer defined. However, it can be shown that $OPT \geq (\alpha+\beta+\gamma) \geq \frac{3}{2}(\alpha+\beta)$, proving $c_{FAC} \leq \frac{2}{3}OPT$. Since $c_{ALG} \leq 3 \cdot c_{OFAC}$ holds as before, we get $c_{ALG} \leq 2 \cdot OPT$. Thus, the same algorithm achieves a factor 2 approximation for 2-Anonymity with Suppression for arbitrary alphabet size.

4.5 Improved Algorithm for 3-Anonymity

We now present a 2-approximation algorithm for 3-Anonymity with a binary alphabet (again generalization is equivalent to suppression in this case). The idea is similar to the algorithm for 2-Anonymity. We construct the graph G corresponding to the 3-Anonymity instance as in the previous algorithm. A 2-factor of a graph is a spanning subgraph with each vertex having degree 2 (in other words, a collection of vertex-disjoint cycles spanning all the vertices). We first run the polynomial-time algorithm to find a minimum-weight 2-factor F of the graph G [Cor88]. We show that the cost of this 2-factor, say c_{OFAC}, is at most $2/3$ times the cost of the optimal 3-Anonymity solution, say OPT. Then, we show how to transform this 2-factor F into a 3-Anonymity solution ALG of cost $c_{ALG} \leq 3 \cdot c_{OFAC}$, giving us a factor-2 approximation algorithm for 3-Anonymity.

Lemma 4.5.1 *The cost of the optimal 2-factor, c_{OFAC} on graph G corresponding to the vectors in the 3-Anonymity instance is at most $\frac{2}{3}$ times the cost of the optimal 3-Anonymity solution, OPT.*

Proof: Consider the optimal 3-Anonymity solution. Observe that it will cluster 3, 4 or 5 vertices together (any larger groups can be broken up into smaller groups of size at least 3, without increasing the cost of the solution). Given an optimal solution to the 3-Anonymity problem, we construct a 2-factor solution as follows: for every cluster of the 3-Anonymity solution, pick the minimum-weight cycle involving the vertices of the cluster. Next, we analyze the cost c_{FAC} of this 2-factor. Define the *charge* of a vertex to be the number of $*$s in the vector corresponding to this vertex in the 3-Anonymity solution. We consider the following three cases:

(a) If a cluster i is of size 3, the 2-factor contains a triangle on the corresponding vertices. Let a, b and c be the lengths of the edges of the triangle. By Observation 4.4.2, we get that $(a+b+c)$ is twice the charge of each vertex in this cluster. Thus, OPT pays a total cost of $OPT_i = \frac{3}{2}(a+b+c)$ while FAC pays $c_{FAC,i} = a+b+c = \frac{2}{3}OPT_i$.

(b) If a cluster i is of size 4, the 2-factor corresponds to the cheapest 4-cycle on the four vertices. Let τ be the sum of the weights of all the $\binom{4}{2} = 6$ edges on these four vertices. Consider the three 4-cycles on these vertices. As each edge appears in two 4-cycles, the average cost of a 4-cycle is $\frac{2}{3}\tau$. By choosing the minimum weight 4-cycle, we ensure that the cost paid by FAC for these vertices $c_{FAC,i} \leq \frac{2}{3}\tau$. Also, by Observation 4.4.2, the charge of any of these 4 vertices is at least half the cost of any triangle on (three of) these four vertices. The cost of the most expensive triangle is at least equal to the average cost over all the $\binom{4}{3} = 4$ triangles, which is equal to $\frac{2}{4}\tau$ (since each edge appears in two triangles). Hence the cost paid by OPT, $OPT_i \geq 4 \cdot \frac{1}{2} \cdot \frac{2}{4} \cdot \tau = \tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

(c) If a cluster i is of size 5, let τ be the sum of weights of all $\binom{5}{2} = 10$ edges on these five vertices. By an argument similar argument to (b), FAC pays $c_{FAC,i} \leq \frac{5}{10}\tau$. Also, the charge of any of these vertices is at least half the cost of any triangle on (three of) these vertices. Since the average cost of a triangle is $\frac{3}{10}\tau$, the number of $*$s in each vertex is at least $\frac{1}{2}\frac{3}{10}\tau$. Thus, cost paid by OPT for cluster i, $OPT_i \geq 5 \cdot \frac{1}{2} \cdot \frac{3}{10} \cdot \tau = \frac{3}{4}\tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

Thus, adding up over all clusters, we get $c_{FAC} \leq \frac{2}{3}OPT$. Thus, $c_{OFAC} \leq \frac{2}{3}OPT$. □

Lemma 4.5.2 *Given a 2-factor F with cost c_F, we can get a solution for 3-Anonymity of cost $c_{ALG} \leq 3 \cdot c_F$.*

Proof: To get a solution for 3-Anonymity, we make every cycle in F with size 3, 4 or 5 into a cluster. Let $\text{len}(C)$ denote the length of a cycle C in the 2-factor. For each cycle larger C, if $\text{len}(C) = 3x$ for x an integer, then we decompose it into x clusters, each containing 3 adjacent vertices of C. Similarly, if $\text{len}(C) = 3x + 1$, x an integer, we decompose it into x clusters: $x - 1$ of size 3, and one of size 4. If $\text{len}(C) = 3x + 2$, x an integer, then we decompose it into $x - 2$ clusters of size 3, and two clusters of size 4. In all these cases, of all the possible decompositions, we choose the one in which the total cost of edges of the cycle within the clusters is minimized. Depending on the size of the cycle C in the 2-factor, we can show that the 3-Anonymity solution ALG pays as follows:

(a) For a triangle, ALG pays $3 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(b) For a 4-cycle, ALG pays at most $4 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(c) For a 5-cycle, ALG pays at most $5 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

The above inequalities follow from an observation similar to Observation 4.4.2, namely that the vertices of a cycle C can differ in at most $\frac{1}{2}\text{len}(C)$ attributes.

(e) For a $(3x+1)$-cycle, $x > 1$, ALG pays at most $\frac{6(x-1)+12}{3x+1} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by considering the minimum 3-Anonymity cost over the $(3x+1)$ possible decompositions into clusters. Each edge e of the cycle C appears in a cluster of size 4 in three decompositions and contributes a cost of at most $4w(e)$ to the k-Anonymity cost of the decomposition. In addition, each edge appears in a cluster of size 3 in $(2(x-1))$ decompositions contributing a cost of at most $3w(e)$ to the k-Anonymity cost of these decompositions. Summing over all edges, the total k-Anonymity cost of all the $3x + 1$ decompositions is at most $(3 \cdot 2(x-1) + 4 \cdot 3) \cdot \text{len}(C)$ and ALG pays no more than the average cost of a decomposition.

(f) For a $(3x+2)$-cycle, $x > 1$, ALG pays at most $\frac{6(x-2)+24}{3x+2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by an analysis similar to (e) above. Note that we get a better bound on

the cost by splitting into $x-2$ clusters of size 3 and two clusters of size 4, instead of $x-1$ clusters of size 3 and one clusters of size 5.

Thus, summing over all clusters, ALG pays no more than three times the total cost of all cycles, *i.e.*, $c_{ALG} \leq 3 \cdot c_F$. □

Note that the above analysis is tight, since equality can hold in case (f), when $x = 2$, e.g. for vectors $\{0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000\}$, where the optimal 2-factor is a cycle through all the vertices in the given order.

Combining the above lemmas, we obtain a factor 2 approximation for 3-Anonymity.

4.6 Summary and Future Work

We showed that the k-Anonymity problem is NP-hard even when the attribute values are ternary and we are allowed only to suppress entries. Then we gave an $O(k)$-approximation algorithm for k-Anonymity with Generalization for arbitrary k and arbitrary alphabet size. For a binary alphabet, we provided improved approximation algorithms for $k = 2$ (an approximation factor of 1.5) and $k = 3$ (an approximation factor of 2). We also showed that for k-Anonymity, it is not possible to achieve an approximation factor better than $k/4$ by using the graph representation. It would also be interesting to see a hardness of approximation result for k-Anonymity without assuming the graph representation.

Releasing a database after k-anonymization prevents definitive *record linkages* with publicly available databases [Swe02]. In particular, for each record in the public database, at least k records in the k-anonymized database could correspond to it, which hides each individual in a crowd of k other people. The privacy parameter k must be chosen according to the application in order to ensure the required level of privacy. One source of concern about the k-anonymization model is that for a given record in the public database, all the k records corresponding to it in the anonymized database might have the same value of the sensitive attribute(s) ("Diseases" in our examples), thus revealing the sensitive attribute(s) conclusively. To address this issue, we could add a constraint that specifies that for each cluster in the k-anonymized database, the sensitive attribute(s) should take at least r distinct values. Recently Machanavajjhala *et al.* [MKGV06] propose imposing additional constraints

that there be a good representation of sensitive attributes for each block of k-anonymized records.

Another interesting direction of research is to extend the basic k-Anonymity model to deal with changes in the database. A hospital may want to periodically release an anonymized version of its patient database. However, releasing several anonymized versions of a database might leak enough information to enable *record linkages* for some of the records. It would be useful to extend the k-Anonymity framework to handle inserts, deletes, and updates to a database.

Chapter 5

Achieving Anonymity via Clustering

In this chapter, we continue to address the problem of publishing information from a table containing personal records, while maintaining individual privacy. As discussed in Chapter 4, the traditional approach of de-identifying records by removing the identifying fields such as name, address, and social security number is not sufficient to protect privacy. This is because joining this de-identified table with a publicly available database (like the voter list) on columns like age, race, gender, and zip code can be used to identify individuals. In fact, Sweeney [Swe00] observed that for 87% of the population in the United States, the combination of non-key fields like Date of Birth, Gender, and Zip Code corresponded to a unique person. Such non-key fields are called *quasi-identifiers*. In what follows we assume that the identifying fields have been removed and that the table has two types of attributes: (1) the quasi-identifying attributes explained above and (2) the sensitive attributes (such as disease) that need to be protected.

In order to protect privacy, Samarati and Sweeney [Sam01, SS98, Swe02] proposed the k-Anonymity model, where some of the quasi-identifier fields are suppressed or generalized so that, for each record in the modified table, there are at least $k-1$ other records in the modified table that are identical to it along the quasi-identifying attributes. For the table in Figure 5.1(a), Figure 5.1(b) shows a 2-anonymized table corresponding to it. The columns corresponding to sensitive attributes, like disease in this example, are retained

Age	Location	Disease
α	β	Flu
α + 2	β	Flu
δ	γ + 3	Hypertension
δ	γ	Flu
δ	γ -3	Cold

(a) Original table

Age	Location	Disease
*	β	Flu
*	β	Flu
δ	*	Hypertension
δ	*	Flu
δ	*	Cold

(b) 2-anonymized version

Age	Location	NumPoints	Disease
α +1	β	2	Flu Flu
δ	γ	3	Hypertension Flu Cold

(c) 2-gather clustering, with maximum radius 3

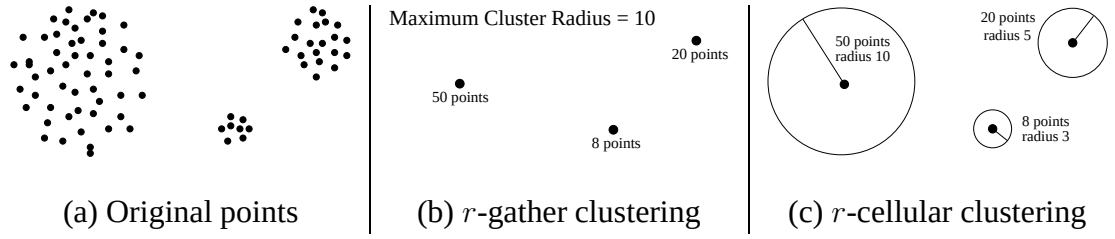
Age	Location	NumPoints	Radius	Disease
α +1	β	2	1	Flu Flu
δ	γ	3	3	Hypertension Flu Cold

(d) 2-cellular clustering, with total cost 11

Figure 5.1: Original table and three different ways of achieving anonymity

without change. The aim is to provide a k-anonymized version of the table with the minimum amount of suppression or generalization of the table entries. An $O(k \log k)$ approximation algorithm was first proposed for the problem of k-Anonymity with suppressions only [MW04]. In Chapter 4, we described our improved algorithm that achieves an $O(k)$ approximation for the general version of the problem [AFK$^+$05b].

In this chapter, instead of generalization and suppression, we propose a new technique for anonymizing tables before their release. We first use the quasi-identifying attributes to define a metric space (*i.e.*, pairwise distances have to satisfy the triangle inequality) over



(a) Original points (b) r-gather clustering (c) r-cellular clustering

Figure 5.2: Publishing anonymized data

the database records, which are then viewed as points in this space. This is similar to the approach taken in [CDM$^+$05], except that we do not restrict ourselves to points in $\mathcal{R}^d$; instead, we allow our points to be in an arbitrary metric space. We then cluster the points and publish only the final cluster centers along with some cluster size and radius information. Our privacy requirement is similar to the k-Anonymity framework – we require each cluster to have at least r points[1]. Publishing the cluster centers instead of the individual records, where each cluster represents at least r records, gives privacy to individual records, but at the same time allows data-mining tools to infer macro trends from the database.

In the rest of the chapter we will assume that a metric space has been defined over the records, using the quasi-identifying attributes. For this, the quasi-identifying attributes may need to be remapped. For example, zip codes could first be converted to longitude and latitude coordinates to give a meaningful distance between locations. A categorical attribute, *i.e.*, an attribute that takes n discrete values, can be represented by n equidistant points in a metric space. Furthermore, since the values of different quasi-identifying attributes may differ by orders of magnitude, we need to weigh the attributes appropriately while defining the distance metric. For example, the attribute location may have values that differ in orders of 10 miles with a maximum of 1000 miles, while the attribute age may differ by a single year with a maximum of 100 years. In this case we assume that the attribute location is divided by 10 and the attribute age retained without change if both attributes are needed to have the same relative importance in the distance metric. For the example we provide in Figure 5.1, we assume that the quasi-identifying attributes have already been scaled. We observe that it is quite complicated to algorithmically derive a metric space over quasi-identifying attributes of records; we leave this problem as an avenue for future work.

[1]We use r instead of k, as k is traditionally used in clustering to denote the number of clusters.

To publish the clustered database, we publish three types of features for each cluster: (1) the quasi-identifying attribute values for the cluster center (age and location in our example), (2) the number of points within the cluster, and (3) a set of values taken by the sensitive attributes (disease in our example). We will also publish a measure of the quality of the clusters. This will give a bound on the error introduced by the clustering.

We consider two cluster-quality measures. The first one is the maximum cluster radius. For this we define the r-GATHER problem, which aims to minimize the maximum radius among the clusters, while ensuring that each cluster has at least r members. As an example, r-GATHER clustering with minimum cluster size $r = 2$, applied to the table in Figure 5.1(a) gives the table in Figure 5.1(c). In this example, the maximum radius over all clusters is 3. As another example, Figure 5.2(b) gives the output of the r-GATHER algorithm applied to the quasi-identifiers, shown as points in a metric space in Figure 5.2(a). Our formulation of the r-GATHER problem is related to, but not to be confused with, the classic k-CENTER problem [HS85]. The k-CENTER problem has the same objective of minimizing the maximum radius among the clusters, however, the constraint is that we can have no more than k clusters in total. The r-GATHER problem is different from k-CENTER problem in that instead of specifying an upper bound on the number of clusters, we specify a lower bound on the number of points per cluster as part of the input. Further the constraint of at least r points per cluster implies that we can have no more than n/r number of clusters, where n is the total number of points in our data set.

We also consider a second (more verbose) candidate for indicating cluster-quality, whereby we publish the radius of each cluster, rather than just the maximum radius among all clusters. For each point within a cluster, the radius of the cluster gives an upper bound on the distortion error introduced. Minimizing this distortion error over all points leads to the *cellular* clustering measurement that we introduce. More formally, the *cellular* clustering measurement over a set of clusters, is the sum, over all clusters, of the products of the number of points in the cluster and the radius of the cluster. Using this as a measurement for anonymizing tables, we define the r-CELLULAR CLUSTERING problem as follows: Given points in a metric space, the goal is to partition the points into cells, a.k.a. clusters, each of size at least r, and the cellular clustering measurement is minimized. Consider again the data in Figure 5.1(a). Figure 5.1(d) shows a r-cellular cluster solution with minimum

cluster size $r = 2$. The total cost is $2 \times 1 + 3 \times 3 = 11$. Also, Figure 5.2(c) gives the output of the r-CELLULAR CLUSTERING algorithm applied to the quasi-identifiers shown as points in a metric space in Figure 5.2(a). The total cost of the solution in Figure 5.2(c) is: $50 \times 10 + 20 \times 5 + 8 \times 3 = 624$. As this cellular clustering objective could be relevant even in contexts other than anonymity, we study a slightly different version of the problem: similar to the FACILITY LOCATION problem [JV99], we add an additional setup cost for each potential cluster center, associated with opening a cluster centered at that point, but we do not have the lower bound on the number of points per cluster. We call this the CELLULAR CLUSTERING problem. In fact, we will use the setup costs in the CELLULAR CLUSTERING problem formulation to help us devise an algorithm that solves r-CELLULAR CLUSTERING.

Comparison with k-Anonymity. While k-Anonymity forces one to suppress or generalize an attribute value even if all but one of the records in a cluster have the same value, the above clustering-based anonymization technique allows us to pick a cluster center whose value along this attribute dimension is the same as the common value, thus enabling us to release more information without losing privacy. For example, consider the table in Figure 5.3 with the Hamming distance metric on the row vectors. If we wanted to achieve 5-Anonymity, we will have to hide all the entries in the table, resulting in a total distortion of 20. On the other hand, a 5-CELLULAR CLUSTERING solution could use $(1, 1, 1, 1)$ as the cluster center with a cluster radius of 1. This will give a total distortion bound of 5 (the actual distortion is only 4).

	Attr1	Attr2	Attr3	Attr4
Record 0	1	1	1	1
Record 1	0	1	1	1
Record 2	1	0	1	1
Record 3	1	1	0	1
Record 4	1	1	1	0

Figure 5.3: A sample table where there is no common attribute among all entries.

Just like k-Anonymity, r-GATHER and r-CELLULAR CLUSTERING are sensitive to

outlier points, with just a few outliers capable of increasing the cost of the clustering significantly. To deal with this problem, we generalize the above algorithms to allow an ϵ fraction of the points to be deleted before publication. By not releasing a small fraction of the database records, we can ensure that the data published for analysis has less distortion and hence is more useful. This can be done as long as our aim is to infer macro trends from the published data. On the other hand, if the goal is to find out anomalies, then we should not ignore the outlier points. There has been no previous work for k-Anonymity with this generalization.

We note that, as in k-Anonymity, the objective function is oblivious to the sensitive attribute labels. Extensions to the k-Anonymity model, such as the notion of l-diversity [MKGV06], can be applied independently to our clustering formulation.

We provide constant-factor approximation algorithms for both the r-GATHER and r-CELLULAR CLUSTERING problems. In particular, we first show that the it is NP-hard to approximate the r-GATHER problem better than 2 and provide a matching upper bound. We then provide extensions of both these algorithms to allow for an ϵ fraction of unclustered points, which we call the (r, ϵ)-GATHER and (r, ϵ)-CELLULAR CLUSTERING, respectively. These are the first constant-factor approximation algorithms for publishing an anonymized database. The best known algorithms [AFK$^+$05b, MW04] for previous problem formulations had an approximation ratio linear in the anonymity parameter r.

The rest of this chapter is organized as follows. In Section 5.1, we present a tight 2-approximation algorithm for the r-GATHER problem and its extension to the (r, ϵ)-GATHER problem. Next, in Section 5.2, motivated by the desire to reduce the sum of the distortions experienced by the points, we introduce the problem of CELLULAR CLUSTERING. We present a primal-dual algorithm for the problem without any cluster-size constraints that achieves an approximation ratio of 4. We then study the additional constraint of having a minimum cluster size of r. Finally, we relax the problem by allowing the solution to leave at most an ϵ fraction of the points unclustered. We conclude in Section 5.3.

5.1 r-Gather Clustering

To publish the clustered database, we publish three types of features for each cluster: (1) the quasi-identifying attribute values for the cluster center, (2) the number of points within the cluster, and (3) a set of values taken by the sensitive attributes. The maximum cluster radius is also published to give a bound on the error introduced by clustering. This is similar to the traditionally studied k-CENTER clustering. In order to ensure r-Anonymity, we do not restrict the total number of clusters, instead, we pose the alternative restriction that each cluster should have at least r records assigned to it. We call this problem r-GATHER, which we formally define below.

Definition 5.1 *The r-GATHER problem is to cluster n points in a metric space into a set of clusters, such that each cluster has at least r points. The objective is to minimize the maximum radius among the clusters.*

We note that the minimum cluster size constraint has been considered earlier in the context of facility location [KM00].

We first show the reduction for NP-completeness and hardness proofs.

5.1.1 Lower Bound

We show that this problem is NP-complete by a reduction from the 3-Satisfiability problem, where each literal belongs to at most 3 clauses [GJ79].

Suppose that we have a boolean formula $\mathcal{F}$ in 3-CNF form with m clauses and n variables. Let $\mathcal{F} = C_1 \wedge \ldots \wedge C_m$, be a formula composed of variables $x_i, i = 1 \ldots n$ and their complements $\overline{x_i}$.

From the boolean formula, we create a graph $G = (V, E)$ with the following property: There is a solution to the r-GATHER problem with a cluster radius of 1, with respect to the shortest distance metric on the graph G, if and only if $\mathcal{F}$ has a satisfying assignment.

We create the graph as follows: For each variable x_i, create two vertices v_i^T and v_i^F, and create an edge (v_i^T, v_i^F) between the two vertices; in addition create a set S_i of $(r-2)$ nodes and add edges from each node in S_i to both v_i^T and v_i^F. Picking v_i^T (v_i^F) as a center corresponds to setting $x_i = T$ (F). (Note that we cannot choose both v_i^T and v_i^F since there

are not enough nodes in S_i.) For each clause C_j, create a new node u_j that is adjacent to the nodes corresponding to the literals in the clause. For example, if $C_1 = (x_1 \vee \overline{x_2})$ then we add edges from u_1 to v_1^T and v_2^F.

If the formula is indeed satisfiable, then there is a clustering by picking v_i^T as a center if $x_i = T$ and picking v_i^F otherwise. Each clause is true, and must have a neighbor chosen as a center. Moreover by assigning S_i to the chosen center, we ensure that each center has at least r nodes in its cluster.

Now suppose there is an r-gather clustering. If $r > 6$ then both v_i^T and v_i^F cannot be chosen as centers. In addition, the clause nodes u_j have degree at most 3 and cannot be chosen as centers. If exactly one of v_i^T or v_i^F is chosen as a center, then we can use this to find the satisfying assignment. The assignment is satisfying as each clause node has some neighbor at distance 1 that is a chosen center, and makes the clause true.

This completes the NP-completeness proof. Note that this reduction also gives us a hardness of 2. We just showed that there is a solution to the r-GATHER problem with a cluster radius of 1 if and only if $\mathcal{F}$ had a satisfying assignment. The next available cluster radius is 2 in the metric defined by the graph G.

5.1.2 Upper Bound

We first use the threshold method used for k-CENTER clustering to guess R, the optimal radius for r-GATHER. The choices for R are defined as follows. We will try all values $\frac{1}{2}d_{ij}$ where d_{ij} is the distance between points i and j. Note that this defines a set of $O(n^2)$ distance values. We find the smallest R for which the following two conditions hold:

Condition (1) Each point p in the database should have at least $r - 1$ other points within distance $2R$ of p.

Condition (2) Let all nodes be unmarked initially. Consider the following procedure: Select an arbitrary unmarked point p as a center. Select all unmarked points within distance $2R$ of p (including p) to form a cluster and mark these points. Repeat this as long as possible, until all points are marked. Now we try to reassign points to clusters to meet the requirement that each cluster has size at least r. This is done as

follows. Create a flow network as follows. Create a source s and sink t. Let C be the set of centers that were chosen. Add edges with capacity r from s to each node in C. Add an edge of unit capacity from a node $c \in C$ to a node $v \in V$ if their distance is at most $2R$. Add edges of unit capacity from nodes in V to t and check to see if a flow of value $r|C|$ can be found (saturating all the edges out of s). If so, then we can obtain the clusters by choosing the nodes to which r units of flow are sent by a node $c \in C$. All remaining nodes of V can be assigned to any node of C that is within distance $2R$. If no such flow exists, we exit with failure.

The following lemma guarantees that the smallest R that satisfies these conditions is a lower bound on the value of the optimal solution for r-GATHER. Suppose we have an optimal clustering $S_1, \ldots, S_\ell$ with ℓ clusters. Let the maximum diameter of any of these clusters be d^* (defined as the maximum distance between any pair of points in the same cluster).

Lemma 5.1.1 *When we try $R = \frac{d^*}{2}$, then the above two conditions are met.*

Proof: By the definition of r-GATHER, every point has at least $r-1$ other points within the optimal diameter, and hence within distance $2R$. Consider an optimal r-GATHER clustering. For each point i, all points belonging to the same optimal cluster c as the point i are within a distance $2R$ of i. Thus, in the procedure of Condition (2), as soon as any point in c is selected to open a new cluster, all unmarked points belonging to c get assigned to this new cluster. So at most one point from each optimal cluster is chosen as a center and forms a new cluster. We would now like to argue that the reassignment phase works correctly as well. Let S be the set of chosen centers. Now consider an optimal solution with clusters, each of size at least r. We can assign each point of a cluster to the center that belongs to that cluster, if a center was chosen in the cluster. Otherwise, since the point was marked by the algorithm, some center was chosen that is within distance $2R$. We can assign this point to the center that had marked it. Each chosen center will have at least r points assigned to it (including itself). □

Since we find the smallest R, we will ensure that $R \leq d^*/2 \leq R^*$ where R^* is the radius of the optimal clustering. In addition, our solution has radius $2R$. This gives us a 2-approximation.

Theorem 5.1.2 *There exists a polynomial time algorithm that produces a 2-approximation to the* r-GATHER *problem.*

5.1.3 (r, ϵ)-Gather Clustering

A few outlier points can significantly increase the clustering cost under the minimum cluster size constraint. We consider a relaxation whereby the clustering solution is allowed to leave an ϵ fraction of the points unclustered, *i.e.*, to delete an ϵ fraction of points from the published k-anonymized table. Charikar *et al.* [CKMN01] studied various facility location problems with this relaxation and gave constant-factor approximation algorithms for them.

For the (r, ϵ)-GATHER problem, where each cluster is constrained to have at least r points and an ϵ fraction of the points are allowed to remain unclustered, we modify our r-GATHER algorithm to achieve a 4-approximation. We redefine the condition to find R. We find the smallest R that satisfies the following condition: There should be a subset S of points containing at least $1 - \epsilon$ fraction of the points, such that each point in S has at least $r - 1$ neighbors within distance $2R$ in S.

This condition can be checked in $O(n^2)$ time by repeatedly removing any point in S that has fewer than $r - 1$ other points in S within distance $2R$ of itself, with S initially being the entire vertex set. It is clear that the smallest R we found is no more than R^*, the optimal radius.

Let R be the value that we found. Let $N(v)$ denote the set of points in S within distance $2R$ of v, including v itself. We know then $N(v) \geq r$. We then consider the following procedure: Select an arbitrary point p from S. If there are at least $r - 1$ other points within distance $2R$ of p, then form a new cluster and assign p and all points within distance $2R$ of p to this cluster. Remove all these points from further consideration and repeat this process until all remaining points have fewer than $r - 1$ other points within distance $2R$ of them. Let U be the set of points left unclustered at the end of this process. For each $u \in U$, there exists a point $p \in N(u)$ such that p is assigned to some cluster c in the procedure of forming clusters. We can see this as follows. Since u was left unassigned at the end of the procedure, there are fewer than r unassigned points remaining in $N(u)$. This implies that there is at least one point p in $N(u)$ which is already assigned to some cluster c. We assign

u to c, which already has at least r points.

Thus, we have assigned all points to clusters, such that each cluster has at least r points. Note that the radius of each cluster is no more than $4R$. This gives us the following theorem.

Theorem 5.1.3 *There exists a polynomial time algorithm that produces a 4-approximation to the* (r, ϵ)-GATHER *problem.*

We note that in the problem formulation of (r, ϵ)-GATHER, if we require the cluster centers to be input points, instead of arbitrary points in the metric, then we can improve the approximation factor to 3 as follows. In the filtering step we define "candidates" as the set of points that have at least r points within radius R. The total number of points within distance R of the candidates should contain at least $1 - \epsilon$ fraction of the points. Call this set S. Each point in S has at least $r - 1$ neighbors within distance $2R$ in S. In the initial phase we greedily pick clusters of radius R (instead of $2R$) that have at least r points and mark those points covered. If a point in S is now uncovered, it must have a candidate within distance R that was unable to form a cluster. This is because some of the points within distance R of the candidate were covered in the first phase by disks of radius R. Hence each point in S can reach such a cluster center within distance $3R$ (through the candidate).

5.1.4 Combining r-Gather with k-Center

We can combine the r-GATHER problem with the k-CENTER problem and have the two constraints present at the same time. That is, we minimize the maximum radius, with the constraint that we have no more than k clusters, each must have at least r members. We call this the (k, r)-CENTER problem.

We note that a similar problem has been studied before in the k-CENTER literature. That is, instead of having a lower bound r on the cluster size as an additional constraint to the original k-CENTER formulation, an upper bound on the cluster size is specified. This is called the CAPACITATED k-CENTER problem [KS00]. Bar-Ilan, Kortsarz, and Peleg [BIKP93] gave the first constant approximation factor of 10 for this problem. The bound was improved subsequently to 5 by Khuller and Sussmann [KS00]. In this subsection, we only concentrate on the (k, r)-CENTER problem defined above.

We note here that the algorithm developed for r-GATHER in Section 5.1.2 can be extended to provide a 2-approximation for the (k, r)-CENTER problem. We just have to add to Condition (2) the extra criteria that if the number of centers chosen exceeds k then exit with failure, *i.e.*, try a different value for R. We can show that Lemma 5.1.1 holds for the modified conditions, hence an approximation factor of 2.

We also consider the outlier version of this problem, namely, the (k, r, ϵ)-CENTER problem.

We show that the following algorithm is a 4-approximation algorithm for the (k, r, ϵ)-CENTER problem.

Fix a guess for the optimal radius R (choose the smallest R that succeeds). For each such guess, we apply the following algorithm. Let $D(v, \delta)$ be the set of points within distance δ of v (including v).

Algorithm:

(Filtering Step) Let S be the set of points v such that $|D(v, 2R)| \geq r$. Check to see if $|S| \geq (1 - \epsilon)n$, otherwise exit with failure. From now on we only consider points in S.

(Greedy Step) We now choose up to k centers. We put the centers in the set Q. Initially Q is empty. All points are uncovered to start with. Let $N(v, \delta)$ be the set of *uncovered points* within distance δ of v (including v itself). Points are either uncovered, or covered. Once a point is covered it is removed from consideration. At each step i, we choose a center c_i that satisfies the following criteria:

(a) c_i is uncovered.

(b) $|N(c_i, 2R)|$ is maximum.

All uncovered points in $N(c_i, 4R)$ are then marked as covered.

After Q is completely decided, check that the total points covered is at least $(1 - \epsilon)n$, otherwise exit with failure.

(Assignment step): Form clusters as follows. For each $c_i \in Q$, form a cluster C_i centered at c_i. Each covered point is assigned to its closest cluster center.

For each c_i, we denote $G_i = N(c_i, 2R)$ and $E_i = N(c_i, 4R)$, which are uncovered points within distance $2R$ and $4R$ of c_i, when c_i is chosen.

In Figure 5.4 we illustrate this algorithm via an example. Consider the Greedy Step and assume for a moment that R is indeed the optimal radius just for illustration purposes. Let

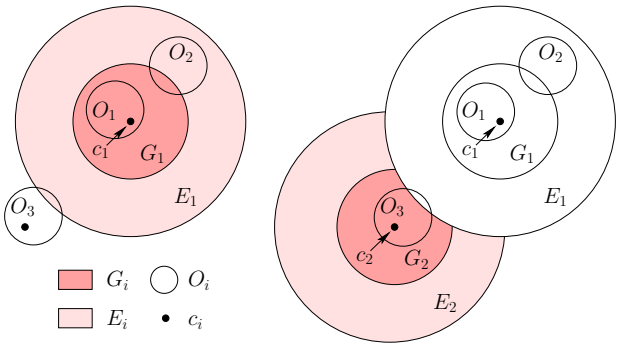


Figure 5.4: Optimal clusters and the greedy step

the optimal solution have clusters $O_1, \ldots, O_k$. In the figure, we only show cluster centers to be picked and omit the rest of the points. The left side illustrates the situation when we are picking c_1. Note that G_1 includes O_1 completely, and overlaps with O_2. Because of this, all points in O_1 and O_2 are in E_1 and marked covered and cannot be chosen as a center later. Note that E_1 in fact will cover points in other optimal clusters as well. For example, when we choose c_1 and cover all points in E_1, we also cover some points in O_3. However, we may still pick a remaining point in O_3 as the next cluster center c_2, as shown in the right side. Note that in the Greedy Step, we completely ignore the constraint of r, as we are not forming any clusters but only picking cluster centers. In fact, G_2 now could have fewer than r uncovered points. The key is that the G_i's are far apart. Hence in the Assignment Step, all the points in $D(c_2, 2R)$ that were initially covered by E_1 will eventually be assigned to the center c_2, giving the whole cluster C_2 at least r points. Detailed proofs are below.

Lemma 5.1.4 *After the assignment step, each cluster formed has at least r points, and radius at most $4R$.*

Proof: Every time a center c_i is selected, we only cover points within distance $4R$, thus the maximum radius is at most $4R$. In the end, each point is assigned to its closest chosen center in Q. Observe that the cluster centers are more than $4R$ apart. Thus for each center c_i and its corresponding cluster C_i, all the points within distance $2R$ of c_i are assigned to the same cluster C_i. By the filtering step, we know that $|C_i| \geq |D(c_i, 2R)| \geq r$. □

Lemma 5.1.5 *The optimal solution on set S is the same as the optimal solution on set V.*

Proof: This is simply true by the filtering step, since every point in the optimal solution belongs to S. □

Lemma 5.1.6 *Consider the guess $R = \frac{d^*}{2}$, where d^* is the maximum distance between any two points in the same optimal cluster, our algorithm covers no less points than the optimal solution on set S.*

Proof: We will prove a stronger statement. We will show that our algorithm covers no less points than the following optimal solution OPT on set S: it has at most k clusters, and the maximum distance between any two points in the same optimal cluster is at most d^*, but there is no requirement on the number of points per cluster. Let $O_1, O_2, \ldots, O_k$ denote the set of optimal clusters in OPT. We claim that:

$$|E_1 \cup \ldots \cup E_k| \geq |O_1 \cup \ldots \cup O_k| \tag{5.1}$$

The proof is by induction on k. The claim is true for $k = 1$, since $|E_1| \geq |G_1| \geq |O_1|$. Assume that $k > 1$. Clearly,

$$\bigcup_{i=1}^{k} (E_1 \cap O_i) \subseteq E_1.$$

Assume that G_1 intersects one of the disks $O_1, \ldots, O_k$ (say, O_1). Then $O_1 \subseteq E_1$ and the following inequality is satisfied.

$$|E_1| \geq |O_1| + \sum_{i=2}^{k} |E_1 \cap O_i|. \tag{5.2}$$

The above inequality is satisfied even if G_1 does not intersect any of the disks $O_1, \ldots, O_k$, since then

$$\bigcup_{i=1}^{k} (E_1 \cap O_i) \cup G_1 \subseteq E_1.$$

Now since $|G_1| \geq \max\{|O_1|, |O_2|, \ldots, |O_k|\} \geq |O_1|$, we have

$$|E_1| \geq |G_1| + \sum_{i=1}^{k} |E_1 \cap O_i| \geq |O_1| + \sum_{i=2}^{k} |E_1 \cap O_i|.$$

In either case, inequality (5.2) is satisfied.

Now consider the $(k-1)$-center problem on the set $S - E_1$. On this set, our algorithm could have picked the sets $E_2, E_3, \ldots, E_k$. Also, for $S - E_1$, it is clear that $O_2 - E_1, O_3 - E_1, \ldots, O_k - E_1$ is a solution, although it is not necessarily an optimal one. By induction, we know that

$$|E_2 \cup \ldots \cup E_k| \geq |\bigcup_{i=2}^{k}(O_i - E_1)| \tag{5.3}$$

Combining inequalities (5.2) and (5.3) proves (5.1). □

Combining the above three lemmas we have the following theorem.

Theorem 5.1.7 *Our algorithm gives a 4-approximation for the* (k, r, ϵ)-CENTER *problem.*

5.2 Cellular Clustering

As mentioned in the beginning of the chapter, a second approach is to publish the radius of each cluster in addition to its center and the number of points within it. In this case, for each point within a cluster, the radius of the cluster gives an upper bound on the distortion error introduced. The CELLULAR CLUSTERING problem aims to minimize the overall distortion error, *i.e.*, it partitions the points in a metric space into cells, each having a cell center, such that the sum, over all cells, of the products of the number of points in the cell and the radius of the cell is minimized. We even allow each potential cluster center to have a facility (setup) cost $f(v)$ associated with opening a cluster centered at it. This will later allow us to solve the problem in the case when each cluster is required to have at least r points within it.

Definition 5.2 *A cluster consists of a center along with a set of points assigned to it. The* radius *of the cluster is the maximum distance between a point assigned to the cluster and the cluster center. To open a cluster with cluster center* v *and radius* r *incurs a* facility cost $f(v)$*. In addition, each open cluster incurs a* service cost *equal to the number of points in the cluster times the cluster radius. The sum of these two costs is called the* cellular cost *of the cluster. The* CELLULAR CLUSTERING *problem is to partition* n *points in a metric space into clusters with the minimum total cellular cost.*

The CELLULAR CLUSTERING problem is NP-complete via reduction from dominating set. We present a primal-dual algorithm for the CELLULAR CLUSTERING problem that achieves an approximation factor of 4.

Let $c = (v_c, d_c)$ denote a cluster c whose cluster center is the node v_c and whose radius is d_c. By definition, the setup cost $f(c)$ for a cluster $c = (v_c, d_c)$ depends only on its center v_c; thus $f(c) = f(v_c)$. For each possible choice of cluster center and radius $c = (v_c, d_c)$, define a variable y_c, a 0/1 indicator of whether or not the cluster c is open. There are $O(n^2)$ such variables. For a cluster $c = (v_c, d_c)$, any point p_i within a distance of d_c of its center v_c is said to be a *potential member* of the cluster c. For all potential members p_i of a cluster c, let x_{ic} be a 0/1 indicator of whether or not point p_i joins cluster c. Note that the pair (i, c) uniquely identifies an edge between p_i and the center of cluster c. We relax the integer program formulation to get the following linear program:

$$
\begin{array}{lll}
\text{Minimize:} & \sum_c (\sum_i x_{ic} d_c + f_c y_c) & \\
\text{Subject to:} & \sum_c x_{ic} \geq 1 & \forall i \\
& x_{ic} \leq y_c & \forall i, c \\
& 0 \leq x_{ic} \leq 1 & \forall i, c \\
& 0 \leq y_c \leq 1 & \forall c
\end{array}
$$

And the dual program is:

$$
\begin{array}{lll}
\text{Maximize:} & \sum_i \alpha_i & \\
\text{Subject to:} & \sum_i \beta_{ic} \leq f_c & \forall c \\
& \alpha_i - \beta_{ic} \leq d_c & \forall i, c \\
& \alpha_i \geq 0 & \forall i \\
& \beta_{ic} \geq 0 & \forall i, c
\end{array}
$$

The above formulation is similar to the primal-dual formulation of facility location [JV99]. However, since the assignment of additional points to clusters increases the service cost incurred by existing members of the cluster, we need a different approach to assign points to clusters.

Algorithm 5 describes the details of the growth of dual variables and the assignment of points to clusters. We say an edge (i, c) is *tight* if $\alpha_i \geq d_c$. When an edge (i, c) becomes

tight, the corresponding cluster c becomes partially open and p_i contributes an amount of $(\alpha_i - d_c)$ to the fixed facility cost of $f(c)$. At any step of the algorithm, a point is labeled *unassigned*, *idle* or *dead*. Initially, all points are *unassigned*. As some cluster becomes tight, all *unassigned* or *idle* points having tight edges to it become *dead*. In addition, some of the *unassigned* points become *idle* as described in the algorithm.

Algorithm 5 A PRIMAL DUAL METHOD FOR CELLULAR CLUSTERING

repeat
  Grow the unfrozen dual variables α_i uniformly.
  if $\alpha_i \geq d_c$ for some cluster c and its potential member p_i, *i.e.*, edge (i, c) is tight, and c has not been shut down **then**
    Open the cluster c partially, and grow the dual variable β_{ic} at the same rate as α_i.
  end if
  if $\sum_i \beta_{ic} = f_c$ for some cluster c **then**
    Freeze all variables α_i for which the edge (i, c) is tight.
    All *unassigned* points with a tight edge to c are assigned to c. Call this set V_c^U.
    Let V_c^I be the set of all *idle* points that have a tight edge to c.
    Permanently shut down any cluster $c' \neq c$ for which a point p_i in $V_c^U \cup V_c^I$ has a tight edge (i, c'). Assign to c all *unassigned* points p_j with a tight edge to c'. Call this newly-assigned set of points V_c^{IU}.
    All points in V_c^{IU} are labeled *idle* and their dual variables are frozen.
    All points in V_c^U and V_c^I are labeled *dead*.
  end if
until All points become *dead* or *idle*.

We now show that the primal solution constructed has a cost of at most 4 times the value of the dual solution found using Algorithm 5. For this, we note the following properties:

(1) At any instant, the value of α_i for all *unassigned* points i is the same. Moreover, this value is no less than the value of α_j for any *dead* or *idle* point j.

(2) Once a point has a tight edge to a particular cluster c (*i.e.*, a cluster is partially open), all *unassigned* potential members of that cluster (*i.e.*points within a distance d_c of the cluster center v_c) have tight edges to it.

(3) When a cluster opens, all its *unassigned* potential members are assigned to it and become *dead*.

(4) When a point p_i becomes *dead*, all but one facility partially supported by p_i is shut down.

(5) When a cluster shuts down, all its *unassigned* potential members are assigned to some open cluster and become *idle*.

Property (1) follows from the definition of our algorithm. Property (2) follows from property (1) and the fact that the edge (i, c) becomes tight when the dual variable α_i equals d_c. Property (3) then follows from (2). Property (4) again follows from the definition of the the algorithm. Property (5) can be seen as follows: we shut down a cluster c only when one of its *unassigned* or *idle* members has a tight edge to the cluster c' currently being opened, and also has a tight edge to c. By property (2), all *unassigned* members of c have tight edges to c. Hence in Steps 10 and 11 of the algorithm, these members will be assigned to c' and become *idle*.

Lemma 5.2.1 *The service cost for each point, $\sum_c x_{ic} d_c$, is no more than $3\alpha_i$.*

Proof: Consider the cluster c to which point i is assigned. When cluster c opens, points in V_c^U and V_c^{IU} are assigned to c. We need to bound the radius of the cluster consisting of $V_c^U \cup V_c^{IU}$. By property (1), all points in V_c^U and V_c^{IU} have the same dual variable value, say α. Let p be the cluster center of c. Clearly, for a point $q \in V_c^U$, $d(q, p) \leq d_c \leq \alpha$. For a point $r \in V_c^{IU}$, let c' be its cluster that was shut down (in Step 10) when r was assigned to c. Let p' be the cluster center of c', and let $q' \in V_c^U$ be the point that was partially supporting c'. Clearly, $\alpha \geq d_{c'}$ since q' is partially supporting c'. Combined with the fact that r and q' are potential members of c', we get that $d(r, p) \leq d(r, p') + d(p', q') + d(q', p) \leq 2d_{c'} + d_c \leq 3\alpha$. Thus, the cluster made of V_c^U and V_c^{IU} has overall radius no more than $3\alpha = 3\alpha_i$. □

Lemma 5.2.2 *The cost of opening the clusters, $\sum_c y_c f_c$, is no more than $\sum_i \alpha_i$.*

Proof: A cluster c is opened when $\sum_i \beta_{ic}$ equals f_c. Thus, for each open cluster c, we need to find $V_c \subseteq V$, s.t. $\sum_i \beta_{ic}$ can be charged to $\sum_{i \in V_c} \alpha_i$. To avoid charging any point i more than once, we need to make sure that the V_c's are disjoint. We begin by noting that when a cluster c opens, only points i with a tight edge to c can contribute to $\sum_i \beta_{ic}$. When a point

is labeled *dead*, by Property 4, all the clusters to which it has a tight edge are shut down and are not opened in future. This implies that clusters which are opened do not have tight edges to *dead* points. Thus, when a cluster c is opened, V_c^U and V_c^I are the only points which have tight edges to c. If we let $V_c = V_c^U \cup V_c^I$, then $\sum_{i \in V_c} \alpha_i \geq \sum_i \beta_{ic}$. Also, since the points in $V_c^U \cup V_c^I$ are labeled *dead* in this iteration, they will not appear in $V_{c'}^U \cup V_{c'}^I$ for any other cluster c'. □

We thus obtain the following theorem.

Theorem 5.2.3 *The primal-dual method in Algorithm 5 produces a 4-approximation solution to the* CELLULAR CLUSTERING *problem.*

5.2.1 r-Cellular Clustering

We now extend the above primal-dual algorithm to get an approximation algorithm for the r-CELLULAR CLUSTERING problem which has the additional constraint that each cluster is required to have at least r members. The notation (r, C) is used to denote a solution having a total cost of C, and having at least r members in each cluster.

Comparison with prior clustering work. Since our algorithm can be viewed as an extension of facility location, we briefly discuss related results. The facility location (and k-median) problems have been studied with the minimum cluster size constraint [KM00], as well as in the context of leaving an ϵ fraction of the points unclustered [CKMN01]. Let OPT_r be the optimal facility location cost with minimum cluster size r. If as stated before (r, C) denotes a solution with minimum cluster size r and solution cost C, bi-criteria approximation for the facility location problem of $(r/2, 5.184OPT_r)$ was achieved independently by Guha, Meyerson and Munagala [GMM00] and by Karger and Minkoff [KM00]. It is not known whether it is possible to achieve a one-sided approximation on facility location cost alone. In contrast, for the r-CELLULAR CLUSTERING problem, we provide an one-sided approximation algorithm, specifically we obtain a $(r, 80OPT_r)$ solution, where OPT_r is the cost of the optimal solution with cluster size at least r,

To achieve this, we first study a *sharing* variant of this problem, where a point is allowed to belong to multiple clusters, thus making it easier to satisfy the minimum cluster size constraint. Interestingly, allowing sharing changes the value of the optimal solution by at

most a constant factor. We note that this observation does not hold for facility location, where a shared solution might be arbitrarily better than an unshared one. The algorithm consists of three main steps:

1. Augmenting with Setup Costs. Given an instance of r-CELLULAR CLUSTERING, we first construct an instance of CELLULAR CLUSTERING as follows: augment the cluster cost f_c of a cluster c by $r \times d_c$. In addition, if a cluster $c = (v_c, d_c)$ has fewer than r points within distance d_c of its center v_c, this cluster is eliminated from the instance. If the original r-CELLULAR CLUSTERING instance has an optimal solution with cost OPT_r, it is not hard to see that the same solution works for the CELLULAR CLUSTERING instance constructed above with a total cost of at most $2OPT_r$. We invoke the 4-approximation algorithm for CELLULAR CLUSTERING on this new instance to find a solution with cost at most $8OPT_r$.

2. Sharing Points between Clusters. We now describe the notion of a *shared* solution for r-CELLULAR CLUSTERING. In a shared solution, points are allowed to be assigned to multiple clusters, as long as they pay the service cost for each cluster they are assigned to. A shared solution is *feasible* if all clusters have at least r (potentially shared) members. We modify the solution obtained above to get a feasible shared solution for r-CELLULAR CLUSTERING as follows: for each open cluster c with center P, assign the r closest neighbors of P to c as well, regardless of where they are initially assigned. The extra service cost of at most $r \times d_c$ for these r points can be accounted for by the extra facility cost of $r \times d_c$ being paid by the open cluster c in the CELLULAR CLUSTERING solution. Thus, we have obtained an $(r, 8OPT_r)$ shared solution for the r-CELLULAR CLUSTERING instance.

3. Making the Clusters Disjoint. Finally we show how to convert a shared solution to a valid solution where each point is assigned to only one cluster, with only a constant blowup in cost. We note that for the corresponding facility location problem, it is not feasible to do this "unsharing" without a large blowup in cost in the worst case.

Initially, all points are labeled *unassigned*. We consider the clusters in order of increasing cluster radius d_c. If a cluster c has at least r *unassigned* members, then it is opened and all its *unassigned* members are assigned to c and labeled *assigned*. We stop this process when all the remaining clusters have fewer than r *unassigned* members each. The remaining clusters are called *leftover* clusters. We temporarily assign each of the *unassigned* points arbitrarily to one of the leftover clusters it belongs to. Since each cluster had

at least r members in the shared solution, each leftover cluster c' must have a member in the shared solution, which is now *assigned* to an open cluster o, s.t. $d_{c'} \geq d_o$. We thus have the situation illustrated in Figure 5.5.

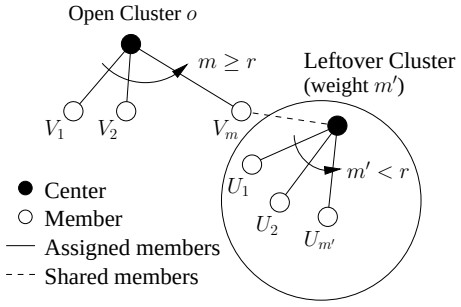


Figure 5.5: Structures of open and leftover clusters

The points are organized in a forest structure, where each tree has two "levels". We can regroup points into clusters, on a per tree basis. It is obvious that each tree has at least r points, since it contains at least one open cluster o. We further simplify the structure into a true two-level structure as in Figure 5.5, by collapsing each leftover cluster into a single node with weight equal to the number of points temporarily assigned to it. Nodes in the first level of the tree have weight 1. We apply the following greedy grouping procedure: first consider only the nodes at the second level of the tree and collect nodes until the total weight exceeds r for the first time. We group these nodes (belonging to leftover clusters) into a cluster, and repeat the process. Notice that since we did not touch the first-level nodes, the total weight of remaining nodes in the tree is at least r. If the total weight of remaining nodes in the second level, W_s, is less than r, then we extend the grouping into the first level nodes. Let m denote the total weight of nodes in the first level. If $W_s + m \geq 2r$, then we group the nodes in the second level with $r - W_s$ first level nodes together into a cluster; the remaining nodes in the first level form a cluster. Otherwise, all the remaining nodes (both the first and second level) are grouped into a cluster. If we break up the tree using the procedure above, each resulting cluster has size at least r.

Lemma 5.2.4 *For a cluster that contains any second-level nodes, the total number of points in the cluster is no more than* $2r - 1$.

Proof: Since a single second-level node has weight less than r, a cluster containing only second-level nodes has at most $2r - 1$ members. If the cluster contains both the first and second-level nodes, then we must have reached the case where the total weight of remaining nodes in the second level is less than r. In that case, by definition, the cluster formed containing these second-level nodes has size either r or less than $2r - 1$. □

There could be a cluster that only contains the first level nodes, and its entire cost (both the service and cluster cost) can be accounted for by its cost in the original $(r, 8OPT_r)$ shared solution. We now bound the cost of clusters containing the second-level nodes.

Lemma 5.2.5 *For each cluster c formed that contains second level nodes, there exists a leftover cluster c' unique to c, such that the following holds: let p be the center of c', if we center the cluster c at p, then the radius of cluster c,* $\text{radius}(c) \leq 5d_{c'}$.

Proof: Among all the leftover clusters that contributed to c, let c' be the one with the maximum radius. By definition, all nodes assigned to a leftover cluster get assigned to a single cluster, guaranteeing the uniqueness of c'. Let d_o be the radius of the open cluster at level 1 of this tree. Consider a point $q \in c$. If q is a first-level node, then $d(q, p) \leq 2d_o + d_{c'} \leq 3d_{c'}$. If q is a second-level node, then let c'' be the leftover cluster that q was assigned to, then $d(q, p) \leq 2d_{c''} + 2d_o + d_{c'} \leq 5d_{c'}$. □

The above lemma implies that by choosing p as the cluster center, the service cost of each point in c is no more than $5d_{c'}$ and the total facility cost incurred within our solution is no more than that of the shared solution. Together with Lemma 5.2.4, we conclude that the service cost of points in c is no more than $10r \times d_{c'}$. Notice that in the shared solution, points in cluster c' are paying a total service cost of at least $r \times d_{c'}$. We thus have the following theorem.

Theorem 5.2.6 *The above procedure produces a solution with minimum cluster size r and total cost no more than $80OPT_r$, i.e., a $(r, 80OPT_r)$ solution, where OPT_r is the value of the optimal solution with a minimum cluster size of r.*

We note that the above algorithm and analysis can be combined with the technique developed in [CKMN01] to give an constant approximation to the (r, ϵ)-CELLULAR CLUSTERING problem. The above algorithm can also be adapted to provide a constant-factor

approximation for the problem where the diameter of any cluster is not allowed to exceed a certain pre-specified threshold.

5.3 Summary and Future Work

Publishing data about individuals without revealing sensitive information is an important problem. The notion of privacy called k-Anonymity has attracted a lot of research attention recently. In a k-anonymized database, values of quasi-identifying attributes are suppressed or generalized so that for each record there are at least $k-1$ records in the modified table that have exactly the same values for the quasi-identifiers. However, the performance of the best known approximation algorithms for k-Anonymity depends linearly on the anonymity parameter k. In this chapter, we introduced clustering as a technique to anonymize quasi-identifiers before publishing them. We studied r-GATHER as well as a newly introduced clustering metric called r-CELLULAR CLUSTERING and provided the first constant-factor approximation algorithms for publishing an anonymized database table. Moreover, we generalized these algorithms to allow an ϵ fraction of points to remain unclustered. Defining the right metric space over the quasi-identifying attributes of records and improving the approximation ratios are interesting avenues for future work.

Chapter 6

Conclusions

We addressed the problem of *statistical disclosure control* – revealing aggregate statistics about a population while preserving the privacy of individuals. We presented models and algorithms for protecting the privacy of individuals in statistical databases while allowing users to mine useful trends and patterns. Our focus was on two frameworks – interactive and non-interactive – for protecting privacy in such databases. We studied methods under both these frameworks as each method is useful in different contexts.

In the first part of the thesis, we considered the interactive framework, in which the user (researcher) queries the database through a privacy mechanism, which may deny the query or alter the answer in order to ensure privacy. For the online query auditing problem, we uncovered the fundamental issue that query denials leak information and introduced the simulatable auditing model to overcome this problem. We also described a probabilistic notion of (partial) compromise, in order to overcome the known limitations of the existing privacy definition. We then presented simulatable auditing algorithms under both these definitions. The second problem we considered is output perturbation, in which the database administrator computes the exact answer to the query and then outputs a perturbed version as the response to the query. Inspired by the desire to enable individuals to retain control over their information, we provided a fault-tolerant distributed implementation of output perturbation schemes, thereby eliminating the need for a trusted database administrator.

In the second part of the thesis, we focused on the non-interactive framework and considered two anonymization methods for publishing data. We presented approximation algorithms for anonymizing databases under the k-Anonymity model. Then we proposed a new method for anonymizing data records, where the data records are clustered and then cluster centers are published, and provided approximation algorithms for the same.

At the end of each chapter, we outlined open problems related to the work described in that chapter. It would be interesting to address privacy concerns arising in contexts other than statistical databases. For example, consider the privacy issues associated with the availability of massive amounts of user-generated data such as the logs maintained by internet search engines. When a person searches for personally identifiable information (such as her home address) followed by a sensitive query, the corresponding logs have the same session information and hence one may be able to identify the person who issued the sensitive query. The challenge is to design a scheme that prevents learning private information through association of the log entries, while allowing useful data mining. Another direction for further research is to develop a unified privacy framework and provide a taxonomy of the different privacy models and techniques with respect to measures such as privacy and utility. This would involve comparing different definitions of privacy and also understanding the trade-off between privacy and utility. Developing such a framework may play a crucial role in fully understanding the relative merits of different privacy techniques.

Bibliography

[AA01] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.

[ABG+04] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, N. Mishra, R. Motwani, U. Srivastava, D. Thomas, J. Widom, and Y. Xu. Vision paper: Enabling privacy for the paranoids. In *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 708–719, 2004.

[AFK+05a] G. Aggarwal, T. Fèder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory*, pages 246–258, 2005.

[AFK+05b] G. Aggarwal, T. Fèder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-Anonymity. *Journal of Privacy Technology*, 2005. Paper number: 20051120001.

[AFK+06] G. Aggarwal, T. Fèder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 153–162, 2006.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k^{th}-ranked element. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 44–55, 2004.

[An96] M. An. Log-concave probability distributions: Theory and statistical testing. Technical Report 9611002, Economics Working Paper Archive at WUSTL, 1996. Available at http://ideas.repec.org/p/wpa/wuwpga/9611002.html.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450, 2000.

[AST05] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 251–262, 2005.

[AW89] N. Adam and J. Wortmann. Security control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[Bar02] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 345–355, 2002.

[BDMN05] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 128–138, 2005.

[Bec80] L. Beck. A security machanism for statistical database. *ACM Transactions on Database Systems*, 5(3):316–338, 1980.

[BGR96] M. Bellare, J. Garay, and T. Rabin. Distributed pseudo-random bit generators – a new way to speed-up shared coin tossing. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, pages 191–200, 1996.

[BIKP93] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15(3):385–415, 1993.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19, 1988.

[CD89] B. Chor and C. Dwork. Randomization in Byzantine agreement. *Advances in Computing Research*, 5:443–497, 1989.

[CDM+05] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 363–385, 2005.

[CGH+85] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem of t-resilient functions. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 429–442, 1985.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 383–395, 1985.

[Chi86] F. Chin. Security problems on inference control for SUM, MAX, and MIN queries. *Journal of the ACM*, 33(3):451–464, 1986.

[CKMN01] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.

[CKS05] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, 2005.

[CO81a] F. Chin and G. Ozsoyoglu. Auditing for secure statistical databases. In *Proceedings of the ACM '81 conference*, pages 53–59, 1981.

[CO81b] F. Chin and G. Ozsoyoglu. Statistical database design. *ACM Transactions on Database Systems*, 6(1):113–139, 1981.

[Cor88] G.P. Cornuejols. General factors of graphs. *Journal of Combinatorial Theory*, 45:185–198, 1988.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology: Proceedings of Crypto*, pages 13–25, 1998.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistisk Tidskrift*, 15:429–444, 1977.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[DFK+06] I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Proceedings of the 3rd Theory of Cryptography Conference*, pages 285–304, 2006.

[DJL79] D. Dobkin, A. Jones, and R. Lipton. Secure databases: protection against user influence. *ACM Transactions on Database Systems*, 4(1):97–106, 1979.

[DKM+06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 486–503, 2006.

[DLN96] C. Dwork, J. Lotspiech, and M. Naor. Digital signets for protection of digital information. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 489–498, 1996.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, pages 265–284, 2006.

[DN03] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 202–210, 2003.

[DN04] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology: Proceedings of Crypto*, pages 528–544, 2004.

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 211–222, 2003.

[Fei99] U. Feige. Noncryptographic selection protocols. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 142–153, 1999.

[FL82] M. Fischer and N. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 4:183–186, 1982.

[FM88] P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 148–161, 1988.

[FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 1–19, 2004.

[Gen96] R. Gennaro. *Theory and practice of verifiable secret sharing*. PhD thesis, MIT, 1996.

[GJ79] M. Garey and D. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[GM82] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 365–377, 1982.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM98] J. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.

[GMM00] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2000.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[Gol04] O. Goldreich. *Foundations of Cryptography - Basic Applications*, volume 2. Cambridge University Press, 2004.

[GRS04] A. Gabizon, R. Raz, and R. Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 394–403, 2004.

[HS85] D. Hochbaum and D. Shmoys. A best possible approximation algorithm for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[JV99] K. Jain and V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[Kan94] V. Kann. Maximum bounded H-matching is MAX SNP-complete. *Information Processing Letters*, 49:309–318, 1994.

[KM00] D. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[KMN05] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 118–127, 2005.

[KPR03] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 66(1):244–253, 2003.

[KS00] S. Khuller and Y. Sussmann. The capacitated K-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

[KU77] J. Kam and J. Ullman. A model of statistical databases and their security. *ACM Transactions on Database Systems*, 2(1):1–10, 1977.

[KZ03] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 92–101, 2003.

[LP02] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[LV03] 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*, pages 640–649, 2003.

[LWWJ02] Y. Li, L. Wang, X. Wang, and S. Jajodia. Auditing interval-based inference. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 553–567, 2002.

[MKGV06] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy beyond k-Anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, 2006.

[MS06] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 143–152, 2006.

[MW04] A. Meyerson and R. Williams. On the complexity of optimal k-Anonymity. In *Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 223–228, 2004.

[NMK+06] 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 Databases*, 2006.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[PR05] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 563–572, 2005.

[Pre95] A. Prekova. *Stochastic Programming*. Akadémiai Kiadó, Budapest and Kluwer, Dordrecht, 1995.

[Rab83] M. Rabin. Randomized Byzantine generals. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 403–409, 1983.

[Rei79] S. Reiss. Security in databases: A combinatorial study. *Journal of the ACM*, 26(1):45–57, 1979.

[Sam01] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[Sha02] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

[SS98] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, page 188, 1998.

[Swe00] L. Sweeney. Uniqueness of simple demographics in the U.S. population. Technical Report LIDAP-WP4, Laboratory for International Data Privacy, Carnegie Mellon University, Pittsburgh, PA, 2000.

[Swe02] L. Sweeney. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[TV00] L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[Yao82] A. Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[Yao86] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.