# ALGORITHMS AND ARCHITECTURES 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

Dilys Thomas

June 2007

# Abstract

The explosive progress in networking, storage, and processor technologies has resulted in an unprecedented volume of digital data. With this increase in digital data, concerns about privacy of personal information have emerged. The ease with which data can be collected, stored in databases and queried efficiently over the internet has worsened the privacy situation, and has raised numerous ethical and legal concerns. Privacy enforcement today is being handled primarily through legislation. We aim to provide technological solutions to achieve a tradeoff between data privacy and data utility. We focus on three problems in the area of database privacy in this thesis.

The first problem is that of data sanitization before publication. Publishing health and financial information for research purposes requires the data be anonymized so that the privacy of individuals in the database is protected. This anonymized information can be (1) used as is or (2) can be combined with another (anonymized) dataset that shares columns or rows with the original anonymized dataset. We explore both these sub-problems in this thesis. Another reason for sanitization is to give the data to an outsourced software developer for testing software applications without the outsourced developer learning information about its client. We briefly explain such a tool in this thesis.

The second part of the thesis studies auditing query logs for privacy. Given certain forbidden views of a database that must be kept confidential, a batch of SQL queries that were posed over this database, and a definition of suspiciousness, we study the problem to determine whether the batch of queries is suspicious with respect to the forbidden views.

The third part of the thesis deals with distributed architectures for data privacy.

The advent of databases as an outsourced service has resulted in privacy concerns on the part of the client storing data with third party database service providers. Previous approaches to enabling such a service have been based on data encryption, causing a large overhead in query processing. In this thesis we provide a distributed architecture for secure database services. We develop algorithms for distributing data and executing queries over this distributed data.

# Acknowledgments

First and foremost I would like to thank my advisor Rajeev Motwani for running one of the most wonderful research groups. His suggestion for papers and ideas during our group lunches and research meetings have fueled a lot of good research in our group. His insightful comments from experience have been helpful from time to time. His group has been the source of a lot of academic and some social activities to keep us lively all the time. They say that a great teacher inspires. Rajeev's enthusiasm to solve relevant and varied research problems have inspired me and all other members of his group to good research.

I would like to express my gratitude to other professors at Stanford. Hector Garcia-Molina for being a active participant of privacy research and for his feedback on presentation style and other practical issues. Jennifer Widom for running the Stream group and for her insights into research at the database group lunches. Dan Boneh for his mathematical puzzles, which provided me with good stimulating thinking many an afternoon.

I would like to thank my reading committee members: Rajeev Motwani, Dan Boneh, and John Mitchell and other members on my orals committee: Hector Garcia-Molina and Ashish Goel.

I would like to thank the various research groups I was a part of: *STREAM* run by Jennifer Widom and Rajeev Motwani, *RAIN* run by Rajeev Motwani, Ashish Goel and Amin Saberi, *PORTIA-PRIVACY* run by Rajeev Motwani, Hector Garcia-Molina, Dan Boneh and John Mitchell and *TRUST* run by John Mitchell, Dan Boneh, Rajeev Motwani and Hector Garcia-Molina.

I would like to thank my internship mentors and managers – I learnt a lot about different styles of research from them. I would like to especially thank Ramakrishnan Srikant, Rakesh Agrawal, Surajit Chaudhuri, Nicolas Bruno, Phillip Gibbons, Sachin Lodha, Anand Rajaraman and Srinivasan Sheshadri.

I would like to thank my professors at the Indian Institute of Technology, Bombay, esp. S. Sudarshan for providing me an excellent undergraduate education and getting me initiated into research. I would also like to thank my batchmates and friends from there.

I thank other students working with Rajeev: Krishnaram Kenthapadi, Gurmeet Manku, Gagan Aggarwal, Rina Panigrahy, Shubha Nabar, Ying Xu, Sergei Vassilvitskii, An Zhu, David Arthur, Aleksandra Korolova, Mayur Datar, Brian Babcock, Liadan Boyen and Aristides Gionis who have provided a wonderful environment to work.

I would like to thank Gaurav Bamania, Anuranjan Jha, Mayur Naik, Joseph Alex, Rajat Raina, Utkarsh Srivastava, Rajiv Agrawal, Omkar Deshpande, Pradeep Kumar, Rob, Jim Cybluski and Blake Blailey for being kind and considerate roommates.

It was a pleasure spending time with people in the Theory and Database and OR groups at Stanford: Zoltan Gyongyi, Prasanna Ganesan, Mayank Bawa, Qi Su, Mukund Sundararajan, Adam Barth, Aaron Bradley, Damon Mosk-Aoyama, Sriram Sankaranarayanan, Anupam Datta, Bobji Mungamuru, Shivnath Babu, Hamid Nazarzadeh, Arvind Arasu, David Menestrina and Arnab Roy are just a few to name.

I would like to thank my coauthors not mentioned above Renato Carmo, Prasenjit Das, A A Diwan, Tomas Feder, Vignesh Ganapathy, Keith Ito, Samir Khuller, Yoshiharu Kohayakawa, Eduardo Sany Laber, Nina Mishra, Itaru Nishizawa, Nikhil Patwardhan, Sharada Sundaram and Rohit Varma.

I would like to thank Kathi DiTommaso, Lynda Harris, Maggie Mcloughin, Wendy Cardamone, Claire Stager, Verna Wong, Indira Chaudhury, Meredith Hutchin, Jam Kiattinant and Peche Turner for taking care of all the important administrative matters during the PhD. I would like to thank Lilian Lao, Andy Kacsmar and Miles Davis for taking care of the machines.

I would like to thank the various outing clubs and groups at Stanford, the Catholic community here, SIA, Rains groups, IVGrad, DB movie and social committee for ensuring life outside work was something to look forward to. I would like to thank Joshua Easow and family and Jojy Michael and family for memorable times with them.

Above all, I thank my grandparents, parents, sister Dina, my cousins and friends for a wonderful childhood with memories of catching butterflies from fields, small fish from streams, spending time on the hills, studying and playing.

**To God**

# 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 that can be collected and analyzed. 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.

## 1.1 Sanitizing data for Privacy

The first problem is that of data sanitization before publication. There are two main reasons for data sanitization before publication. Publishing health and financial information for research purposes requires the data be anonymized so that the privacy of individuals in the database is protected. This anonymized information can be used as is or can be combined with another (anonymized) dataset for analysis. We explore both these scenarios in this thesis. Another reason for sanitization is to give the data to an out-sourced software developer for software development without the out-sourced data handler learning information about its client. We briefly explain such a tool in this thesis.

### 1.1.1 Privacy Preserving OLAP

Publishing health, financial, personal information requires the data be anonymized so that the privacy of individuals in the database is protected. This information can be combined with another (anonymized) dataset for analysis. We explore this under Privacy Preserving OLAP in Chapter 2 in this Thesis. We present techniques for privacy-preserving computation of multidimensional aggregates on data partitioned across multiple clients. Data from different clients is perturbed (randomized) in order to preserve privacy before it is integrated at the server. We develop formal notions of privacy obtained from data perturbation and show that our perturbation provides guarantees against privacy breaches.We develop and analyze algorithms for reconstructing counts of subcubes over perturbed data. We also evaluate the tradeoff between privacy guarantees and reconstruction accuracy and show the practicality of our approach.

### 1.1.2 Clustering for Anonymity

Publishing data for analysis from a table containing personal records, while maintaining individual privacy, is a problem of increasing importance today. The traditional approach of de-identifying records is to remove identifying fields such as social security number, name etc. However, recent research has shown that a large fraction of the US population can be identified using non-key attributes (called quasi-identifiers) such as date of birth, gender, and zip code [Swe00]. Sweeney [Swe02b] proposed the $k$-anonymity model for privacy where non-key attributes that leak information are suppressed or generalized so that, for every record in the modified table, there are at least $k-1$ other records having exactly the same values for quasi-identifiers. 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. This is the first set of algorithms for the anonymization problem where the performance is independent of the anonymity parameter $k$. We further observe that a few outlier points can significantly increase the cost of anonymization. Hence, we extend our algorithms to allow an $\epsilon$ 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.

### 1.1.3 Probabilistic Anonymity

In this age of globalization, organizations need to publish their micro-data owing to legal directives or share it with business associates in order to remain competitive. This puts personal privacy at risk. To surmount this risk, attributes that clearly identify individuals, such as `Name`, `Social Security Number`, `Driving License Number`, are generally removed or replaced by random values. But this may not be enough because such de-identified databases can sometimes be joined with other public databases on attributes such as `Gender`, `Date of Birth`, and `Zipcode` to re-identify individuals who were supposed to remain anonymous. In literature, such an identity-leaking attribute combination is called as a quasi-identifier. It is always critical to be able to recognize quasi-identifiers and to apply to them appropriate protective measures to mitigate the identity disclosure risk posed by join attacks.

In Chapter 4, we start out by providing the first formal characterization and a practical technique to identify quasi-identifiers. We show an interesting connection between whether a set of columns forms a quasi-identifier and the number of distinct values assumed by the combination of the columns. We then use this characterization to come up with a probabilistic notion of anonymity. Again we show an interesting connection between the number of distinct values taken by a combination of columns and the anonymity it can offer. This allows us to find an ideal amount of generalization or suppression to apply to different columns in order to achieve probabilistic anonymity. We work through many examples and show that our analysis can be used to make a published database conform to privacy acts like HIPAA. In order to achieve the probabilistic anonymity, we observe that one needs to solve multiple 1-dimensional $k$-anonymity problems. We propose many efficient and scalable algorithms for achieving 1-dimensional anonymity. Our algorithms are optimal in a sense that they minimally distort data and retain much of its utility.

### 1.1.4 A Tool for Data Privacy: Masketeer

Major countries like the U.S., Japan, Canada, Australia and EU have come up with strict data distribution laws which demand their organizations to implement proper data security measures that respect personal privacy and prohibit dissemination of raw data outside the country.

Since companies are not able to provide real data, they often resort to completely random data. It is obvious that such a data would offer complete privacy, but would have very low utility. This has serious implications for IT services companies since application development and testing environments rely on realistic test data to verify that the applications provide the functionality and reliability they were designed to deliver. It is always desirable that the test data is *similar* to, if not the same as, the production data. Hence, deploying proven tools that make de-identifying production data easy, meaningful and cost-effective is essential.

Data masking methods came into existence to permit the legitimate use of data and avoid misuse. In Chapter 5, we consider various such techniques to be able to come up with a comprehensive solution for data privacy requirements. We present the data masking product MASKETEER™ (developed at TCS) which implements these techniques for providing maximum privacy for data while maintaining good utility.

## 1.2 Auditing

The second part of the thesis deals with algorithms to audit query logs for privacy in Chapter 6.

We study the problem of auditing a batch of SQL queries: Given certain *forbidden views* of a database that must be kept confidential, a batch of SQL queries that were posed over this database, and a definition of *suspiciousness*, determine whether the batch of queries is suspicious with respect to the forbidden views. In this paper, we define two different notions of suspiciousness — *semantic suspiciousness*, corresponding to the definition introduced in [ABF$^+$04], where the problem was studied for a single SQL query in isolation and (2) a database instance-independent notion of *syntactic suspiciousness*. For a given database instance we provide a polynomial time algorithm for detecting if a batch of select-project-join queries is semantically suspicious. This algorithm requires actual execution of the queries against the database. Since syntactic suspiciousness of a query batch is independent of the underlying database instance, it may seem more desirable. However we show that it is in fact NP-hard to achieve even when we restrict ourselves to the class of conjunctive queries. We therefore weaken the notion of syntactic suspiciousness and present a polynomial time algorithm for auditing a batch of conjunctive SQL queries under this weaker definition. Finally, we provide a synthesis of recent research in the areas of query auditing and access control and illustrate the relationship between the notions of perfect privacy studied in [MS04, MG06], semantic and syntactic suspiciousness introduced here, as well as the notion of unconditional validity of a query introduced in database access control literature [Mot89, RS00, RS01, RSD99, RMSR04].

## 1.3 Distributed Architectures for Privacy

The final part of the thesis deals with distributed architectures to store private information at a database server. Data is distributed at multiple sites so that a hacker or insider having access to a single site is unable to compromise the private information stored in the database.

Recent trends towards database outsourcing, as well as concerns and laws governing data privacy, have led to great interest in enabling secure database services. Previous approaches to enabling such a service have been based on data encryption, causing a large overhead in query processing. We propose a new, distributed architecture that allows an organization to outsource its data management to *two* untrusted servers while preserving data privacy. We show how the presence of two servers enables efficient partitioning of data so that the contents at any one server are guaranteed not to breach data privacy. We show how to optimize and execute queries in this architecture, and discuss new challenges that emerge in designing the database schema.

# Part I

# Sanitizing Data for Privacy

# Chapter 2

# Privacy Preserving OLAP

The results in this chapter appear in [AST05].

## 2.1 Introduction

On-line analytical processing (OLAP) is a key technology employed in business-intelligence systems. The computation of multidimensional aggregates is the essence of on-line analytical processing. We present techniques for computing multidimensional count aggregates in a privacy-preserving way.

We consider a setting in which clients $C_1, C_2, \ldots C_n$ are connected to a server $S$. The server has a table $T(A_1, A_2, \ldots, A_m)$, where each column $A_i$ comes from a numeric domain. Each client $C_i$ contributes a row $r_i(a_{i_1}, a_{i_2}, \ldots, a_{i_m})$ to $T$. The server runs aggregate queries of the form

`select count(*) from` $T$
`where` $P_{j_1}$ `and` $P_{j_2}$ $\ldots$ `and` $P_{j_k}$.

Here $P_{j_i}$ is a range predicate of the form $a_{l_i} \leq A_{j_i} \leq a_{h_i}$, denoted as $A_{j_i}[a_{l_i}, a_{h_i}]$. We use count$(P_{j_1} \wedge P_{j_2} \ldots \wedge P_{j_k})$ to succinctly represent the above aggregate query.

We take the randomization approach to preserving privacy. The basic idea is that every client $C_i$ perturbs its row $r_i$ before sending it to the server $S$. The randomness used in perturbing the values ensures information-theoretic row-level privacy.

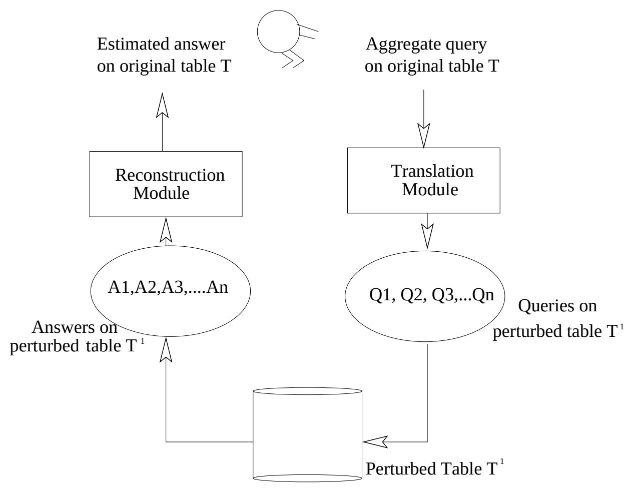

Estimated answer on original table T
Aggregate query on original table T
Reconstruction Module
Translation Module
A1,A2,A3,....An
Q1, Q2, Q3,...Qn
Answers on perturbed table T 1
Queries on perturbed table T 1
Perturbed Table T 1

Figure 2.1: *Privacy preserving computation of multidimensional count aggregates.*

Figure 2.1 gives the schematic of our approach. $S$ runs queries on the resultant perturbed table $T'$. The query meant for the original table $T$ is translated into a set of queries on the perturbed table $T^{'}$. The answers to these queries are then reconstructed to obtain the result to the original query with bounded error. We show that our techniques are safe against privacy breaches.

The perturbation algorithm is publicly known; the actual random numbers used in the perturbation, however, are hidden. To allow clients to operate independently, we use local perturbations so that the perturbed value of a data element depends only on its initial value and not on those of the other data elements. Different columns of a row are perturbed independently. We use *retention replacement* schemes where an element is decided to be retained with probability $p$ or replaced with an element selected from a probability distribution function (p.d.f.) on the domain of elements.

The proposed techniques can also be used for database tables in which some of the columns are categorical. They are also applicable in the settings in which the database tables are partitioned horizontally or vertically. The organization of the rest of the paper is as follows. We start off with a discussion of related work in Section 2.2. Section 2.3 formally defines the retention replacement perturbation. Section 2.4 presents the reconstruction algorithms. Section 2.5 presents the guarantees against privacy breaches offered by our techniques. In Section 2.6, we discuss how our techniques can be extended to categorical data. We also discuss some additional perturbation techniques and describe how our techniques can be used in data mining by showing how to build a decision tree classifier. Section 2.7 presents an empirical evaluation of our techniques. We conclude with a summary and directions for future work in Section 2.8.

## 2.2 Related Work

The techniques for preserving privacy while answering statistical queries developed in the statistical database literature can be classified into *query restriction*, *input perturbation* and *output perturbation* [AW89]. Both query restriction and output perturbation are applicable when the entire original unperturbed data is available in a single central repository, which is not true in our setting, where clients randomize their data before providing it to the server. Our scenario fits in the framework of input perturbation, where the goal is to create a version of the database that can be publicly released (e.g. census data), yet the individual rows should not be recoverable. Local perturbation for a single column has been studied in [War65]. However most previous work (e.g., [Jr.]) assume that during perturbation the entire database is available at a single site, while we require local perturbations at each client.

The use of local perturbation techniques to preserve privacy of individual rows while allowing the computation of data mining models at the aggregate level was proposed in [AS00]. They used an additive perturbation technique, in which a random perturbation is added to the original value of the row, where the perturbation is picked from another probability distribution function (e.g. Gaussian). They showed that it was possible to build accurate decision tree classification models on the perturbed data.

However, it is difficult to provide guarantees against privacy breaches when using additive perturbation. For instance, if we add a Gaussian random variable with a mean 0 and variance 20 to age, and for a specific row the randomized value happens to be $-60$, one can estimate with high confidence that the original value of age was (say) less than 20. Additive schemes are also restricted to numeric data. Finally, the algorithms in [AS00] reconstruct each column independently. Since OLAP requires queries over multiple columns, it is essential to be able to reconstruct them together.

The problem of privacy-preserving association-rule mining was studied in [EGS03, ESAG02, RH02]. The randomization schemes used in these works are similar to the retention replacement schemes we consider. However these studies are restricted to boolean data.

Formal definitions of privacy breaches were proposed in [EGS03], and an alternate approach to defining privacy guarantees was proposed in [CDM$^+$05]. We adapt the definitions from [EGS03] to allow more accurate reconstruction while still providing strong privacy guarantees. As our notion of privacy encompasses multiple correlated columns over vertically partitioned tables, it extends to privacy breaches (called disclosure risk) considering row linkage, studied in statistical disclosure control methods and [FT03].

There has been recent work [WJW, WWJ04] to specify authorization and control inferences for OLAP data cubes. However the model assumes that the data resides at a single server, unlike our problem, where private data is integrated from multiple clients.

Another related area is that of *secure multiparty computation* [GMW87, Yao86], that allows any function, whose inputs are shared between multiple clients to be evaluated, such that nothing other than the result is revealed. Since the general protocols are expensive, efficient protocols have been proposed for specific database and data mining operations, e.g. [AES03, CKL$^+$03, FNP04, HFH99, LP00]. However, these protocols are designed for a small number of clients.

## 2.3 Data Perturbation

A single record of the table is referred to as a *row*, while an attribute is referred to as a *column*. A single column from a single row is the granularity of perturbation and is referred to as a *data element*.

**Definition 2.1 Perturbation Algorithm:** *A* perturbation algorithm $\alpha$ *is a randomized algorithm that given a table $T$ creates a table $T'$ having the same number of* *rows and columns.*

We will denote the unperturbed table as $T$ and the perturbed table as $T^{'}$. The perturbation algorithm is public. However, the actual random numbers used by it are hidden.

Let $t_{ij}$ and $t^{'}_{ij}$ denote the value of the element in the $i^{th}$ row of the $j^{th}$ column in tables $T$ and $T^{'}$ respectively. The perturbation algorithm is said to be *local* if $t^{'}_{ij}$ depends only on $t_{ij}$, while it is said to be *global* if $t^{'}_{ij}$ depends on other elements in the $j^{th}$ column of $T$.

Let $D_j$ denote the domain of elements in the $j^{th}$ column of $T$. $D_j$ is said to be continuous for numeric columns, and discrete for categorical columns. For the class of perturbation algorithms we study, for every column being perturbed, we require the perturbation algorithm to select a fixed probability density function (p.d.f.) on the column's domain. For the $j^{th}$ column we call this p.d.f. the *replacing p.d.f.* on $D_j$. Both $D_j$ as well as the replacing p.d.f. on $D_j$ are public.

**Definition 2.2 Retention Replacement Perturbation:** Retention replacement *perturbation is a perturbation algorithm, where each element in column $j$ is retained with probability $p_j$, and with probability $(1-p_j)$ replaced with an element selected from the replacing p.d.f. on $D_j$. That is,*

$$t^{'}_{ij} = \begin{cases} t_{ij} \quad \textit{with probability } p_j \\ \textit{element from replacing p.d.f. on } D_j \textit{ with probability (1-}p_j\textit{).} \end{cases}$$

If column $j$ of the table can be revealed without perturbation we set $p_j = 1$.

Retention replacement perturbation, where the replacing p.d.f. is the uniform p.d.f. is called *uniform perturbation.* We assume that each column of the table $T^{'}$ has been perturbed independently using uniform perturbation. In Section 2.6.2, we show that uniform perturbation provides better privacy guarantees for rare events. Other alternatives and comparisons are also given in the same section.

## 2.4 Reconstruction

An aggregate function on the original table $T$, must be reconstructed by accessing the perturbed table $T^{'}$. The accuracy of the reconstruction algorithm is formalized below by the notion of approximate probabilistic reconstructability.

**Definition 2.3 Reconstructible Function:** *Given a* perturbation $\alpha$ *converting table $T$ to $T^{'}$, a numeric function $f$ on $T$ is said to be $(n, \epsilon, \delta)$ reconstructible by a function $f^{'}$, if $f^{'}$ can be evaluated on the perturbed table $T^{'}$ so that $|f^{'} - f| < max(\epsilon, \epsilon f)$ with probability greater than $(1 - \delta)$ whenever the table $T$ has more than $n$ rows. The probability is over the random choices made by $\alpha$.*

For boolean functions, $(n, \delta)$ reconstructability needs $f$ and $f^{'}$ to agree exactly with probability greater than $(1 - \delta)$.

Referring to Figure 2.1, to answer the aggregate query $count(P_1 \wedge P_2 \wedge \ldots P_k)$ on $k$ columns of the original table, $T$, a set of $2^k$ queries, $count(P_1 \wedge P_2 \wedge \ldots P_k)$, $count(\neg P_1 \wedge P_2 \wedge \ldots P_k)$, $count(P_1 \wedge \neg P_2 \wedge \ldots P_k)$, $count(\neg P_1 \wedge \neg P_2 \wedge \ldots P_k) \ldots count(\neg P_1 \wedge \neg P_2 \wedge \ldots \neg P_k)$ are generated. These queries are evaluated on the perturbed table $T^{'}$. The answers on $T^{'}$ are reconstructed into estimated answers for the same queries on $T$, which include the answer to the original query.

Without loss of generality, assume that the predicates are only over perturbed columns. We present reconstruction algorithms for numeric columns. These algorithms can be extended to categorical columns too as shown in Section 2.6.

### 2.4.1 Reconstructing Single Column Aggregates

Consider the uniform retention replacement perturbation with retention probability $p$ applied on a database with $n$ rows and a single column, $C$, with domain $[min,\ max]$. Consider the predicate $P = C[low,\ high]$. Given the perturbed table $T^{'}$, we show how to estimate an answer to the query $count(P)$ on $T$.

Let tables $T$, $T^{'}$ each have $n$ rows. Let $n_r = count(P)$ evaluated on table $T^{'}$, while $n_o = count(P)$ estimated for table $T$. Given $n_r$ we estimate $n_o$ as

$$n_o = \frac{1}{p}(n_r - n(1-p)b) \quad , \quad where \quad b = \frac{high - low}{max - min}.$$

The intuition is that out of the $n$ rows in table $T$, the expected number of rows that get perturbed is $n(1-p)$. For uniform perturbation, a $b$ fraction of these rows, i.e. $n(1-p)b$ rows, will be expected to lie within the $[low, high]$ range. The total number of rows observed in range $[low, high]$ in $T'$, $n_r$, can be seen as the sum of those rows that were decided to be perturbed into $[low, high]$ (from outside, or perturbed and retained within the interval) and those rows that were unperturbed in the original interval. Subtracting the $n(1-p)b$ perturbed rows from $n_r$, we get an estimate for the number of unperturbed rows, with values in $[low, high]$ in $T$. This is scaled up by $1/p$ to get the total number of original rows in $T$ in $[low, high]$, as only a $p$ fraction of rows were retained.

The fraction $f$ of rows originally in $[low, high]$ is therefore estimated as

$$f' = \frac{n_o}{n} = \frac{n_r}{pn} - \frac{(1-p)(high - low)}{p(max - min)}.$$

Not only is the above estimator a Maximum Likelihood Estimator (MLE) as shown in Section 2.4.2, it reconstructs an approximate answer with high probability.

**Theorem 2.4.1** *Let the fraction of rows in $[low, high]$ in the original table $f$ be estimated by $f'$, then $f'$ is a $(n, \epsilon, \delta)$ estimator for $f$ if $n \geq 4\log(\frac{2}{\delta})(p\epsilon)^{-2}$.*

**Proof:** Let $Y_i$ be the indicator variable for the event that the $i^{th}$ row $(1 \leq i \leq n)$ is perturbed and the perturbed value falls within $[low, high]$. $Y_i$ are i.i.d. with $Pr[Y_i = 1] = (1-p)b = q$ (say), $Pr[Y_i = 0] = 1 - q$. Let $X_i$ be the indicator variable for the event that the $i^{th}$ row is not perturbed and it falls within $[low, high]$. Once again $X_i$ are i.i.d. with $Pr[X_i = 1] = pf = r$ (say), $Pr[X_i = 0] = 1 - r$. Let $Z_i$ be the indicator variable for the event that the $i^{th}$ randomized row falls in $[low, high]$. We have $Z_i = X_i + Y_i$, and $Pr[Z_i = 1] = q + r = t$ (say), and $Pr[Z_i = 0] = 1 - t$. Let $Z = \sum_{i=1}^{n} Z_i = n_r$, the number of randomized values in range $[low, high]$. Since $Z_i$'s are independent Bernoulli random variables, $0 \leq t \leq 1$ and $n \geq 4\log(\frac{2}{\delta})(p\epsilon)^{-2} \times t$, applying Chernoff bounds[Che52] we get

$$Pr[|Z - nt| > nt\theta] < 2e^{\frac{-nt\theta^2}{4}} \leq \delta \quad for \;\; \theta = \frac{p\epsilon}{t}$$

Thus with probability $> 1-\delta$, we have $-np\epsilon < Z - nt = n_r - n(pf + (1-p)b) < np\epsilon$, which implies

$$f - \epsilon < \frac{n_r}{pn} - \frac{(1-p)(high - low)}{p(max - min)} < f + \epsilon$$

Hence $|f - f^{'}| < \epsilon$ with probability $> 1-\delta$.

$\square$

We now formalize the above reconstruction procedure. This formalization provides the basis for the reconstruction of multiple columns in Section 2.4.2.

Let vector $y = [y_0, y_1] = [count(\neg P), count(P)]$ be the answers on table $T^{'}$, and let vector $x = [x_0, x_1] = [count(\neg P), count(P)]$ denote the estimates for table $T$. Let $b$ be defined as before and $a = 1 - b$. As only table $T^{'}$ is available, $x$ is estimated using the constraint $xA = y$, which gives the estimator $x = yA^{-1}$. Here $A$ is the following transition matrix

$$\begin{bmatrix} (1-p)a + p & (1-p)b \\ (1-p)a & (1-p)b + p \end{bmatrix}.$$

The element in the first row and first column of $A$, $a_{00} = (1-p)a + p$ is the probability that an element originally satisfying $\neg P$ in $T$ after perturbation satisfies $\neg P$ in $T^{'}$. This probability was calculated as the sum of the probabilities of two disjoint events. The first being that the element is retained, which occurs with probability $p$. The second being that the element is perturbed and after perturbation satisfies $\neg P$, which together has probability $(1-p)a$. The element $a_{01}$ is the probability that an element satisfying $\neg P$ in $T$ after perturbation satisfies $P$ in $T^{'}$. The element $a_{10}$ is the probability that an element satisfying $P$ in $T$ after perturbation satisfies $\neg P$ in $T^{'}$. The element $a_{11}$ is the probability that an element satisfying $P$ in $T$ after perturbation satisfies $P$ in $T^{'}$. Their values were similarly derived.

If $y = [n - n_r, n_r]$ and $x = [n - n_o, n_o]$, the solution to the equation below gives

| Query | Estimated on $T$ | Evaluated on $T'$ |
|---|---|---|
| $count(\neg P_1 \wedge \neg P_2)$ | $x_0$ | $y_0$ |
| $count(\neg P_1 \wedge P_2)$ | $x_1$ | $y_1$ |
| $count(P_1 \wedge \neg P_2)$ | $x_2$ | $y_2$ |
| $count(P_1 \wedge P_2)$ | $x_3$ | $y_3$ |

Figure 2.2: *Answering query* $count(P_1 \wedge P_2)$

the same estimator as derived earlier:

$$\begin{bmatrix} n - n_o & n_o \end{bmatrix} \begin{bmatrix} (1-p)a + p & (1-p)b \\ (1-p)a & (1-p)b + p \end{bmatrix} = \begin{bmatrix} n - n_r & n_r \end{bmatrix}.$$

## 2.4.2 Reconstructing Multiple Column Aggregates

Assume now that the uniform retention replacement perturbation, with retention probability $p$, has been applied to each of $k$ columns of a table, $T$. Consider the aggregate query $count(P_1 \wedge P_2 \wedge ...P_k)$ on table $T$. In practice $k$ is small.

We create a $k \times 2$ matrix, $R$, with $k$ rows and 2 columns, having 1 row for each query column. $R_{i,1}$ gives the probability that a number randomly selected from the replacing p.d.f. for column $i$ will satisfy predicate $P_i$, while $R_{i,0}$ is the probability of the complementary event, that a number selected from the replacing p.d.f. will satisfy $\neg P_i$.

Take for instance the query, Q=count(age[30-45] $\wedge$ salary[50k-120k] $\wedge$ house-rent[700-1400]) with the domains for age, salary and house-rent being [0-100], [25k-200k], [500-2500]. Then $R$ will be $[[0.85, 0.15], [0.6, 0.4], [0.65, 0.35]]$, since the first column being age[30-45] implies $R_{1,1} = (45 - 30)/(100 - 0) = 0.15$, while $R_{1,0} = 1 - 0.15 = 0.85$, etc.

As stated earlier, to answer the query $count(P_1 \wedge P_2 \dots P_k)$, we ask $2^k$ aggregate queries on the perturbed table, $T'$. The $2^k$ answers on perturbed table $T'$ are converted into estimated answers to these $2^k$ aggregate queries on the original table T, which includes the estimated answer to the original query.

Let $y$ be a row vector of size $2^k$ that has the answers to the above queries on perturbed table $T'$, and let $x$ be a row vector of size $2^k$ that has the reconstructed estimated answers to the queries on original table $T$. We order the answers to the $2^k$ queries in vectors $x$, $y$ using the bit representation of the vector index as shown in Figure 2.2. Let $Q(r,1)$ denote the predicate($P_r$) on the $r^{th}$ column of query $Q$, and $Q(r,0)$ its negation ($\neg P_r$). Let $bit(i,r)$ denote the $r^{th}$ bit from the left in the binary representation of the number $i$ using k bits. Then,
$x_i = count(\wedge_{r=1}^{k} Q(r, bit(i,r)))$ in $T$, for $0 \leq i \leq 2^k - 1$;
$y_i = count(\wedge_{r=1}^{k} Q(r, bit(i,r)))$ in $T'$, for $0 \leq i \leq 2^k - 1$.

For example, for the query *count*(age[30-45] $\wedge$ salary[50k-120k] $\wedge$ house-rent[700-1400]), $y[6_{10}] = y[110_2] =$ *count*(age[30-45] $\wedge$ salary[50k-120k] $\wedge$ $\neg$ house-rent[$700 - 1400$])

By a single scan through the perturbed table $T'$ vector $y$ can be calculated. Vector $x$ is reconstructed from vector $y$ using the matrix inversion technique or the iterative Bayesian technique described below. The data analyst may either be interested only in the component $x_{2^k-1}$, which is the answer to the $count(\wedge_{r=1}^{k} P_r)$ query on $T$, or she may be interested in the entire vector $x$.

### Matrix Inversion technique

If $p_r$ is the retention probability for the $r^{th}$ column, we calculate vector $x$ from vector $y$ as $x = yA^{-1}$. The transition matrix, $A$, with $2^k$ rows and $2^k$ columns, can be calculated as the tensor product [HK71] of matrices

$$A = A_1 \otimes A_2 \otimes A_3 .... \otimes A_k$$

where the matrix $A_r$, for $1 \leq r \leq k$ is the transition matrix for column $r$ (see Section 2.4.1).

$$A_r = \begin{bmatrix} (1-p_r)a_r + p_r & (1-p_r)b_r \\ (1-p_r)a_r & (1-p_r)b_r + p_r \end{bmatrix}$$

where $b_r = R_{r,1}$ and $a_r = R_{r,0} = 1 - R_{r,1}$.

The entries of the tensor product matrix, $A$, can be explicitly calculated to be $a_{ij} = \prod_{r=1}^{k}((1-p_r) \times R_{r,bit(j,r)} + p_r \times \delta_{(bit(i,r),bit(j,r))})$, $\forall 0 \leq i < 2^k, 0 \leq j < 2^k$ where $\delta_{(c,d)} = 1$ if $c = d$, and $0$ if $c \neq d$, for $c, d \in \{0, 1\}$.

We split the space of possible evaluations of a row into $2^k$ states, according to which of the $2^k$ mutually exclusive predicate combinations the row satisfies. We say a row is said to belong to state $i$ if it satisfies the predicate $\wedge_{r=1}^{k} Q_(r, bit(i,r))$. For example, from Figure 2.2, a row in state 0 satisfies $\neg P_1 \wedge \neg P_2$ while a row in state 1 satisfies $\neg P_1 \wedge P_2$ etc.

The entry $a_{ij}$ of matrix $A$ above represents the probability that a row belonging to state $i$ in $T$, after perturbation belongs to state $j$ in $T'$. As each column was independently perturbed the probability of transition from state $i$ to state $j$ is the product of the probabilities for the transitions on all columns. The contribution from the $r^{th}$ column to the transition probability is the sum of $(1-p_r) \times R_{r,bit(j,r)}$, if the element was decided to be perturbed, and $p_r \times \delta_{(bit(i,r),bit(j,r))}$, if the element was decided to be retained. The term $\delta_{(bit(i,r),bit(j,r))}$ ensures that the retention probability $p_r$ adds up only if the source and destination predicates on the $r^{th}$ column are the same for states $i$ and $j$. Thus the probability of transition from state $i$ to state $j$ on the $r^{th}$ column is $(1-p_r) \times R_{r,bit(j,r)} + p_r \times \delta_{(bit(i,r),bit(j,r))}$. The product of this probability over all columns gives the probability of transition from state $i$ to state $j$, $a_{ij}$.

**Theorem 2.4.2** *The vector $x$ calculated as $A^{-1}y$ is the maximum likelihood estimator (MLE) of the relaxed a priori distribution ( $\sum_i x_i = n$ and $0 \leq x_i \leq n$ are the exact constraints, the relaxed constraint only ensures $\sum_i x_i = n$) on the states that generated the perturbed table.*

**Proof:** Let $V = (v^1, v^2, v^3, \ldots .v^n)$ be the observed state values for the $n$ perturbed rows in $T'$. Let $t = 2^k - 1$. Note that $v^i \in \{0, 1, 2, \ldots., t\}$ for all $i \in [1..n]$. If $L(x)$ is the likelihood of the observations, $V$, given a probability distribution on the states, $x$, in the original data, then $L(x) = Pr(V|x) = \prod_{i=1}^{n} Pr(v^i|x) = \prod_{i=1}^{n}(\frac{1}{n}\sum_{j=0}^{t} a_{jv^i}x_j)$ $= \prod_{i=0}^{t}(\frac{1}{n}\sum_{j=0}^{t} a_{ji}x_j)^{y_i}$ (reordering according to the values of $v^i$.) Maximizing $L(x)$

is equivalent to maximizing $\log(L(x))$.

$$\max \log(L(x)) = \sum_{i=0}^{t}(y_i \times \log(\frac{1}{n}\sum_{j=0}^{t} x_j a_{ji}))$$

subject to the constraint $\sum_{j=0}^{t} x_j = n$.
This is equivalent to

$$\max_x \min_\lambda l(x,\lambda) = \sum_{i=0}^{t}(y_i \log(\frac{1}{n}\sum_{j=0}^{t} x_j a_{ji})) - \lambda(\sum_{j=0}^{t} x_j - n)$$

where $\lambda$ is the Lagrangian multiplier.

If $\sum_{j=0}^{t} x_j - n > 0$ then setting $\lambda$ to arbitrarily large positive value, you can minimize the term $-\lambda(\sum_{j=0}^{t} x_j - n)$ to an arbitrarily small negative number, similarly when $\sum_{j=0}^{t} x_j - n < 0$, as $\lambda$ tends to $-\infty$ the term becomes arbitrarily small. So the optimum ensures that the constraint $\sum_{j=0}^{t} x_j = n$ is satisfied.

To maximize the expression, setting the partial derivatives to be zero we get,

$$\frac{\partial l}{\partial x_s} = \sum_{i=0}^{t} y_i \frac{a_{si}}{\sum_{j=0}^{t} x_j a_{ji}} - \lambda = 0 \ \forall 0 \le s \le t$$

and $\frac{\partial l}{\partial \lambda} = \sum_{j=0}^{t} x_j - n = 0$.

Matrix $A$ is stochastic, i.e. $\sum_{i=0}^{t} a_{si} = 1 \ \forall 0 \le s \le t$, as they are probabilities of transition out of a state. Consider the row vector, $x = yA^{-1}$ calculated by the algorithm. For this vector $x$, $\sum_{j=0}^{t} x_j a_{ji} = y_i$. Hence substituting above, $\frac{\partial l}{\partial x_s} = \sum_{i=0}^{t} y_i \frac{a_{si}}{y_i} - \lambda = \sum_{i=0}^{t} a_{si} - \lambda = 1 - \lambda$

Also $\sum_{j=0}^{t} y_j = \sum_{j=0}^{t}\sum_{i=0}^{t} x_i a_{ij} = \sum_{i=0}^{t}\sum_{j=0}^{t} x_i a_{ij}$
$= \sum_{i=0}^{t} x_i \sum_{j=0}^{t} a_{ij} = \sum_{i=0}^{t} x_i$. As $\sum_{j=0}^{t} y_j = n$ we have $\sum_{i=0}^{t} x_i = n$, satisfying $\frac{\partial l}{\partial \lambda} = 0$.

Thus at $x$, given by $x = yA^{-1}$, and $\lambda = 1$ we get a local maximum of $l(x,\lambda)$. We show that the local maximum is the global maximum, by analyzing the Hessian matrix, $H$, of $l(x,\lambda)$ and showing $x^T H x \le 0$, for all $x \in R^t$. Elements of H are given by,

$$h_{si} = \frac{\partial l}{\partial x_s \partial x_i} = -\sum_{h=1}^{m} y_h \frac{a_{ih}a_{sh}}{(\sum_{j=0}^{t} x_j a_{jh})^2} \forall 0 \leq s, i \leq m$$

.

$$\begin{aligned} x^T Hx &= \sum_{i=0}^{t}\sum_{s=0}^{t} h_{si}x_s x_i \\ &= \sum_{i=0}^{t}\sum_{s=0}^{t} -\sum_{h=0}^{t} y_h \frac{a_{ih}a_{sh}}{(\sum_{j=0}^{t} x_j a_{jh})^2} x_s x_i \\ &= -\sum_{i=0}^{t}\sum_{s=0}^{t}\sum_{h=0}^{t} \phi_h a_{ih} a_{sh} x_i x_s \end{aligned}$$

where

$$\phi_h = \frac{y_h}{(\sum_{j=0}^{t} x_j a_{jh})^2} \geq 0$$

Thus

$$\begin{aligned} x^T Hx &= -\sum_{h=0}^{t} \phi_h \sum_{i=0}^{t}\sum_{s=0}^{t} a_{ih}a_{sh}x_i x_s \\ &= -\sum_{h=0}^{t} \phi_h \sum_{i=0}^{t} a_{ih}x_i(\sum_{s=0}^{t} a_{sh}x_s) \\ &= -\sum_{h=0}^{t} \phi_h(\sum_{s=0}^{t} a_{sh}x_s)\sum_{i=0}^{t} a_{ih}x_i \\ &= -\sum_{h=0}^{t} \phi_h(\sum_{s=0}^{t} a_{sh}x_s)(\sum_{i=0}^{t} a_{ih}x_i) \\ &= -\sum_{h=0}^{t} \phi_h(\sum_{i=0}^{t} a_{ih}x_i)^2 \leq 0 \end{aligned}$$

□

The multiple column aggregate is $(n, \epsilon, \delta)$ reconstructible, is shown by applying the Chernoff bound, to bound the error in $y$, and then bounding the error added during inversion.

**Iterative Bayesian technique**

Let vectors $x$ and $y$ of size $2^k$ be the a priori distribution on states of the original rows, and posteriori distribution on states of perturbed rows, as introduced above. Let the original states of rows in $T$ selected from the a priori distribution be given by random variables $U_1, U_2, ....U_n$, while the states of the $n$ perturbed rows in $T'$ be given by the random variables $V_1, V_2, ...V_n$. Then for $0 \leq p, q \leq t = (2^k - 1)$ and $1 \leq i \leq n$, we have $Pr(V_i = q) = y_q/n$, and $Pr(U_i = p) = x_p/n$. Also $Pr(V_i = q|U_i = p) = a_{pq}$ is the transition probability from state $p$ to $q$.

From Bayes rule, we get

$$\begin{aligned} Pr(U_i = p|V_i = q) &= \frac{P(V_i = q|U_i = p)P(U_i = p)}{P(V_i = q)} \\ &= \frac{P(V_i = q|U_i = p)P(U_i = p)}{\sum_{r=0}^{t} P(V_i = q|U_i = r)P(U_i = r)} \\ &= \frac{a_{pq}\frac{x_p}{n}}{\sum_{r=0}^{t} a_{rq}\frac{x_r}{n}} \\ &= \frac{a_{pq}x_p}{\sum_{r=0}^{t} a_{rq}x_r}. \end{aligned}$$

We iteratively update $x$ using the equation

$$Pr(U_i = p) = \sum_{q=0}^{t} Pr(V_i = q)Pr(U_i = p|V_i = q).$$

This gives us the update rule,

$$x_p^{T+1} = \sum_{q=0}^{t} y_q \frac{a_{pq}x_p^T}{\sum_{r=0}^{t} a_{rq}x_r^T},$$

where vector $x^T$ denotes the iterate at step $T$, and vector $x^{T+1}$ the iterate at step $T + 1$.

We initialize the vector, $x^0 = y$, and iterate until two consecutive $x$ iterates do not differ much. This fixed point is the estimated a priori distribution. This algorithm is similar to the iterative procedure proposed in [AS00] for additive perturbation and shown in [AA01] to be the *Expectation Maximization* (EM) algorithm converging to the *Maximum Likelihood Estimator* (MLE).

### Error in Reconstruction

We provide here a brief analysis of the error in the reconstruction procedures. A quantitative analysis of the magnitude of error is easy for the inversion method, but such an analysis is much harder for the iterative method. Due to the randomization in the perturbation algorithm there are errors in the transition probabilities in matrix $A$. This causes $y$, the posteriori distribution after perturbation calculated from $T^{'}$, to have errors. Hence the reconstructed $x$ will have errors.

The error decreases as the number of rows, $n$, increases. Let $a^{'}_{ij}$ denote the actual fraction of original rows of state $i$ that were converted to state $j$. Then as $n$ increases, $a_{ij}$ will be a closer approximation to $a^{'}_{ij}$. The error decreases as $n^{-0.5}$ as indicated by Theorem 2.4.1, and verified empirically in Section 2.7.

The error in reconstruction increases as the number of reconstructed columns, $k$, increases, and the probability of retention, $p$, decreases. The largest and smallest eigenvalues of $A$ can be shown to be 1 and $p^k$ respectively and the condition number of the matrix $A$ grows roughly as $p^{-k}$ (see Section 2.7). The condition number of a matrix is a good indicator of the error introduced during inversion [GL].

## 2.5 Guarantees against privacy breaches

Private data from multiple clients is perturbed before being integrated at the server. In this section, we formalize the privacy obtained by this perturbation.

The notion of a $(\rho_1, \rho_2)$ privacy breach was introduced in [EGS03]. We extend this to introduce a new privacy metric, called the $(s, \rho_1, \rho_2)$ privacy breach. Consider a database of purchases made by individuals. It is quite likely that many people buy bread, but not many buy the same prescription medicine. The new metric is more concerned about whether an adversary can infer from the randomized row which medicine a person bought, and is less concerned about the adversary determining with high probability that the original row had bread, as most individuals buy bread and it does not distinguish the individual from the rest of the crowd.

Assume that the adversary has access to the entire perturbed table $T^{'}$ at the server, and the exact a priori distribution on the unperturbed data (which can be reconstructed [AS00])[1]. Also assume that any external information is already incorporated into the database.

### 2.5.1 Review of $(\rho_1, \rho_2)$ Privacy Breach

Consider a data element of domain $V_X$ perturbed by a perturbation algorithm into another domain $V_Y$.

**Definition 2.4** $(\rho_1, \rho_2)$ **Privacy Breach***[EGS03]: Let $Y$ denote the random variable corresponding to the perturbed value and $X$ that corresponding to the original value obtained from the a priori distribution. We say that there is a $(\rho_1, \rho_2)$ privacy breach with respect to $Q \subseteq V_X$ if for some $S \subseteq V_Y$ $P[X \in Q] \leq \rho_1$ and $P[X \in Q | Y \in S] \geq \rho_2$ where $0 < \rho_1 < \rho_2 < 1$ and $P[Y \in S] > 0$.*

Intuitively suppose the probability of an event, (age $\leq 10$) (say), according to the a priori probability is $\leq \rho_1 = 0.1$ (say). After observing the perturbed value, if the posteriori probability of the same event increases to $\geq \rho_2 = 0.95$ (say), then there is a (0.1,0.95) privacy breach with respect to the event (age $\leq 10$).

### 2.5.2 $(s, \rho_1, \rho_2)$ Privacy Breach

In retention replacement perturbations, which are of interest to us, the column is perturbed back into the same domain, and hence $V_X = V_Y$. Let $S \subseteq V_X$, with $P[X \in S] = p_s$, for $X \in_o V_X$ where $\in_o$ represents selecting an element from $V_X$ according to the a priori distribution on $V_X$. Let $P[Y \in S] = m_s$, for $Y \in_r V_X$, where $\in_r$ represents selecting an element from $V_X$ according to the replacing distribution,

[1]From Section 2.4.1, the error in the reconstructed a priori distribution for very selective predicates is large. This adds to the privacy of the perturbed rows.

which is different from the distribution of the perturbed table. The ratio $p_s/m_s$ is called the *relative a priori probability* of the set $S$.

The relative a priori probability is a dimensionless quantity that represents how frequent a set is according to its a priori probability as compared to the replacing p.d.f. (the uniform p.d.f.). In a database of purchases, medicines will have low relative a priori probability since different people take different medicines, while bread will have high relative a priori probability.

**Definition 2.5** $(s, \rho_1, \rho_2)$ **Privacy Breach:** *Let $Y$ denote the random variable corresponding to the perturbed value and $X$ that corresponding to the original value obtained from the a priori distribution.*

*Let $S \subseteq V_X$, we say that there is a $(s, \rho_1, \rho_2)$ privacy breach with respect to $S$ if the relative a priori probability of $S$, $p_s/m_s < s$, and if $P[X \in S] = p_s \leq \rho_1$ and $P[X \in S|Y \in S] \geq \rho_2$ where $0 < \rho_1 < \rho_2 < 1$ and $P[Y \in S] > 0$.*

The value of $s$ in the privacy breach is addressed by the next result.

**Theorem 2.5.1** *The median value of relative a priori probability, over all subsets $S$, $S \subseteq V_X$, is 1.*

**Proof:** Consider, any subset $S \subseteq V_X$, and $\overline{S} = V_X - S$. Using notation as in Definition 2.5 we have $p_s + p_{\overline{s}} = 1$ and $m_s + m_{\overline{s}} = 1$. Hence if $p_s/m_s \geq 1$ , we have $p_{\overline{s}}/m_{\overline{s}} \leq 1$ and if $p_s/m_s < 1$ we have $p_{\overline{s}}/m_{\overline{s}} > 1$ Since this is true for any pair of complementary subsets, among all subsets of $V_X$, half the subsets have relative a priori probability $\geq 1$ and half $\leq 1$. Hence the median value of $s$ over all subsets of $V_X$ will be 1, if the median is not constrained to be one of the values attained.

$\square$

We define rare sets as those that have relative a priori probability smaller than 1. We next show that privacy breaches do not happen for rare sets.

### 2.5.3 Single Column Perturbation

**Theorem 2.5.2** *Let $p$ be the probability of retention, then uniform perturbation applied to a single column is secure against a $(s, \rho_1, \rho_2)$ breach, if*

$$s < \frac{(\rho_2 - \rho_1)(1-p)}{(1-\rho_2)p}.$$

**Proof:** Let $S \subseteq V_X$ with $P[S] = p_s$ according to the a priori distribution and $P[S] = m_s$ according to the replacing p.d.f. Let $X$ and $Y$ denote the random variables for the original and perturbed value respectively. Let $R$ denote the event that $X$ was replaced and $R^c$ it being retained. For a $(\rho_1, \rho_2)$ privacy breach with respect to $S$ we need $P[X \in S] \leq \rho_1$. Also $P[(X \in S)|(Y \in S)]$

$$= \frac{P[(X \in S) \cap (Y \in S) \cap R] + P[(X \in S) \cap (Y \in S) \cap R^c]}{P[Y \in S]}$$

$$\leq \frac{\rho_1(1-p)m_s + pp_s}{(1-p)m_s + pp_s}$$

This is because $P[(X \in S) \cap (Y \in S) \cap R] = P[X \in S]P[Y \in S|R]P[R] \leq \rho_1(1-p)m_s$ and $P[(X \in S) \cap (Y \in S) \cap R^c] = P[(X \in S) \cap R^c] = pp_s$. Thus if $P[X \in S|Y \in S] \geq \rho_2$,

$$\frac{\rho_1(1-p)m_s + pp_s}{(1-p)m_s + pp_s} \geq \rho_2$$

Hence for a $(\rho_1, \rho_2)$ privacy breach with respect to $S$, we need

$$\frac{p_s}{m_s} \geq \frac{(\rho_2 - \rho_1)(1-p)}{(1-\rho_2)p}$$

□

As a concrete example, for uniform perturbation, with p=0.2, there are no (68, 0.1, 0.95) breaches. This means for any set $S$, if $\rho_2 > 0.95$ with uniform perturbation, $\rho_1$ will be large ($> 0.1$) when $p_s/m_s < 68$. In fact, for a rare set, with $s < 1$, there will be no $(0.937, 0.95)$ privacy breaches in the original $(\rho_1, \rho_2)$ model for this perturbation.

### 2.5.4 Multiple Independently Perturbed Columns

Let $D_i$ be the domain for column $i$ in a $k$ column table. Then the domain of the table, $D = D_1 \times D_2 \times \ldots D_k$. Each column of the table is perturbed independently by a retention replacement perturbation scheme.

There is an a priori probability distribution of the rows in table $T$. Let $S_i \subseteq D_i$ be a subset of the domain of the $i^{th}$ column for $1 \leq i \leq k$. Let $S = S_1 \times S_2 \times \ldots S_k$, then $S \subseteq D$. Let $P[S] = p_{S_1 \times S_2 \times \ldots S_k} = p_s$ (say) be the a priori probability of $S$. Let $P[Y_i \in S_i] = m_{S_i}$, for $Y_i \in_{\alpha_i} D_i$, where $\in_{\alpha_i}$ denotes selecting randomly from the replacing p.d.f. on $D_i$, for all $1 \leq i \leq k$. Then $P[Y \in S] = m_{S_1} m_{S_2} .. m_{S_k} = m_s$ (say) for $Y = (Y_1, Y_2, \ldots Y_k) \in_\alpha D$, where $\in_\alpha$ denotes selecting randomly from the replacing p.d.f. for each column independently. $p_s/m_s$, the relative a priori probability, is the ratio of the a priori probability to the replacing probability, of the combination of values for the columns together. Correlated columns with higher a priori probabilities have larger values of $p_s/m_s$.

**Theorem 2.5.3** *There will not be a $(\rho_1, \rho_2)$ privacy breach with respect to $(S_1 \times S_2 \times \ldots S_k) = S \subseteq D$, if*

$$\frac{p_s}{m_s} < \frac{\rho_2(1-\rho_1)(1-p)^k}{(1-\rho_2)\prod_{i=1}^{k}((1-p)m_{S_i} + p)}.$$

**Proof:** Let $X = (X_1, X_2, ..., X_k)$ be the random variable corresponding to the original value of the $k$ column row from the a priori distribution on table $T$, and $Y = (Y_1, Y_2, ....Y_k)$ that corresponding to the perturbed row, where each column is perturbed independently by a retention replacement perturbation. For $A_i, B_i \subseteq D_i$ we have $P[Y_i \in B_i | X_i \in A_i] = (1-p)m_{B_i} + p\frac{p_{A_i \cap B_i}}{p_{A_i}}$, for $1 \leq i \leq k$. Thus $(1-p)m_{B_i} \leq P[Y_i \in B_i | X_i \in A_i] \leq (1-p)m_{B_i} + p$. Thus for $A, B \subseteq D$ we have $L_B$ (say) $= \prod_{i=1}^{k}(1-p)m_{B_i} \leq P[Y \in B | X \in A] \leq \prod_{i=1}^{k}((1-p)m_{B_i} + p) = U_B$ (say). $P[(X \in S)|(Y \in S)] =$

$$\frac{P[(Y \in S)|(X \in S)]P[X \in S]}{P[(Y \in S)|(X \in S)]P[X \in S] + P[(Y \in S)|\neg(X \in S)]P[\neg(X \in S)]}$$

$$\leq \frac{U_S p_s}{U_S p_s + L_S(1-p_s)}$$

Suppose there is a $(\rho_1, \rho_2)$ privacy breach with respect to $S$, we need $P[X \in S] \leq \rho_1$, and $P[(X \in S)|(Y \in S)] \geq \rho_2$ Thus

$$\frac{U_S p_s}{U_S p_s + L_S(1-p_s)} \geq \rho_2$$

This implies

$$\frac{p_s}{L_S(1-p_s)} \geq \frac{\rho_2}{U_S(1-\rho_2)}$$

Substituting values of $U_S$, $L_S$ and noting that $p_s \leq \rho_1$ hence $1 - p_s \geq 1 - \rho_1$, we get

$$\frac{p_s}{\prod_{i=1}^{k} m_{S_i}} \geq \frac{\rho_2(1-\rho_1)(1-p)^k}{(1-\rho_2)\prod_{i=1}^{k}((1-p)m_{S_i}+p)}$$

□

$S_i$ denotes the subset on column $i$ within which the original value must be identified for the privacy breach. In the case, $S_i$ denotes a single value or a small range within the domain of a continuous column, hence $(1-p)m_{S_i} \ll p$. We approximate $(1-p)m_{S_i} + p$ by $p$ to get

$$\frac{\rho_2(1-\rho_1)(1-p)^k}{(1-\rho_2)\prod_{i=1}^{k}((1-p)m_{S_i}+p)} \geq \frac{\rho_2(1-\rho_1)(1-p)^k}{(1-\rho_2)p^k}(1-\epsilon)$$

for some small constant $\epsilon$. Thus for some small constant $\epsilon$, uniform perturbation applied individually to $k$ columns is secure against $(s, \rho_1, \rho_2)$ breaches for

$$s < \frac{\rho_2(1-\rho_1)(1-p)^k}{(1-\rho_2)p^k}(1-\epsilon).$$

As an example, for uniform perturbation with p=0.2 applied independently to two columns, there are no (273,0.1,0.95) breaches for joint events on the columns (when $m_{S_i}$ are small).

## 2.6 Extensions

### 2.6.1 Categorical Data

Consider a categorical column, $C$, having discrete domain $D$. Let $S \subseteq D$. A predicate $P$, on column $C$, using $S$ is defined as

$$P(x) \;=\; \begin{cases} true & \text{if } x \in S \\ false & otherwise. \end{cases}$$

Given the a priori and replacing p.d.f. on $D$, the reconstruction algorithms in Section 2.4 and the privacy guarantees in Section 2.5 can be directly applied to the categorical data by computing the probability of the predicate, $P$, being true.

### 2.6.2 Alternative Retention Replacement Schemes

Our analysis so far considered retention replacement perturbations where the replacing p.d.f is the uniform distribution. We now discuss some other interesting retention replacement schemes:

1 *Identity perturbation:* If the original data element is decided to be perturbed, the data element is replaced by a random element selected uniformly among all data elements [LCL85] (i.e. the replacing p.d.f. is the same as the a priori distribution).

2 *Swapping:* Swapping is closely related to identity perturbation. In swapping with probability $p$ we retain a data element, and with probability $(1 - p)$ we decide to replace it. Numbers decided to be replaced are then randomly permuted amongst themselves.

Identity perturbation and swapping are different from uniform perturbation which is a local perturbation. Identity perturbation can be local if there is knowledge of the a priori distribution before perturbation. Swapping is not a local perturbation and requires multiple rows at the client.

### Reconstructing Aggregates

Identity perturbation and swapping do not affect the answers to single column aggregate queries, i.e. answers to single column aggregate queries on the perturbed table, $T^{'}$, are returned directly as answers to those queries on the original table, $T$.

The difference in multi-column reconstruction for identity perturbation and swapping as compared to uniform perturbation is in the evaluation of vector $R$ in Section 2.4.2. Recall that $R_{i,1}$ is the probability that an element selected from the replacing p.d.f. on column $i$ satisfies the predicate on the $i^{th}$ column, $P_i$. The replacing p.d.f. (which is the original p.d.f. for identity perturbation and swapping) is required for reconstruction. This requires the server to have the original p.d.f. for each column. This requirement is however obviated by the observation in the previous paragraph, that the fraction of elements satisfying $P_i$ in $T$ is the same as the fraction of elements satisfying $P_i$ in $T^{'}$. Hence $R_{i,1}$ can be calculated from $T^{'}$. $R_{i,0}$ as before is calculated as $1 - R_{i,1}$.

The reconstruction error after identity perturbation and swapping will be smaller than that compared to uniform perturbation for sets, $S$, with small relative a priori probability. This is because in uniform perturbation the noise due to the perturbed data elements that now belong to $S$, but did not before perturbation, exceeds significantly the number of data elements that were in $S$ originally and retained during perturbation.

### Guarantees against Privacy Breaches

The guarantees for identity perturbation and swapping can be obtained using $m_{S_i} = p_{S_i}$ in Theorems 2.5.2 and 2.5.3. As an example we restate Theorem 2.5.2 for identity perturbation.

**Lemma 2.6.1** *For a single column, identity perturbation is secure against* $(s, \rho_1, \rho_2)$ *privacy breaches for*

$$\rho_1 < \frac{\rho_2 - p}{1 - p}.$$

**Proof:** For identity perturbation, $m_s = p_s$, hence $p_s/m_s = 1 \quad \forall S$. Repeating the argument in Theorem 2.5.2 we get $(\rho_2 - \rho_1)(1-p) > (1-\rho_2)p$, which implies the result. □

The above $(\rho_1, \rho_2)$ guarantee for identity perturbation is independent of the subset $S$. Uniform perturbation gives better $(\rho_1, \rho_2)$ guarantees for a set of rare data elements, i.e. a set with $p_s/m_s < 1$ and worse for sets with $p_s/m_s > 1$. Identity perturbation and swapping have a privacy breach in the presence of external knowledge about rare values (eg. the largest or smallest value). Rare values need to be suppressed (i.e. blanked out) [HT98] for privacy with these perturbations.

### 2.6.3 Application to Classification

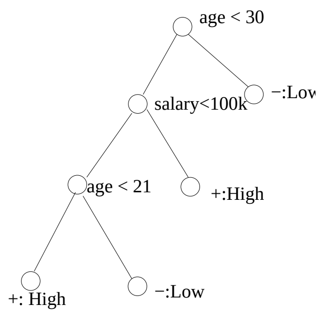


Figure 2.3: *Decision Tree Example*

We show how aggregate queries on multiple columns can be used for privacy preserving construction of decision trees [AS00]. Consider the tree in Figure 2.3 built on randomized table $T'$ with schema (age, salary, house-rent, class-variable) to predict the column class-variable. The column class-variable can take two values: $+$ and $-$ representing high and low credit-risk (say). The private columns among age, salary, house-rent and class-variable, are each independently perturbed by a retention replacement perturbation. Let $Q$ denote the predicate (class-variable = '+') while $\neg Q$ denote the predicate (class-variable='-').

For the first split, say on (age $<$ 30), the gini index is calculated using the estimated answers of the four queries: count(age[0-30] $\wedge$ $\neg$ Q), count($\neg$ age[0,30] $\wedge$ $\neg$ Q), count(age[0-30]$\wedge$ Q ) and count($\neg$ age[0,30]$\wedge$ Q ) on $T$. Now consider the left subtree of elements having (age $< 30$) using the predicate (salary $< 100k$). We do not partition the randomized rows at any level in the decision tree. Previously with additive perturbation, randomized rows were partitioned, and the columns were reconstructed independently [AS00]. With multi-column reconstruction the queries count(age[0-30] $\wedge$ salary[25k-100k] $\wedge$ $\neg$ Q), count(age[0,30] $\wedge$ salary[100k-200k] $\wedge$ $\neg$ Q ), count(age[0-30] $\wedge$ salary[25k-100k] $\wedge$ Q ) and count(age[0,30] $\wedge$ salary[100k-200k] $\wedge$ Q ) are reconstructed for $T$, to calculate the gini index or another split criterion at this level.

Now consider the third split, on age once again, but this time (age $< 21$), is decided after the queries count(age[0-21] $\wedge$ salary[25k-100k] $\wedge$ $\neg$ Q ), count(age[21-30] $\wedge$ salary[25k-100k] $\wedge$ $\neg$ Q ) count(age[0-21] $\wedge$ salary[25k-100k] $\wedge$ Q ) and count(age[21-30] $\wedge$ salary[25k-100k] $\wedge$ Q ) are reconstructed for $T$. The number of columns in the count query did not increase at this split on age, which was already present among the original set of queried columns.

## 2.7 Experiments

We next present an empirical evaluation of our algorithms on real as well as synthetic data. For real data, we used the Adult dataset, from the UCI Machine Learning Repository [BM98], which has census information. The Adult dataset contains about 32,000 rows with 4 numerical columns. The columns and their ranges are: age[17 - 90], fnlwgt[10000 - 1500000], hrsweek[1 - 100] and edunum[1 - 16].

For synthetic data, we used uncorrelated columns of data having Zipfian distribution with zipf parameter 0.5. We create three such tables with different number of rows. The number of rows is varied in factors of 10 from $10^3$ to $10^5$. The frequencies of occurrences are such that the least frequent element occurs 5 times. This results in the number of distinct values to be approximately one tenth of the number of rows in the table.

### 2.7.1 Randomization and Reconstruction

In this Section we assume that the vectors, $x$, $y$ described in Section 2.4.2 have been normalized, i.e. all elements have been divided by $n$, the number of rows, so that the sum of the elements of each vector is 1. These vectors will also be referred to as probability density function (p.d.f.) vectors. $x$ is the reconstructed p.d.f. vector, obtained by the inversion or iterative method in Section 2.4.2, while $y$ is the p.d.f. vector on the perturbed table before reconstruction. Let the exact original value of the p.d.f. vector calculated directly on the unperturbed table, $T$, be $x^{'}$. The $l_1$ norm of the difference between the estimated ($x$) and actual ($x^{'}$) p.d.f. vectors is used as the metric of error, and is referred to as the *reconstruction error*. The results of the reconstruction algorithm are quite accurate when the reconstruction error is much smaller than 1.

**Reconstruction algorithms:** We first study the reconstruction error while reconstructing multiple columns of the Adult dataset for varying retention probabilities. The predicates being reconstructed are age[25-45], fnlwgt[100000-1000000] and hrsweek[30-60]. Figure 2.4 shows the errors on first two among the above predicates while Figure 2.5 shows the errors on all three predicates. The retention probability, $p$, plotted on the $x$-axis, is the same for all columns. The reconstruction error is plotted on the $y$ axis. There are three curves in each figure. The curve *randomized*, shows the $l_1$ norm of the difference between the perturbed p.d.f. vector $y$ and the original p.d.f. vector $x^{'}$. It serves as a baseline to study the reduction in error after reconstruction of $y$ to $x$. The other two curves represent the reconstruction errors after the *iterative* and the *inversion* algorithms.

The iterative procedure gives smaller errors than the inversion procedure, especially when a larger number of columns are reconstructed together, and the probability of retention, $p$, is small. This is reconfirmed later by Figures 2.7 and 2.8, and similar experiments on synthetic data (which we do not show for the lack of space). This may seem unintuitive as the inversion algorithm was shown to give the MLE estimator for $x$, satisfying $\sum_i x_i = 1$ (after normalization). This can be explained by noting that the iterative algorithm gives the MLE estimator in the constrained space, i.e. for the subspace of $\sum_i x_i = 1$ that satisfies $0 \leq x_i \leq 1 \;\; \forall i$. Since the number of rows are always non-negative, this is the subspace that contains the exact original p.d.f. vector $x'$. When the retention probability decreases, and the number of columns to be reconstructed increases, the error during randomization and reconstruction increases, and the inversion algorithm may return a point outside the constrained space. The reconstruction error by the inversion method can grow arbitrarily. However, the iterative algorithm being constrained, will have a reconstruction error of at most two.

**Condition number:** Figure 2.6 shows the condition number [GL] of the transition matrix using a logarithmic scale on the $y$ axis, and the number of columns reconstructed on the $x$ axis, for different retention probabilities (p= 0.2, 0.5 etc.). The selectivity of each predicate is set to 0.5. The condition number (which is independent of the dataset) increases as the retention probability decreases and increases exponentially as the number of columns reconstructed increase. The condition number is a good indicator of the reconstruction error by the inversion algorithm [GL], and by the iterative Bayesian algorithm at small error values. Unlike the continuous exponential growth in error as the number of reconstructed columns increases for the inversion algorithm, the error flattens out for the iterative algorithm, as it is bounded above by two as discussed earlier.

## 2.7.2 Scalability

Next we study, how the reconstruction error varies as the number of columns reconstructed, retention probability, number of rows, and selectivity of the predicates vary.

**Number of columns and retention probability:** We study the reconstruction errors for varying number of columns and retention probabilities on the Adult dataset by the iterative and inversion algorithms. The predicates being reconstructed are age[ 25 - 45], fnlwgt[ 100000 - 1000000], hrsweek[ 30 - 60] and edulevel[ 5 - 10]. For the $i$ ( $1 \leq i \leq 4$ ) column experiment, the first $i$ among the above predicates are selected in the query. Figure 2.7 shows the reconstruction errors with the iterative algorithm, while Figure 2.8 shows the reconstruction errors with the inversion algorithm. Both iterative and inversion algorithms show an exponential increase in the error as the number of columns increases and as the probability of retention decreases. For smaller number of columns and higher retention probabilities both algorithms give comparable reconstruction errors. However for larger number of columns and lower retention probabilities the iterative algorithm gives smaller errors than the inversion algorithm. As explained in Section 2.7.1, unlike the iterative method, the reconstruction error by the inversion method can grow arbitrarily, whereas the error by the iterative method flattens out after an initial exponential increase.

For all experiments on the Zipfian dataset, the predicate on each column has an independent selectivity of 0.5. Figure 2.9 shows the reconstruction error after the iterative algorithm is applied to the perturbed Zipfian dataset of size $10^5$. The figure shows the increase in the reconstruction error, plotted on the $y$ axis, for increasing number of columns, plotted on the $x$ axis, for different retention probabilities. After an initial exponential increase, the reconstruction error flattens out.

**Number of rows in the table:** Figure 2.10 shows how the reconstruction error decreases as the number of perturbed rows available for reconstruction increase, for the the iterative reconstruction algorithm. In Figure 2.10 the retention probabilities are varied while the number of columns remains fixed at 8. For large values of $n$ the reconstruction error decreases as $n^{-0.5}$ as suggested by Theorem 2.4.1. This is also ratified by the factor 10 displacement between the reconstruction error lines for $10^3$ and $10^5$ rows in Figures 2.11 and 2.12. As the number of rows increases, it is possible to reconstruct more columns together at smaller retention probabilities.

**Selectivity of the predicates:** Recall that $e = x_{2^k-1}$ is estimate for the aggregate query and $a = x'_{2^k-1}$ is the actual answer for this query. $|e - a|$ is the called the absolute error while $|e - a|/a$ is called the relative error. Since we are interested in the variation of the error in the aggregate query with the selectivity of its predicate, for this set of experiments, we use the absolute and relative errors, instead of the $l_1$ norm of the difference of the p.d.f.vectors, as the error metric.

For the experiments a single Zipfian column is used with uniform perturbation with retention probability $p = 0.2$. We vary the selectivity of the predicate of the numeric column by varying the size of the interval in the range predicate. Figure 2.11

and Figure 2.12 study the variation in absolute and relative errors respectively, as the size of the interval being queried changes. The fractional interval width, i.e. the ratio of the size of the interval being queried to the entire domain of the column, is plotted on the $x$ axis while the error is plotted on the $y$ axis. The absolute error in Figure 2.11 does not vary much with the interval width. However the relative error in Figure 2.12 increases as the interval width decreases. Both the absolute and relative errors decrease as the number of rows available for reconstruction increases.

### 2.7.3 Privacy Breach Guarantees

We study privacy breaches possible after perturbation on the Adult dataset. Figure 2.13 and Figure 2.14 show the maximum retention probability that avoids breaches for varying values of $\rho_1$ for fixed $\rho_2 = 0.95$, according to Theorem 2.5.3. To compute the values of $s$ for sample predicates (subsets) of this dataset, we divide each column into 10 equiwidth intervals and consider predicates that are subsets formed by the cross product of the intervals. Thus for two columns we consider $10^2$ subsets and for three columns we consider $10^3$ subsets. The maximum values of $s$ were observed to be 15 and 30 for two and three columns respectively. The median value of $s$ has been shown to be one in Theorem 2.5.1. The two figures plot the maximum retention probability, $p$, that would avoid a $(s, \rho_1, \rho_2)$ breach, on the $y$ axis against the a priori probability, $\rho_1$, on the $x$ axis for different values of relative a priori probability, s. The values of $s$ used are the maximum value of $s$, the median value $s = 1$, and $s = 0.1$ for a rare set. Both figures show that if it suffices to just hide rare properties (i.e., with $s \leq 0.1$), then for $\rho_1 > 0.5$, the retention probability $p$ can be as high as 0.8. If we need to hide all the above properties, i.e. even for the largest $s$ (the most common property), then for $\rho_1 > 0.5$ the retention probability can be selected to be as high as $p = 0.3$. For $p = 0.3$ both Figure 2.4 and Figure 2.5 show low reconstruction error. Thus reconstructability of 2 and 3 aggregates together, and privacy of data elements, are both achieved by perturbation for the Adult dataset, with $p = 0.3$. Thus our experiments indicate $(s, \rho_1, \rho_2)$-privacy as well as multi-column aggregate reconstructability.

## 2.8 Conclusions

The contributions of this Chapter are:

- We introduce the problem of privacy preserving OLAP in a distributed environment.
- We introduce the formalism for reconstructible functions on a perturbed table, and develop algorithms to reconstruct multiple columns together. We provide privacy guarantees that take into account correlations between any combination of categorical and numeric columns.
- We provide two reconstruction algorithms to work with retention replacement perturbation: an iterative Bayesian algorithm, and a matrix inversion algorithm that also yields the maximum likelihood estimator. These algorithms can reconstruct count aggregates over subcubes without assuming independence between columns.
- We evaluate proposed reconstruction algorithms both analytically and empirically. We study the privacy guarantees we get for different levels of reconstruction accuracy and show the practicality of our techniques.
- We show the use of our techniques to related applications like classification.

  Future work includes extending this work to other aggregates over subcubes.

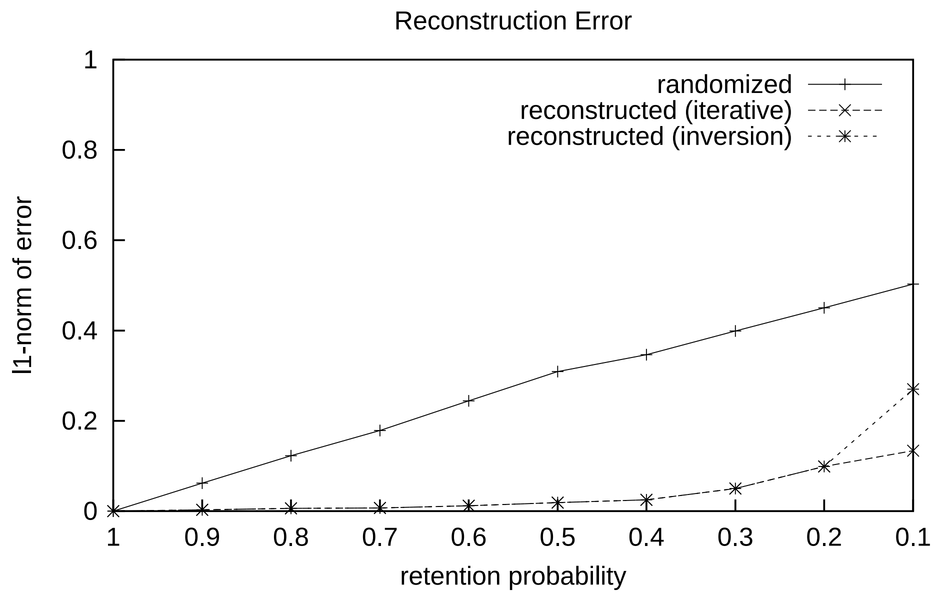


Figure 2.4: *Reconstruction errors for conjunction of 2 predicates for Adult data.*

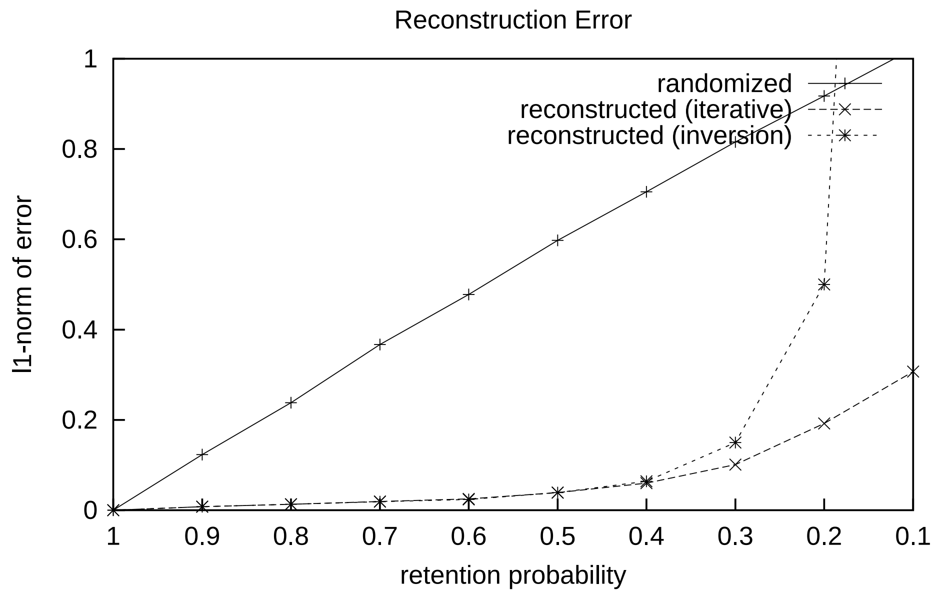


Figure 2.5: *Reconstruction errors for conjunction of 3 predicates for Adult data.*

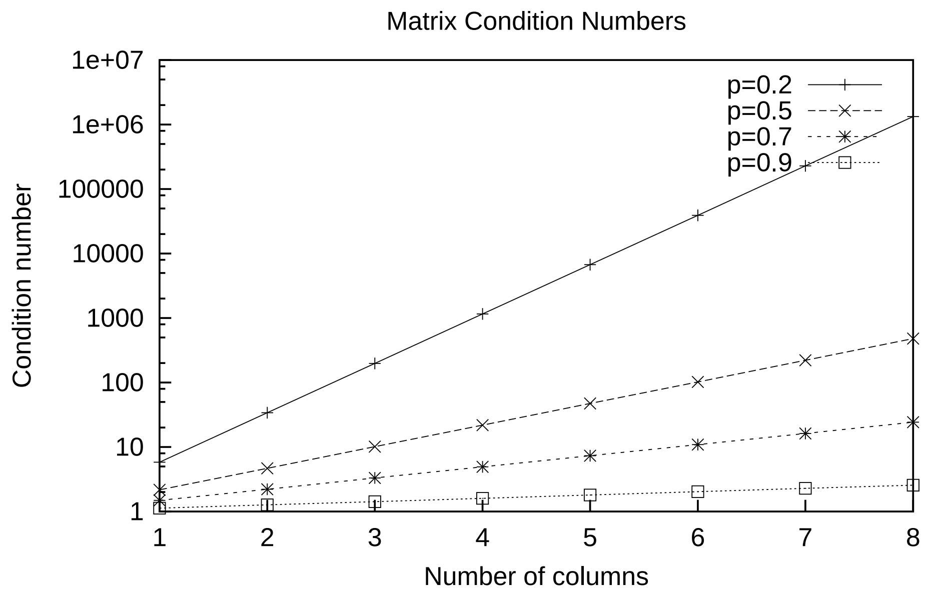


Figure 2.6: *Condition number of the transition matrix*

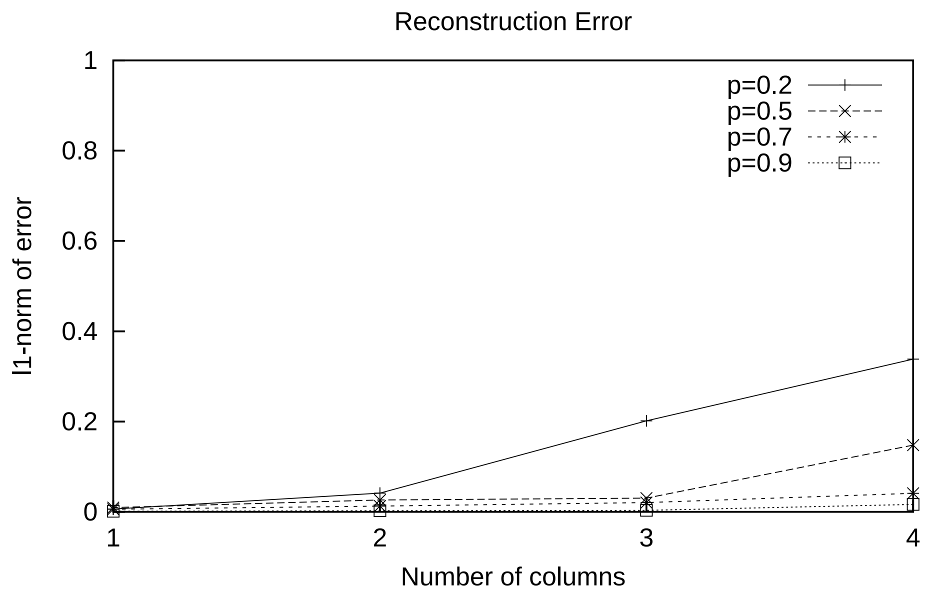


Figure 2.7: *Reconstruction errors for the Adult dataset for varying retention probabilities, p, by the iterative algorithm.*

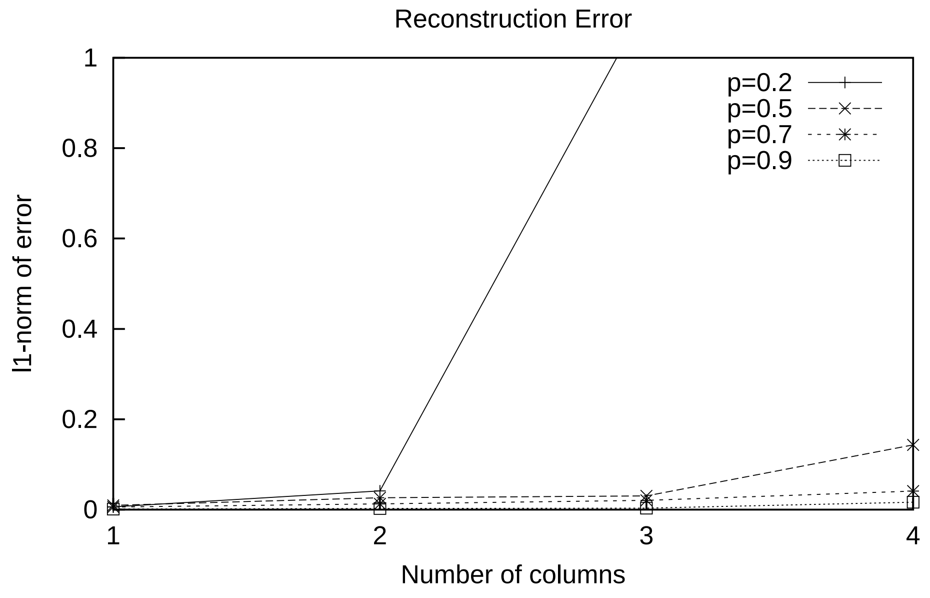


Figure 2.8: *Reconstruction errors for the Adult dataset for varying retention probabilities, p, by the inversion algorithm.*

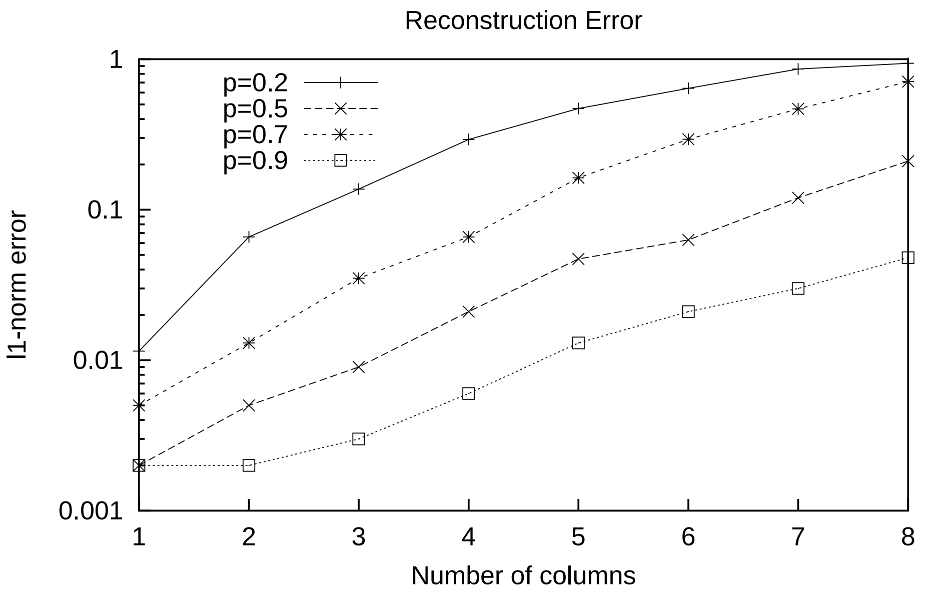


Figure 2.9: *Reconstruction error by iterative method on Zipfian dataset with* $10^5$ *rows varying number of columns*

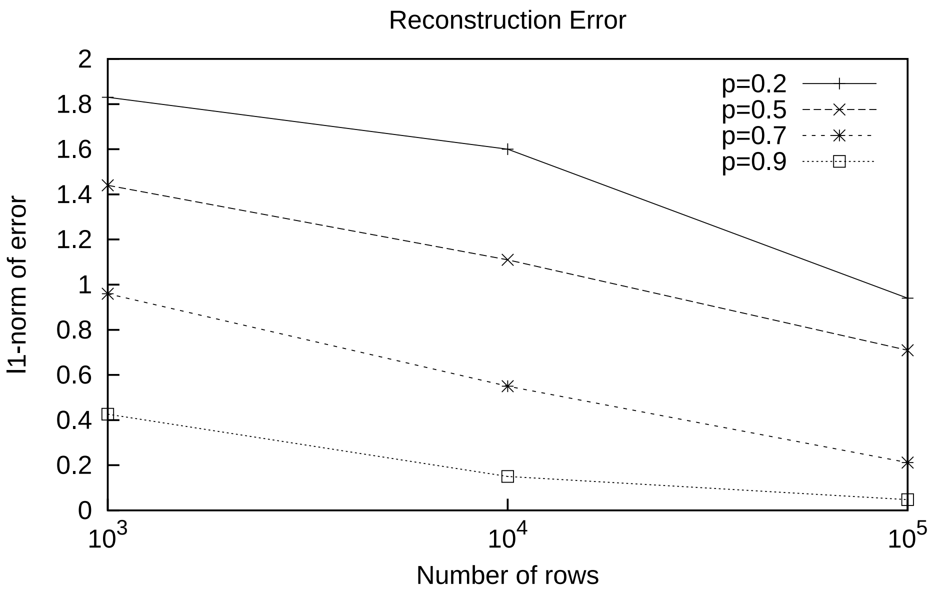


Figure 2.10: *Reconstruction error by iterative method on Zipfian dataset varying number of rows for 8 columns.*

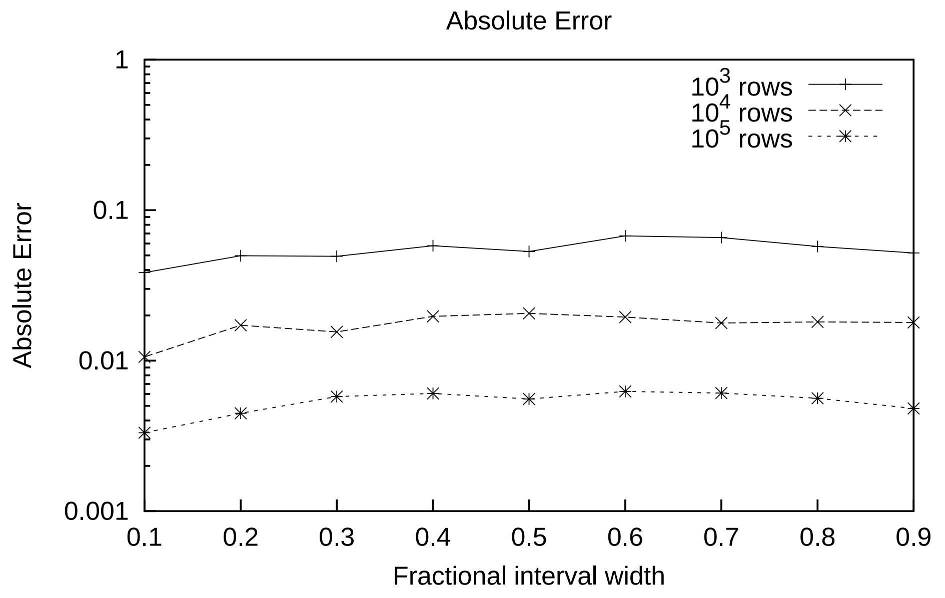


Figure 2.11: *Absolute Error for the Zipfian dataset for p=0.2 for varying interval sizes.*

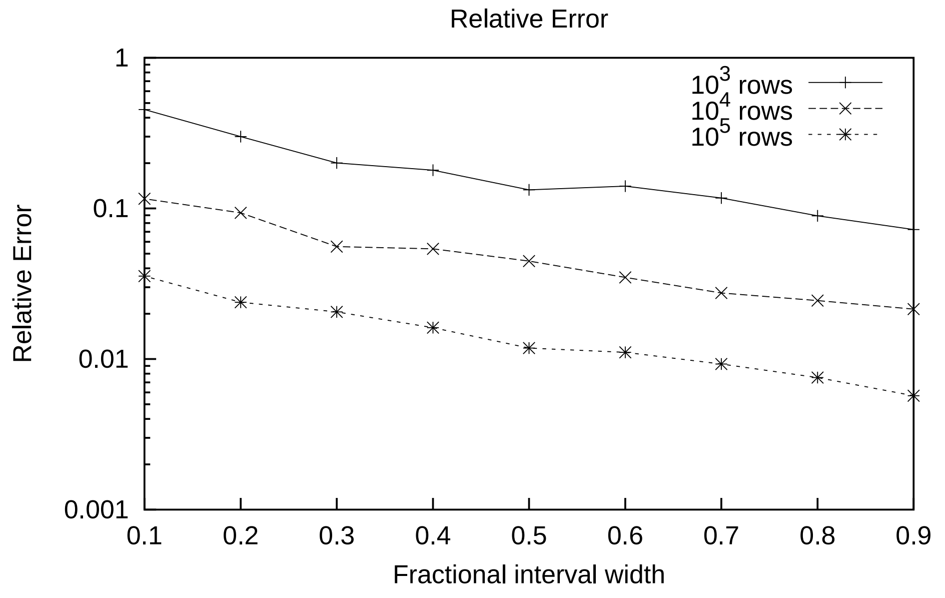


Figure 2.12: *Relative Error for the Zipfian dataset for p=0.2 for varying interval sizes.*

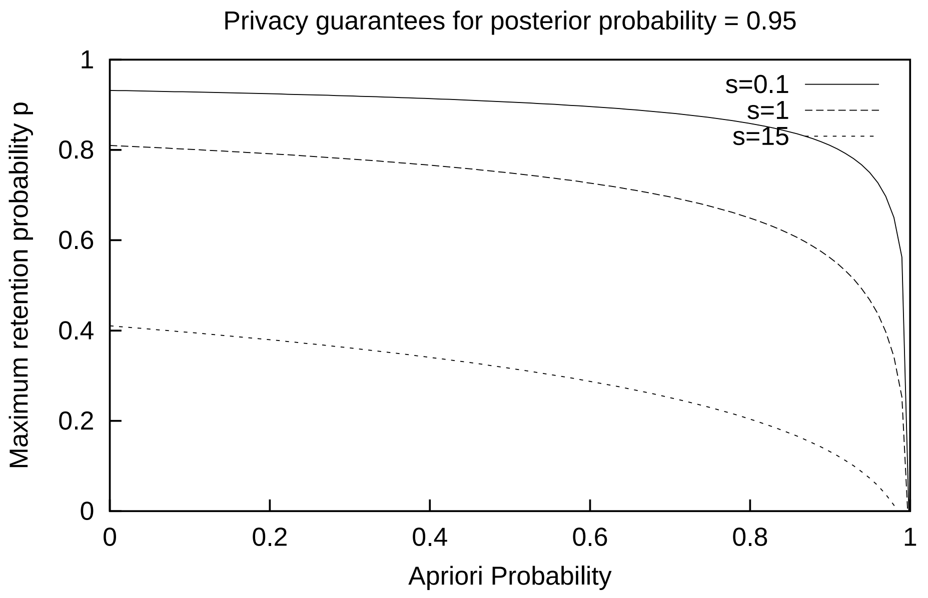


Figure 2.13: *Privacy for two columns for Adult data.*

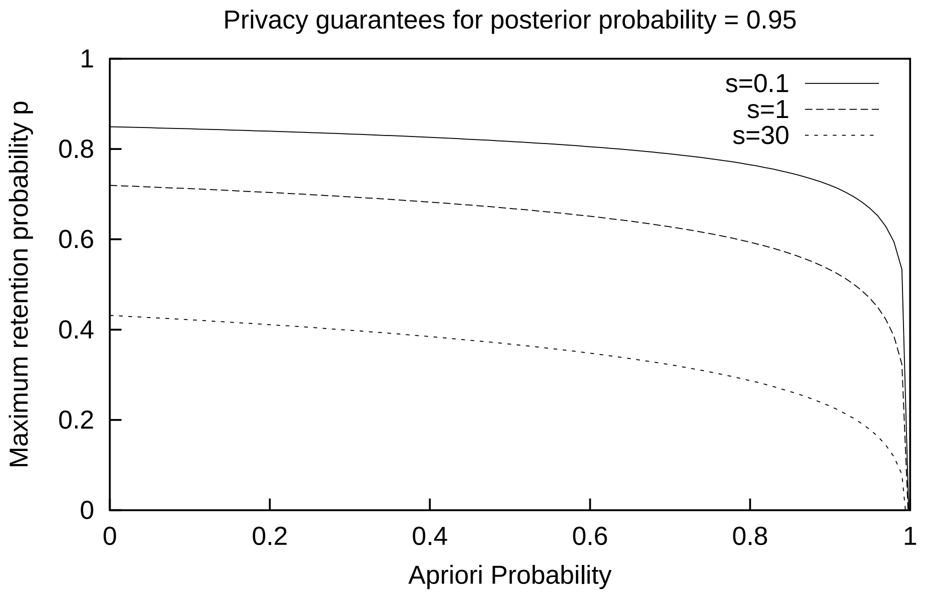


Figure 2.14: *Privacy for three columns for Adult data.*

# Chapter 3

# Clustering for Anonymity

The results in this chapter appear in [AFK+06].

## 3.1 Introduction

With the rapid growth in database, networking, and computing technologies, a large amount of personal data can be integrated and analyzed digitally, leading to an increased use of data-mining tools to infer trends and patterns. This has raised universal concerns about protecting the privacy of individuals [Tim97].

| Age | Location | Disease |
|---|---|---|
| $\alpha$ | $\beta$ | Flu |
| $\alpha + 2$ | $\beta$ | Flu |
| $\delta$ | $\gamma + 3$ | Hypertension |
| $\delta$ | $\gamma$ | Flu |
| $\delta$ | $\gamma$ -3 | Cold |

(a) Original table

| Age | Location | NumPoints | Disease |
|---|---|---|---|
| $\alpha$ +1 | $\beta$ | 2 | Flu |
| | | | Flu |
| | | | Hypertension |
| $\delta$ | $\gamma$ | 3 | Flu |
| | | | Cold |

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

| Age | Location | Disease |
|---|---|---|
| * | $\beta$ | Flu |
| * | $\beta$ | Flu |
| $\delta$ | * | Hypertension |
| $\delta$ | * | Flu |
| $\delta$ | * | Cold |

(b) 2-anonymized version

| Age | Location | NumPoints | Radius | Disease |
|---|---|---|---|---|
| $\alpha$ +1 | $\beta$ | 2 | 1 | Flu |
| | | | | Flu |
| | | | | Hypertension |
| $\delta$ | $\gamma$ | 3 | 3 | Flu |
| | | | | Cold |

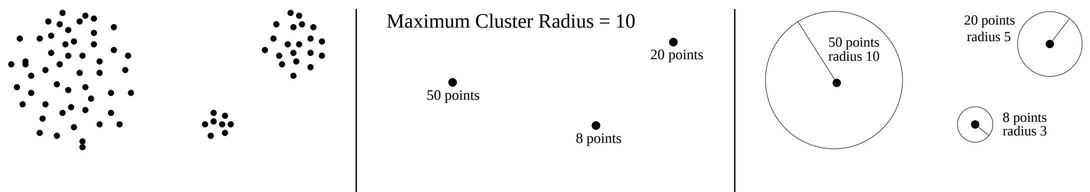
(d) 2-cellular clustering, with total cost 11

Figure 3.1: *Original table and three different ways of achieving anonymity*

Combining data tables from multiple data sources allows us to draw inferences which are not possible from a single source. For example, combining patient data from multiple hospitals is useful to predict the outbreak of an epidemic. The traditional approach of releasing the data tables without breaching the privacy of individuals in the table is to de-identify records by removing the identifying fields such as name, address, and social security number. However, joining this de-identified table with a publicly available database (like the voters database) on columns like race, age, and zip code can be used to identify individuals. Recent research [Swe00] has shown that for 87% of the population in the United States, the combination of non-key fields like date of birth, gender, and zip code corresponds 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, Sweeney [Swe02b] 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 3.1(a), Figure 3.1(b) shows a 2-anonymized table corresponding to it. The columns corresponding to sensitive attributes, like disease in this example, are retained 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. There has been a lot of recent work on $k$-anonymizing a given database table [BA05, LDR05a]. An $O(k \log k)$ approximation algorithm was first proposed for the problem of $k$-Anonymity with suppressions only [MW04]. This was recently improved to an $O(k)$ approximation for the general version of the problem [AFK$^+$05b].

In this paper, 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 satisfying the triangle inequality) over 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



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

Figure 3.2: *Publishing anonymized data*

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 paper 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 3.1, we assume that the quasi-identifying attributes have already been scaled. As we see above, it is quite complicated to algorithmically derive a metric space over quasi-identifying attributes of records; we do not pursue it any further in

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

this paper and leave it for future work.

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'll also publish a measure of the quality of the clusters. This will give a bound on the error introduced by the clustering.

In this paper 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 3.1(a) gives the table in Figure 3.1(c). In this example, the maximum radius over all clusters is 3. As another example, Figure 3.2(b) gives the output of the $r$-Gather algorithm applied to the quasi-identifiers, shown as points in a metric space in Figure 3.2(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. It's also worth noting that 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 in this paper. 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 3.1(a). Figure 3.1(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 3.2(c) gives the output of the $r$-Cellular Clustering algorithm applied to the quasi-identifiers shown as points in a metric space in Figure 3.2(a). The total cost of the solution in Figure 3.2(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 don't have the lower bound on 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 3.3 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).

Just like $k$-Anonymity, $r$-Gather and $r$-Cellular Clustering is 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 $\epsilon$ 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

| | 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 3.3: *A sample table where there is no common attribute among all entries.*

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, like 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 $\epsilon$ fraction of unclustered points, which we call the $(r, \epsilon)$-Gather and $(r, \epsilon)$-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 the paper is organized as follows. First, in Section 3.2, we present a tight 2-approximation algorithm for the $r$-Gather problem and its extension to the $(r, \epsilon)$-Gather problem. In Section 3.3, 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 $\epsilon$ fraction of the points unclustered. We conclude in Section 3.4.

## 3.2 $r$-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 don't 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 3.1** *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.

### 3.2.1 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 $G$.

### 3.2.2 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 $\ell$ 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 3.2.1** *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 remaining 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 assignment 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 it to the center that marked it covered. Each chosen center will have at least $r$ points assigned to it (including itself). $\square$

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 3.2.2** *There exists a polynomial time algorithm that produces a 2-approximation to the $r$-Gather problem.*

### 3.2.3 $(r, \epsilon)$-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 $\epsilon$ fraction of the points unclustered, *i.e.*, to delete an $\epsilon$ 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 $(r, \epsilon)$-Gather problem, where each cluster is constrained to have at least $r$ points and an $\epsilon$ 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 $G$ 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 $v$ from $G$. 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 3.2.3** *There exists a polynomial time algorithm that produces a 4-approximation to the* $(r,\epsilon)$-GATHER *problem.*

We note that in the problem formulation of $(r,\epsilon)$-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. We defer the details to the full version of the paper.

### 3.2.4 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.

It is worth mentioning 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 [JBIP93] 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 though we only concentrate on the $(k, r)$-CENTER problem defined above.

We note here that the algorithm developed for $r$-GATHER in Subsection 3.2.2 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 3.2.1 holds for the modified conditions, hence an approximation factor of 2.

We also consider the outlier version of this problem, namely, the $(k, r, \epsilon)$-CENTER problem. Combining the techniques presented in this paper and the techniques for the $(k, \epsilon)$-CENTER problem by Charikar et. al [CKMN01], one can devise a 4-approximation algorithm. We defer the details to the full version of the paper.

## 3.3 Cellular Clustering

As mentioned in the introduction, 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 3.2** *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:

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

And the dual program is:

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

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.

Procedure 1 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 procedure, 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 procedure.

We now show that the primal solution constructed has a cost of at most 4 times the value of the dual solution found using Procedure 1. For this, we note the following properties:

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

**Procedure 1** *A Primal Dual Method*

```
1: repeat
2:   Grow the unfrozen dual variables $\alpha_i$ uniformly.
3:   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
4:     Open the cluster $c$ partially, and grow the dual variable $\beta_{ic}$ at the same rate
       as $\alpha_i$.
5:   end if
6:   if $\sum_i \beta_{ic} = f_c$ for some cluster $c$ then
7:     Freeze all variables $\alpha_i$ for which the edge $(i, c)$ is tight.
8:     All unassigned points with a tight edge to $c$ are assigned to $c$. Call this set
       $V_c^U$.
9:     Let $V_c^I$ be the set of all idle points that have a tight edge to $c$.
10:    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 $V_c^{IU}$.
11:    All points in $V_c^{IU}$ are labeled idle and their dual variables are frozen.
12:    All points in $V_c^U$ and $V_c^I$ are labeled dead.
13:  end if
14: until All points become dead or idle.
```

(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 procedure. Property (2) follows from property (1) and the fact that the edge $(i, c)$ becomes tight when the dual variable $\alpha_i$ equals $d_c$. Property (3) then follows from (2). Property (4) again follows from the definition of the the procedure. 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 procedure, these members will be assigned to $c'$ and become *idle*.

**Lemma 3.3.1** *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 $\alpha$. 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 3.3.2** *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 3.3.3** *The primal-dual method in Procedure 1 produces a 4-approximation solution to the* Cellular Clustering *problem.*

## 3.3.1 $r$-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 $\epsilon$ 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 $(r/2, 5.184 OPT_r)$ was achieved independently by Guha, Meyerson and Munagala and by Karger and Minkoff [GMM00, 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 $(r, 80 OPT_r)$ 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 3.4.

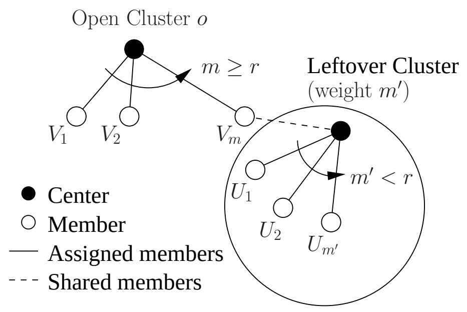


Figure 3.4: *Structures of open and leftover clusters*

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 3.4, 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 3.3.4** *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 $2r - 1$. $\square$

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 3.3.5** *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 $d(q, p) \leq 2d_{c''} + 2d_o + d_{c'} \leq 5d_{c'}$. $\square$

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 3.3.4, 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 3.3.6** *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 $(r, \epsilon)$-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. Details are deferred to the full version of the paper.

## 3.4 Conclusions

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 paper, 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 $\epsilon$ fraction of points to remain unclustered.

# Chapter 4

# Probabilistic Anonymity

The results in this chapter appear in [LT06].

## 4.1 Introduction

*"Over a year and a half, one individual impersonated me to procure over* $50,000 *in goods and services. Not only did she damage my credit, but she escalated her crimes to a level that I never truly expected: she engaged in drug trafficking. The crime resulted in my erroneous arrest record, a warrant out for my arrest, and eventually, a prison record when she was booked under my name as an inmate in the Chicago Federal Prison."* - An excerpt from the verbal testimony of Michelle Brown to a US Senate Committee [Bro00].

Unfortunately, in today's highly networked digital world, incidents like the above with Michelle Brown are commonplace. According to Bureau of Justice Statistics Bulletin [Bau06], 3.6 million households, representing 3% of the households in the United States, discovered that at least one member of the household had been the victim of identity theft during the previous 6 months in 2004. According to the same report, the estimated loss as a result of identity theft was about $ 3.2 billion. Needless to say that preventing identity thefts is one of the top priorities for government, corporations and society alike.

Globalization further complicates this picture. Due to legal directives or business associations, there are multiple scenarios where in organizations need to share or publish their micro-data to remain competitive. This puts personal privacy at further risk. To surmount this risk, attributes that clearly identify individuals, such as `Name`, `Social Security Number`, `Driving License Number`, are generally removed or replaced by random values. But this may not be enough because such de-identified databases can sometimes be joined with other public databases on seemingly innocuous attributes to re-identify individuals who were supposed to remain anonymous. For example, according to one study [Swe02b], approximately 87% of the population of the United States can be uniquely identified on the basis of `Gender`, `Date of Birth`, and 5-digit `Zipcode`. The uniqueness of such attribute combinations leads to a class of attacks where data is re-identified by joining multiple and often publicly available data-sets. This type of attack was illustrated by Sweeney in [Swe02b] where the author was able to join a public voter registration list and the de-identified patient data of Massachusetts' state employees to determine the medical history of the state's governor.

In literature, such an identity-leaking attribute combination is called as a *quasi-identifier*. It is always critical to be able to recognize quasi-identifiers and to apply to them appropriate protective measures to mitigate the identity disclosure risk posed by join attacks. In fact, Sweeney herself proposed a $k$-anonymity model in [Swe00] for the same. According to her, a database table is said to be $k$-anonymous if for each row in the table there are $k-1$ other rows in the table that are *identical* along the quasi-identifier attributes. Clearly, a join with a $k$-anonymous table would give rise $k$ or more matches and create confusion. Thus, an individual is hidden in a crowd of size $k$ giving her $k$-anonymity. It also means that the identity disclosure risk is at most $1/k$ for "join" class of attacks.

Although such a simple and clear quantification of privacy risk makes $k$-anonymity model attractive, its widespread use in practice is severely hampered owing to the following factors:

1. Choice of $k$ is not clear. From pure privacy point of view, larger $k$ would mean more privacy, but it comes at the cost of utility [Agg05]. What is the right

choice of $k$ for the given data and the given notion of utility has not been very well understood yet.

2. For $k$-anonymity model to be effective, it is critical that there is a complete understanding of the quasi-identifiers for the give data-set. But there is no real formalism available for deciding whether an attribute combination could form a quasi-identifier. This is currently done manually, based on folk-lore and human expertise.

3. For a given $k$, the goal is always to minimally suppress or generalize the data such that the resultant data-set is $k$-anonymous. However, for some natural notions of measuring this resultant distortion, the minimization problems turn out to be NP-Hard [MW04, AFK$^+$05a, AFK$^+$06].

   On the approximation front, no efficient but good approximation algorithms are currently known. The known algorithms are either $\tilde{O}(k)$ approximations [MW04, AFK$^+$05a] or super-linear [AFK$^+$06] - thus making them inefficient or expensive.

### 4.1.1 Organization and Contributions

In this chapter, we start out by providing the first formal characterization and a practical technique to identify quasi-identifiers. In Section 4.2, we also show an interesting connection between whether a set of columns forms a quasi-identifier and the number of distinct values assumed by the combination of the columns.

We then use this characterization in Section 4.3 to come up with a probabilistic notion of anonymity. Again we show an interesting connection between the number of distinct values taken by a combination of columns and the anonymity it can offer. This allows us to find an ideal amount of generalization or suppression to apply to different columns in order to achieve probabilistic anonymity. We work through many examples and show that our analysis can be used to make a published database conform to privacy acts like HIPAA.

In order to achieve the probabilistic anonymity, we observe that one needs to solve multiple 1-dimensional $k$-anonymity problems. In Section 4.4, we propose many efficient and scalable algorithms for achieving 1-dimensional anonymity. Our algorithms are optimal in a sense that they minimally distort data and retain much of its utility. The algorithms provided are a stark contrast to previous NP-hard results and comparatively more complicated algorithms for the previous notion of anonymity called $k$-anonymity [Swe02b].

We then experimentally verify our algorithms on real life data sets in Section 4.5. We sketch the related work in Section 4.6 and finally conclude in Section 4.7.

## 4.2 Automatic Detection of Quasi-identifiers

**Definition 4.1** *A quasi-identifier set $Q$ is a minimal set of attributes in table $T$ that can be joined with external information to re-identify individual records (with sufficiently high probability).*

Above definition is from [SS98]. A similar definition can be found in an earlier paper of Dalenius [Dal86]. As the reader can sense, this definition is informal since it does not make "external information" and "sufficiently high probability" explicit. Possibly because of this, we do not know any formal procedure or test for identifying quasi-identifiers. Almost always, researchers and practitioners assume that quasi-identifier attribute sets are known based on specific knowledge domain [LDR05b].

We present a more formal definition of quasi-identifier below. In our definition, we do not insist on minimality of attribute set as such although one could easily accommodate it if required. The external information is the *universal table* $\mathcal{U}$ having information about entire (relevant) population. It has $n$ rows. Typically, $\mathcal{U}$ would mean census records that many countries make readily available [Bur].

**Definition 4.2 $\alpha$-quasi-identifier** *An $\alpha$ quasi-identifier is a set of attributes along which an $\alpha$ fraction of rows in the universe can be uniquely identified by values along the combination of these attribute columns.*

**Example 1** *Empirically it has been observed that 87% of the people in the U.S. can be uniquely identified by the combination of `Gender`, `Date of Birth` and `Zipcode`.*

*Therefore (`Gender`, `Date of Birth`, `Zipcode`) forms a* $0.87$*-quasi-identifier for the U.S. population. Note that the U.S. census table is our universal table* $\mathcal{U}$ *here.*

Ideally, given an $\alpha$ and $\mathcal{U}$, it is straight-forward to figure out whether some particular attribute combination forms an $\alpha$-quasi-identifier in $\mathcal{U}$ by simply measuring the number of singletons in that attribute combination. One may even try an apriori like approach [AS94] and calculate all $\alpha$-quasi-identifiers in $\mathcal{U}$. In practice, there are errors in $\mathcal{U}$ that come in during data collection phase itself [CGGM03, Cen] and the knowledge about $\mathcal{U}$ is never exact. This would lead to erroneous conclusions about a quasi-identifier. Therefore, it does not justify the expensive calculations given above. In fact, one then prefers a quick and inexpensive approach that gives a good *estimate* of the same.

In what follows, we assume that the universal table $\mathcal{U}$ itself is not known. What we know is that it is a *random sample* built *with replacement* from a probability space. Thus our analysis is probabilistic. For the sake of analysis, we require that there is a probability distribution, but in reality, our final results are independent of this probability distribution. Moreover, we work only with the expectations since our goal is to give *good estimates* quickly. Since the sum of random variables is tightly concentrated around the expectation (by bounds like the Chernoff bounds [Che52]), our analysis and results are quite fair. We do not work out the Chernoff analysis though in order to keep our results and presentation simple.

We build our probability space on the distinct values that an attribute combination can take. Therefore, we need to know the number of distinct values for every attribute combination. Since one can get (or reasonably estimate) the count of distinct values for each attribute in $\mathcal{U}$ [Gib01], we simplify our task with the following assumption.

**Definition 4.3 Multiple Domain Assumption** *Let* $d_1$*,* $d_2$*, …,* $d_k$ *be the number of distinct values along columns* $C_1$*,* $C_2$*, …,* $C_k$ *respectively. Then, the total number of distinct values taken by the* $(C_1, C_2, \ldots, C_k)$ *column set is* $D = d_1 \times d_2 \times \ldots d_k$*.*

**Example 2** *We study the number of distinct values taken by the set of columns (`Gender`, `Date of Birth`, `Zipcode`). The number of distinct values of column*

*`Gender` ($C_1$) is $d_1 = 2$. The number of distinct values of column `Date of Birth` ($C_2$) can be approximated as $d_2 = 60 * 365 \approx 2 * 10^4$.*[1] *The number of distinct values along column `Zipcode` ($C_3$) is $d_3 = 10^5$. The number of distinct values of the column-set (`Gender`, `Date of Birth`, `Zipcode`) is $D = d_1 \times d_2 \times d_3 = 2 * (2 * 10^4) * 10^5 = 4 * 10^9$.*

*As another example, consider the set of columns (`Nationality`, `Date of Birth`, `Occupation`). The number of distinct values of column `Nationality` ($C_1$) is $d_1 = 200$. Once again, the number of distinct values of column `Date of Birth` ($C_2$) can be approximated as $d_2 = 60 * 365 \approx 2 * 10^4$. The number of distinct values of column `Occupation` ($C_3$) is roughly $d_3 = 100$. Thus $D = d_1 \times d_2 \times d_3 = 200 * (2 * 10^4) * 100 = 4 * 10^8$.*

Suppose that a set of columns take $D$ different values with probabilities $p_1$, $p_2$, ..., $p_D$, where $\sum_{i=1}^{D} p_i = 1$. Let us first calculate the probability that the $i^{\text{th}}$ element is a singleton in the universal table $\mathcal{U}$. It means first selecting one of the entries in the table (there are $n$ choices), setting it to be this $i^{\text{th}}$ element (which has probability $p_i$), and setting all other entries in the table to something else (which happens with probability $(1 - p_i)^{n-1}$). Thus, the probability of $i^{\text{th}}$ element being a singleton in the universal table $\mathcal{U}$ is $np_i(1 - p_i)^{n-1}$.

Let $X_i$ be the indicator variable representing whether $i^{\text{th}}$ element is a singleton. Then, its expectation

$$E[X_i] = P[X_i = 1] = np_i(1 - p_i)^{n-1} \approx np_i e^{-np_i}.$$

Let $X = \sum_{i=1}^{D} X_i$ be the counter for the number of singletons. Now its expectation is given by

$$E[X] = \sum_{i=1}^{D} E[X_i] = \sum_{i=1}^{D} np_i e^{-np_i}.$$

Let us analyze which distribution maximizes this expected number of singletons. We aim to maximize $\sum_{i=1}^{D} x_i e^{-x_i}$, subject to $\sum_{i=1}^{D} x_i = n$ and $0 \le x_i$, $\forall 1 \le i \le D$.

[1]Throughout this chapter we assume that the ages of people belonging to the database comes from an interval of size 60 years.

**Theorem 4.2.1** *If $D \leq n$, then the expected number of singletons is bounded above by $\frac{D}{e}$.*

**Proof:** If $f(x) = xe^{-x}$, $f^{'}(x) = (1-x)e^{-x}$ and $f^{''}(x) = (x-2)e^{-x}$. Thus, the function $f$ has a global maximum at $x = 1$, since $f^{'}(1) = 0$ and $f^{''}(1) < 0$.

Now the expected number of singletons,

$$\sum_{i=1}^{D} x_i e^{-x_i} \leq \sum_{i=1}^{D} e^{-1} = \frac{D}{e}.$$

This expression is a tight upper bound on the expected number of singletons for $D \leq n$. For example, it is almost obtained by setting $x_i = 1$, for $i = 1, 2, \ldots, D-1$, and $x_D = n - D + 1$. □

**Theorem 4.2.2** *If $D \geq n$, then the expected number of singletons is bounded above by $ne^{\frac{-n}{D}}$.*

**Proof:** If $f(x) = xe^{-x}$, $f^{'}(x) = (1-x)e^{-x}$ and $f^{''}(x) = (x-2)e^{-x}$. The function $f$ has a point of inflection at $x = 2$, since $f^{''}(x) < 0$ for $x < 2$ implying the function is concave here, and $f^{''}(x) > 0$ for $x > 2$ implying the function is convex here.

First we claim that on maximizing $\sum_{i=1}^{D} x_i e^{-x_i}$, no $x_i \geq 2$. Suppose otherwise: after maximizing $\sum_{i=1}^{D} x_i e^{-x_i}$, some $x_a \geq 2$. As $D \geq n$, and $\sum_{i=1}^{D} x_i = n$, some $x_b < 1$. For some small $\delta$, replacing $x_a$ by $x_a - \delta$ and $x_b$ by $x_b + \delta$ we retain $\sum_{i=1}^{D} x_i = n$. As $f(x) = xe^{-x}$ increases towards x=1, $f(x_a - \delta) > f(x_a)$ and $f(x_b + \delta) > f(x_b)$. Thus $\sum_{i=1}^{D} x_i e^{-x_i}$ is increased, contradicting the fact that it was maximized. Thus, $\forall 1 \leq i \leq D$, $x_i \leq 2$ .

Now $f^{''}(x) < 0$ for $0 \leq x \leq 2$. Since $f$ is concave, we can apply Jensen's inequality [Rud87] [2] to get

$$\begin{aligned}\sum_{i=1}^{D} x_i e^{-x_i} &= D\sum_{i=1}^{D}\frac{1}{D}x_i e^{-x_i}\\ &\leq D\cdot(\sum_{i=1}^{D}\frac{x_i}{D})e^{-(\sum_{i=1}^{D}\frac{x_i}{D})}\\ &= ne^{\frac{-n}{D}}.\end{aligned}$$

Thus, if $D \geq n$, the expected number of singletons is bounded above by $ne^{\frac{-n}{D}}$. □

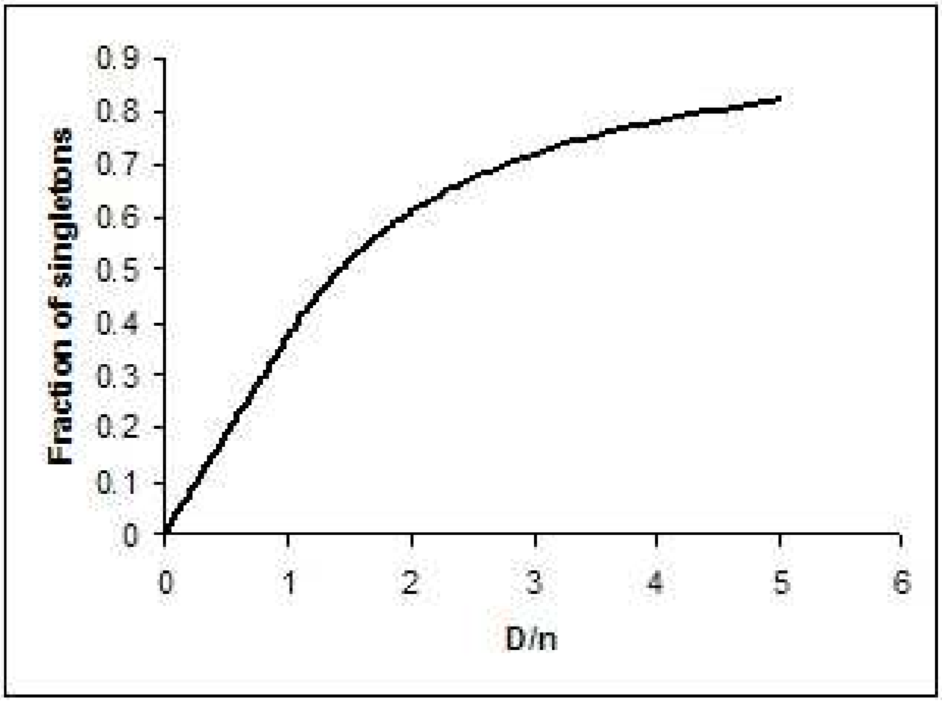


Figure 4.1: *Quasi-Identifier Test*

Figure 4.1 shows how the maximum expected fraction of singletons or unique rows in a collection of $n$ rows behaves, as the number of distinct values, $D$, varies. The graph plots the maximum expected fraction of unique rows as a function of $\frac{D}{n}$. It is the line $\frac{D}{en}$ for $\frac{D}{n} \leq 1$ according to Theorem 4.2.1. For $\frac{D}{n} \geq 1$, it is the curve $e^{\frac{-n}{D}}$

[2]If $f$ is a concave function, and $\sum_{i=1}^{m} p_i = 1$, with $p_i \geq 0\ \forall i$, then $\sum_{i=1}^{m} p_i f(x_i) \leq f(\sum_{i=1}^{m} p_i x_i)$.

according to Theorem 4.2.2. The curve is both continuous and smooth (differentiable) at $\frac{D}{n} = 1$ with $f(1) = \frac{1}{e}$ and $f^{'}(1) = \frac{1}{e}$.

Figure 4.1 forms a ready reference table in order to test whether a set of attributes forms a probable quasi-identifier. For example, if for a set of attributes $D < 3n$, then it is unlikely that the set of attributes will form a 0.75 quasi-identifier. If a set of attributes do not form an $\alpha$-quasi-identifier according to the the number of distinct values in Figure 4.1, then they almost certainly do not form an $\alpha$-quasi-identifier as the plot gives the maximum expected fraction of singletons (as per Theorem 4.2.1 and Theorem 4.2.2).

**Example 3** *We now show how (`Gender`, `Date of Birth`, `Zipcode`) forms a quasi-identifier when restricted to the U.S. population. The size of the U.S. population can be approximated as $3 * 10^8$, that is, the size of the universal table $n$ is $3 * 10^8$. The number of distinct values taken by the attribute set (`Gender`, `Date of Birth`, `Zipcode`) is $4 * 10^9$ from Example 2. Therefore, by Theorem 4.2.2, the maximum expected fraction of rows with singleton occurrence is $e^{-3*10^8/4*10^9} = e^{-0.075} \approx 0.93$. Thus, (`Gender`, `Date of Birth`, `Zipcode`) is a potential 0.93 quasi-identifier. Please recall that this combination is already known to be a 0.87 quasi-identifier [Swe02b].*

**Example 4** *We now give an example of a set of attributes that does not form a quasi-identifier. Let us consider (`Nationality`, `Date of Birth`, `Occupation`). The number of distinct values along these columns is given from Example 2 as $D = 4*10^8$. Here the size of the universal table is $n = 6*10^9$, that is, equal to the world population. Since $D < n$, we use Theorem 4.2.1 and find that the expected fraction of rows with singleton occurrence is bounded above by $D/en = 4 * 10^8/2.7 * 6 * 10^9 \approx 0.025$. Thus these columns almost certainly do not form even a 0.05 quasi-identifier as 0.025 is an upper bound on the expected fraction of singletons over all possible probability distributions over quasi-identifier values.*

We now provide a simple test to decide whether a combination of attributes forms a potentially dangerous quasi-identifier, that is, say $\alpha \geq 0.5$.

**Theorem 4.2.3** *Given a universe of size* $n$*, a set of attributes can form an* $\alpha$*-quasi-identifier (where* $0.5 \leq \alpha < 1$*) if the number of distinct values along the columns,* $D > \frac{n}{ln(1/\alpha)}$.

**Proof:** Note that $D > n$. If not, then, by Theorem 4.2.1, the maximum expected fraction of rows taking unique values is $D/en \leq 1/e < \alpha$.

From Theorem 4.2.2, the maximum expected fraction of rows taking unique values along the columns with $D$ distinct values is $e^{-n/D}$. For the the set of rows to form an $\alpha$-quasi-identifier, this fraction must be larger than $\alpha$. Thus, $e^{-n/D} > \alpha$, which implies that $D > \frac{n}{ln(1/\alpha)}$. □

### 4.2.1 Distinct Values and Quasi-Identifiers

In this section, we have provided an interesting connection between whether a set of columns forms a quasi-identifier and the number of distinct values assumed by the combination of the columns. The main contributions of this association are as follows.

1. We provide a fast and efficient technique to test whether a set of columns forms a quasi-identifier. However there may be false positives. A set of columns signaled as a probable $\alpha$ quasi-identifier may only be a $\beta$ quasi-identifier for some $\beta < \alpha$.

2. We do not assume anything about the distribution on the values taken by the quasi-identifier. The expected number of singletons is bounded by the expression provided in this section for all possible distributions over the values taken by the quasi-identifier.

3. When a set of columns is declared not to be a quasi-identifier by the test in this section, the set of columns is almost certainly not a quasi-identifier, that is, there is a minuscule chance of false negatives.

## 4.3 Probabilistic Anonymity

In Sweeney's anonymity model [Swe02b], every row of the dataset is required to be identical with $k$ other rows in the dataset along $Q$. In the following notion of anonymity, we insist that each row of the anonymized dataset should match with at least $k$ or more rows of the universal table $\mathcal{U}$ along $Q$. Since $\mathcal{U}$ is represented in a probabilistic fashion, we want this event to happen with high probability.

**Definition 4.4** *A dataset is said to be probabilistically* $(1-\beta, k)$*- anonymized along a quasi-identifier set* $Q$*, if each row matches with at least* $k$ *rows in the universal table* $\mathcal{U}$ *along* $Q$ *with probability greater than* $(1-\beta)$.

Our notion of anonymity is similar to that of [Swe02b] for an adversary who is *oblivious*, that is, she is not really looking for some *particular* individuals, but is trying to do a join on $Q$ and checking if she is "lucky". This kind of attack is quite a possibility in today's outsourcing scenarios where in an attacker, say, from a call center, would want to know identities in her client's data without really knowing whom to look for. If an adversary is looking for a *particular* individual in the anonymized dataset, then Sweeney's model would generally provide better privacy than our model for it would always yield $k$ matches. For our model to work well against such an adversary, we need to declare the original dataset itself as the universal table $\mathcal{U}$ and carry out anonymization.

In what follows, we build on the strong connection between the number of distinct values assumed by a set of attributes $Q$ and its identity revealing potential that was discovered in Section 4.2. Intuitively, it is clear from Theorems 4.2.1, 4.2.2 and 4.2.3 that the potency of $Q$ as a quasi-identifier would decrease if we reduce the number of distinct values assumed by $Q$. This is to be done with appropriate *generalization.* We borrow the following definition of generalization from [Swe02b] which has an excellent discussion on this topic.

**Definition 4.5** *Generalization involves replacing (or recoding) a value with a less specific but semantically consistent value.*

**Example 5** *The original ZIP codes {02138, 02139} can be generalized to 0213*, thereby stripping the rightmost digit and semantically indicating a larger geographical area.*

One way of looking at generalization is creating $<< D$ partitions of the space of $D$ distinct values and choosing a representative for each partition. In fact, it would give us $k$-anonymity if we could ensure that most of these partitions are represented by $k$ or more of their own members in the universal table $\mathcal{U}$ with high probability. To make this work, let us suppose that we have got a $D'$-partition of original $D$ size space such that each partition has probability $1/D'$ (or $O(1/D')$ to be precise). Given a $< p_1, p_2, \ldots, p_D >$ probabilities of the original $D$ size space, such partitioning is certainly possible using techniques we show in Section 4.4 for a single dimension. Now, we analyze below the bound on $D'$ that is necessary is order to ensure that most of these partitions are represented $k$ or more times in $\mathcal{U}$ with high probability. Please recall that $\mathcal{U}$ has size $n$ and it is built by sampling with replacement.

**Theorem 4.3.1** *A data set is probabilistically $(1-\beta, k)$-anonymized with respect to a universal table $\mathcal{U}$ of size $n$ along the quasi-identifier $Q$ if the number of distinct values along $Q$, $D' < \frac{n}{k}(1-c)$ for some small constant $c$.*

Before we proceed with the proof, please note that Theorem 4.3.1 provides a recommendation for $D'$, the number of partitions of $D$ size space of $Q$. If the probabilities $< p_1, p_2, \ldots, p_D >$ are known, then as per our earlier assumption, one could cluster these probabilities such that $D'$ equi-probable partitions are created. This concretizes generalization which could be used by any data-holder for anoymizing its data before release.

**Proof:** Let us suppose that we have got a $D'$-partition of original $D$ size space of quasi-identifier $Q$ such that each partition has probability $1/D'$. Let $X_i$ denote the indicator variable if $\geq k$ rows in the universal table $\mathcal{U}$ are chosen from the $i^{\text{th}}$

partition.

$$\begin{aligned}
P[X_i = 1] &= \sum_{j=k}^{n} \binom{n}{j} (\frac{1}{D'})^j (1 - \frac{1}{D'})^{n-j} \\
&= 1 - \sum_{j=0}^{k-1} \binom{n}{j} (\frac{1}{D'})^j (1 - \frac{1}{D'})^{n-j} \\
&\geq 1 - exp(\frac{-D'(n/D' - (k-1))^2}{2n}) \\
&\quad \text{(by Chernoff bounds [Che52])} \\
&= 1 - exp(\frac{-(n-(k-1)D')^2}{2nD'}).
\end{aligned}$$

For $1-\beta$ probability guarantee, we would like to have

$$1 - exp(\frac{-(n-(k-1)D')^2}{2nD'}) \geq 1 - \beta,$$

that is,

$$\frac{-(n-(k-1)D')^2}{2nD'} \leq ln\beta.$$

This is true when,

$$0 \leq D'^2 + \frac{2nD'}{k-1}\left(\frac{ln\beta}{k-1} - 1\right) + \left(\frac{n}{k-1}\right)^2,$$

that is,

$$D' \leq \frac{n}{k-1}(1 + x - \sqrt{x^2 + 2x}),$$

where

$$x = \frac{-ln\beta}{k-1}.$$

This implies that

$$D' \leq \frac{n}{k}(1-c)$$

is sufficient for some small constant $c$. □

**Example 6** *Let $\mathcal{U}$ be the U.S. Census Table of size $n = 3 * 10^8$. Consider the* *columns $Q$ = (`Gender`, `Date of Birth`, `Zipcode`). By Example 2, $D = 4 * 10^9$. According to Theorem 4.3.1, a dataset is $(0.9, 100)$ anonymized along $Q$ with respect to $\mathcal{U}$ if we make $D'$ partitions (or generalizations) of the $D$ size space where*

$$D' \leq \frac{n}{125} = 2.4 * 10^6.$$

*Thus, we have to reduce the number of possibilities for $Q$ by a factor of $D/D' < 1700$. Consider the following generalization (`Gender`, `Half-year of Birth`, `First Four Digits of Zipcode`). Now $D' = d'_1 * d'_2 * d'_3$. $d'_1$, the number of distinct values of `Gender`, is 2. $d'_2$ is $60 * 2 = 120$, and $d'_3 = 10^4$. Therefore, $D' = 2.4 * 10^6$. This should be good enough to make each row 100-anonymous with probability at least 0.9.*

## 4.3.1 Privacy vs Utility

Note that (`Gender`, `Half-year of Birth`, `First Four Digits of Zipcode`) was just one of many different ways we could have compressed the $D$ size space in Example 6 by factor 1700. Ideally, we would like to devise this generalization such that there is little or no loss in the *data utility.* We frame this problem as an optimization problem below where the goal is to retain maximum utility given privacy constraints.

Let there be $m$ columns $< C_1, C_2, \ldots, C_m >$ that need generalization and $w_1, w_2, \ldots, w_m$ be their respective weights giving their relative importance. We aim to anonymize this multi-column database so that maximum utility is retained in the probabilistically $k$-anonymized output.

Let $d'_1, d'_2, \ldots, d'_m$ be the number of distinct values along columns $C_1, C_2, \ldots, C_m$ after probabilistic $k$-anonymization. Then, by Theorem 4.3.1,

$$\prod_{i=1}^{m} d'_i \;=\; \frac{n}{k}(1-c) = D'.$$

Let us suppose that the quantile based anonymization from Section 4.4 is used. Thus, $d'_i$ different quantiles are used along the column $C_i$. Then, the rank difference of the transformation (from Section 4.4) is approximately $(\frac{n}{d'_i})^2 \times d'_i = \frac{n^2}{d'_i}$.

The sum of the distortion along all columns weighted by the column weights is, therefore, $n^2(\sum_{i=1}^{m} \frac{w_i}{d'_i})$. Minimizing this is equivalent to minimizing $\sum_{i=1}^{m} \frac{w_i}{d'_i}$ subject to $\prod_{i=1}^{m} d'_i = D'$. For a fixed value of product, the sum of numbers is minimized when all the numbers are equal. Therefore,

$$\frac{w_1}{d'_1} = \frac{w_2}{d'_2} = \ldots \frac{w_m}{d'_m} = \frac{1}{d} \text{ (say)}.$$

Therefore, $d'_i = d \times w_i \; \forall 1 \leq i \leq m$. The product condition implies, $\prod_{i=1}^{m} d'_i = d^m \prod_{i=1}^{m} w_i = D'$. Therefore,

$$d = (\frac{D'}{\prod_{i=1}^{m} w_i})^{1/m},$$

$$d'_i \;=\; (\frac{D'}{\prod_{i=1}^{m} w_i})^{1/m} \times w_i. \tag{4.1}$$

Note that if $d'_i$ is less than the number of distinct values in column $i$ initially, say $d_i$, it suggests applying an approach like quantiles proposed here on column $C_i$. If $d'_i$ is greater than the number of distinct values in column $C_i$ initially, say $d_i$, then the column $C_i$ is left untouched. The number of distinct elements for other columns can be recalculated (and increased) after this. That is, if $d'_i > d_i$, then the optimization problem over all other variables is first solved after column $C_i$ is eliminated, i.e. Maximize $\sum_{j=1,j\neq i}^{m} \frac{w_j}{d'_j}$ subject to $\prod_{j=1,j\neq i}^{m} d'_j = D'/d_i$.

**Example 7** *Suppose that we want to probabilistically* $(0.9, 100)$*-anonymize a dataset with* 3 *columns (`Gender`, `Date of Birth`, `Zipcode`) and all columns are equally important, that is , they have equal weight.*

*As worked out in Example 9, each row is given* 100*-anonymity with probability at least* 0.9 *if* $D' = 2.4 * 10^6$*. As all* 3 *columns have equal weight, we get* $d'_1 = d'_2 = d'_3 \approx 133$*. However `Gender` has only* $2 < d'_1$ *values. This means we have to leave it untouched and work with the remaining two attributes. That gives* $d'_2 * d'_3 = 1.2 * 10^6$*. Since both the columns have equal weight, we get* $d'_2 = d'_3 \approx 1.1 * 10^3$*. As* $d'_2 = 1.1 * 10^3$ *is approximately* 60 *(years)*$*$12 *(number of months per year), `Date of Birth` is approximated to the month of birth. Also the number of distinct values of `Zipcode` being* $O(10^3)$ *implies that the last two digits of `Zipcode` are starred out. Thus* *the anonymization produced is (`Gender`, `Month of Birth`, `First Three Digits of Zipcode`).*

Note that this anonymization was entirely worked out in constant time in the above example. For general case, where the number of columns is $m$, it would require $O(m^2)$ time. Previous techniques to provide anonymity were not only $NP$-hard in the input size (that means it took exponential time in the dataset) [MW04, AFK$^+$05b] but even approximations required many passes over the database [AFK$^+$05b, AFK$^+$06]. [LDR05b] required passes to be exponential in the number of columns to be anonymized as the lattice developed there took exponential time to be built.

**Example 8** *According to HIPAA [HIP], each person must be anonymized in a crowd of $k = 20,000 = 2 * 10^4$ people. Now, suppose we want to anonymize a medical records table with columns (`Gender`, `Age` (In Years), `Zipcode`, `Disease`).*

*As always, the U.S. Census Table is the universal table $\mathcal{U}$ with $n = 3 * 10^8$ rows. The quasi-identifier is (`Gender`, `Age` (In Years), `Zipcode`). As the number of distinct values of `Gender` and `Age` are 2 and 100 respectively, the number of distinct values of `Zipcode` allowed is approximately $3 * 10^8/((2 * 10^4) * 2 * 100) = 75$ by Theorem 4.3.1. Therefore, `Zipcode` must be anonymized to its first two digits and should only indicate the State.*

### 4.3.2 The Curse of Dimensionality

As the number of dimensions (columns) increase, the number of distinct values per column on anonymization decrease rapidly. For example, consider a database table with 25 columns. The aim is to anonymize the table so that 10-anonymity is achieved for the U.S. population of size $3 * 10^8$. Further suppose that all the columns are given equal weight (importance). Applying Theorem 4.3.1 and the Multiple Domain Assumption, the number of distinct values per column can be obtained to be roughly 2. Thus all values in a column are generalized to two intervals or converted to two types of values. This hints at reduced data utility measured by any reasonable metric.

This phenomenon was also observed as the curse of dimensionality on $k$-anonymity [Agg05]. However, we must notice that the previous analysis should only be applied to columns that are available publicly. For example, in the Adults database [BM98], columns `capgain`, `caploss`, `fnlwgt` and `income` can be assumed to be sensitive columns that are present only in the database itself and are not available for an external join.

### 4.3.3 Distinct Values and Anonymity

In this section, we have provided an interesting connection between the number of distinct values taken by a combination of columns and the anonymity it can offer. The main contributions of this association are as follows.

1. This association between distinct values and anonymity guarantee results in an easy technique to obtain a $k$-anonymized dataset. Merge similar distinct values taken by a column so that the number of distinct values assumed by the column is reduced. The appropriate reduction in the number of distinct values leads to the conversion of a quasi-identifier into $k$-anonymous columns. As explained in Section 4.3.1, this would also help retain much of data utility since it minimally distorts ranks. We shall discuss this angle in more detail in the next section.

2. It also helps in coming up with the right kind of generalization for publicly known attributes so that published database can conform to laws like HIPAA.

## 4.4 1-dimensional Anonymity

The results of Section 4.3 provide us with the right amount of generalization for each publicly known attribute in order to achieve probabilistic $k$-anonymity for the entire $m$ column dataset. From any particular attribute point of view, the suggested generalization tries to create appropriate number of buckets (or partitions) in its distinct values space so that each bucket has $k' \gg k$ individuals from the universal table $\mathcal{U}$. Thus, in nutshell, there are $m$ 1-dimensional Sweeney's $k$-anonymity problems, of course, each with different value of $k$. Before we proceed further, we will like the reader to take a note of this strong underlying connection between our notion of probabilistic $k$-anonymity and Sweeney's notion of $k$-anonymity.

Now $k$-anonymity for multiple columns is known to be NP-hard [MW04, AFK$^+$05b, LDR05b]. Thankfully we found that this is not the case for a single column. In the remainder of this section, we showcase various algorithms that help achieve 1-dimensional $k$-anonymity while retaining maximum possible data utility.

### 4.4.1 Numerical Attributes

We start out with algorithms for numerical attributes. Note that they are also applicable to attributes of type date and `Zipcode`.

**Definition 4.6 $k$-Anonymous Transformation** *A $k$-anonymous transformation is a function, $f$, from $S = \{s_1, s_2, \ldots s_n\}$ to $S$ such that $\forall s_j : |\{f^{-1}(s_j)\}| \geq k$ or $|\{f^{-1}(s_j)\}| = 0$, that is, at least $k$ elements are mapped to each element (which has some element mapped to it) in the range.*

**Example 9** *Consider $S = \{1, 12, 4, 7, 3\}$, and a function $f$ given by $f(1) = 3, f(3) = 3, f(4) = 3, f(7) = 7$ and $f(12) = 7$. Then $f$ is a 2-anonymous transformation.*

#### Dynamic Programming

Our goal is to find a $k$-anonymous transformation that minimizes, say, the maximum cluster size amongst all clusters [Vaz04], or the sum of distances to the cluster centers [JV99], or the sum over all clusters the radius of the cluster times the number of points in the cluster [AFK$^+$06]. All these problems are known to be NP-hard for a general metric space. However, for points in a single dimension, we showcase an optimal polynomial time algorithm based on dynamic programming. The details of the algorithm can be found below.

If not already sorted, first sort the input and suppose that it is $p_1 < p_2 < \ldots < p_n$. For $1 \leq a < b \leq n$, let Cluster$(a, b)$ be the cost to cluster elements $p_a, \ldots, p_b$.

Consider the optimal clustering of the input points. Note that each cluster in the optimal clustering contains a set of contiguous elements. Moreover, each cluster is of size at least $k$ by the $k$-anonymity requirement. Since any cluster of size $\geq 2k$ can be broken into two contiguous clusters of size at least $k$ each and that would reduce the clustering cost, the size of a cluster in the optimal clustering will be at most $2k-1$.

The optimal clustering of the $n$ input points is, therefore, the optimal clustering of points $p_1, p_2, p_{n-i}$ and one single cluster of the points $(p_{n-i+1}, \ldots, p_n)$, where $i$ is the size of the last cluster. Note that $k \leq i < 2k$ by the previous analysis. Therefore we find the optimal clustering by trying out all possible values of $i \in \{k, k+1, \ldots, 2k-1\}$. Now, the dynamic programming recursive equation is given by

$$\text{ClusterCost}(1, n) = min_{k \leq i < 2k}\ \text{Cost}(\text{ClusterCost}(1, n-i), \text{Cluster}(n-i+1, n)).$$

Here Cost$(A, B)$ is the sum for a metric like the $k$-median [JV99] or cellular [AFK$^+$06] metric which minimizes the sum of costs over all clusters. It is the maximum function for the $k$-center metric [Vaz04] which minimizes the maximum of cluster sizes amongst all clusters.

ClusterCost$(a, b)$ is initially set to $\infty$ if $b - a + 1 < k$. For $b - a + 1 \geq k$, ClusterCost$(a, b)$ is initially set to the cost of clubbing all points into a single cluster, that is, Cluster$(a, b)$.

**Time Complexity** This algorithm needs input in the sorted order. Therefore, its time complexity has two components: 1. Time taken for sorting the input, and 2. time required for the dynamic programming. For input of size $n$ points, sorting takes $O(n \log n)$ time. The dynamic programming part requires time $O(nk)$ as evaluating ClusterCost$(1 \ldots i)$ takes $O(k)$ time for each $i$. Thus, overall time complexity is $O(n(k + \log n))$.

### Quantiles

The algorithm from previous section requires sorting of the input. For large $n$, this would entail external sort. It is not very desirable in practice. In this section, we explore efficient algorithms that cluster the data in time required to make 1 or 2 sequential passes over the data and use very little extra memory.

**Definition 4.7 Rank** *Given a set of distinct elements $S = \{s_1, s_2, \ldots, s_n\}$, the rank of an element $s_i$ is $r$ if $s_i$ is the $r^{th}$ largest element in the set.*

For a multi-set containing duplicates, different occurrences of the same element are given consecutive ranks.

**Example 10** *Among elements* $S = \{1, 12, 4, 7, 3\}$, $7$ *has rank* $4$, *while* $3$ *has rank* $2$.

**Definition 4.8 Rank difference of a transformation** *Given a set* $S = \{s_1, s_2, \ldots, s_n\}$ *of* $n$ *numbers, and a* $k$*-anonymous transformation* $f$, *let* $\pi(s_i)$ *represent the rank of element* $s_i$. *Then, the rank difference incurred by* $s_i$ *under the transformation* $f$ *is defined as* $|\pi(f(s_i)) - \pi(s_i)|$. *The rank difference of the transformation* $f$ *is the sum of rank difference over all elements, that is,* $\sum_{i=1}^{n} |\pi(f(s_i)) - \pi(s_i)|$.

**Example 11** *For set* $S = \{1, 12, 4, 7, 3\}$, $\pi(1) = 1$, $\pi(12) = 5$, $\pi(4) = 3$, $\pi(7) = 4$ *and* $\pi(3) = 2$. *For* $f$ *from Example 9,* $\pi(f(1)) = 2$, $\pi(f(12)) = 4$, $\pi(f(4)) = 2$, $\pi(f(7)) = 4$, *and* $pi(f(3)) = 2$. *The rank difference of this transformation is* $3$.

**Definition 4.9 Quantile Transformation** *Suppose that* $n = qk + r$, *where* $0 \leq r < k$. *Then, the quantile transformation is a* $k$*-anonymous transformation that partitions the elements into* $q$ *contiguous groups of size* $(k + \lfloor r/q \rfloor)$ *or* $(k + \lceil r/q \rceil)$ *each. All elements in a group are mapped to the median element of the group.*

**Theorem 4.4.1** *The quantile transformation has the minimum rank difference among all* $k$ *anonymous transformations.*

**Proof:** The proof is by a simple greedy argument. □

### Efficient Approximate Quantiles using Samples

It is possible to implement the exact quantile transformation. But finding the exact median(quantile) in $p$ passes over the data requires $n^{1/p}$ memory [MP78]. Thus, to get the exact quantile transformation in 2 passes, would require $\Omega(\sqrt{n})$ memory.

For those who work with smaller memory and/or look for something easier to implement, we sketch a sampling based approach here. We maintain a uniform sample of size $s = \frac{1}{\epsilon^2} log(\frac{1}{\delta})$ using Vitter's sampling technique [Vit85]. The rank $t$ element in the original set is approximated by the rank $st/n$ element in the sample, where $n$ is the size of the original dataset over which the sample is maintained. This element has rank between $t - (\epsilon n)$ and $t + (\epsilon n)$ in the original data with probability greater than $(1 - \delta)$ if the sample size $s$ is chosen as given above [MRL99]. For example suppose that we maintain a uniform sample of 100 elements out of a total $100,000$ elements. Then the $5,000$th element in sorted order among the $100,000$ elements can be approximated well by the 5th element in sorted order from amongst the sample of 100 elements.

### 4.4.2 Categorical Attributes

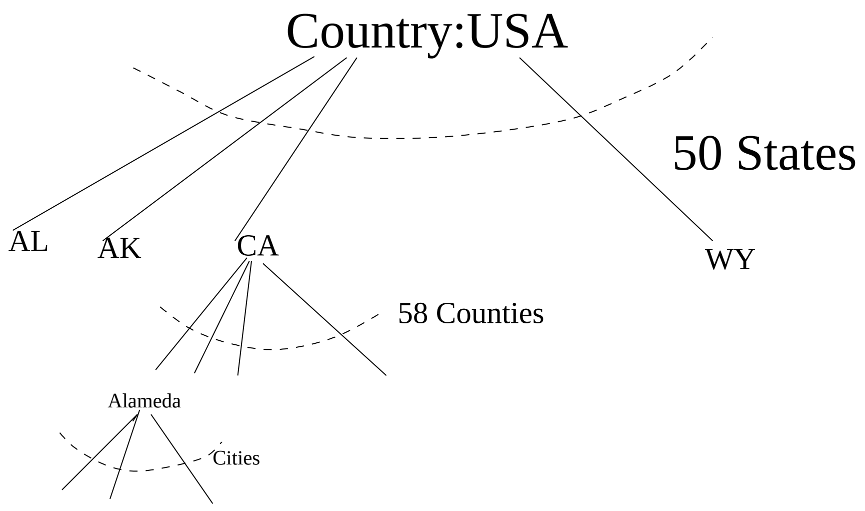


Figure 4.2: *A Categorical Attribute*

In the previous sub-section, we discussed how to create appropriate buckets or categories for numerical (ordered) attributes. But many a times, there is an attribute with no intrinsic ordering among its value-set. Such an attribute is called as a *categorical attribute*

For categorical attributes we create a layered tree graph as explained. The first layer consists of a node for each category value. The next layer groups together nodes that generalize into one general categorical value, so that they form a single node.

This is set to be the parent of the generalized values. This is repeated till there is a single category. Consider for example location information shown in Figure 4.2. Zipcodes are generalized to cities which are generalized to counties to state and finally to country. The top three levels of the generalization hierarchy are shown. To anonymize this dataset so that there are $d$ distinct values, the generalization is carried till the level that there are $d$ values. For example, to generalize location so that there are 50 different values, the state information would be retained. However to generalize it to 3000 distinct values, the county information would be retained.

## 4.5 Experiments

### 4.5.1 Quasi-Identifiers

We counted the number of singletons in the Adult Database available from the UCI machine learning repository [BM98]. The Adult Database has got 32561 rows with 15 attributes, we considered 10 of them and dropped the remaining 5. The dropped attributes are sensitive attributes (not quasi-identifiers): `fnlwgt`, `capgain`, `caploss`, `income` and the attribute `edunum` which is equivalent to the attribute education. In our experiments, we varied the size of the attribute set $Q$ under consideration from 1 to the maximum of 10. The table in Figure 4.3 shows some of the results that we obtained.

Labels **A1**, **A2**, ..., **A10** denote the 10 columns of the table. The first row gives the number of distinct values each attribute **A1**, **A2**, ..., **A10** takes. All other rows (which are labeled with row numbers from 1 to 12) of the table represent publishing the projection of the table along the columns marked ‘x’. For example, the row 1 represents publishing the database projected on the `Age` (**A1**) column while the row 12 represents publishing all 10 columns in the database. The column **Size** gives the number of ‘x’ marks in each row, that is, the number of columns that constitute the quasi-identifier $Q$ under consideration.

The column **S** is the number of rows uniquely identified by the projection of these columns, that is, the number of rows uniquely identified in the published projection.

| **Row** | **Size** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** | **A8** | **A9** | **A10** | **S** | $\mathbf{F_1}$ | **D** | $\mathbf{F_2}$ | **k-Anon** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 60 | 8 | 15 | 7 | 14 | 6 | 5 | 2 | 20 | 40 | | | | | |
| 1 | 1 | x | | | | | | | | | | 2 | $6.1 * 10^{-5}$ | 60 | $7.4 * 10^{-8}$ | $5 * 10^{6}$ |
| 2 | 2 | x | | | | | | | | x | | 986 | 0.03 | 1200 | $1.48 * 10^{-6}$ | $2.5 * 10^{5}$ |
| 3 | 3 | x | | | | | | x | x | | | 65 | 0.002 | 600 | $7.4 * 10^{-7}$ | $5 * 10^{5}$ |
| 4 | 4 | x | x | x | | x | | | | | | 5056 | 0.16 | $1 * 10^{5}$ | $1.2 * 10^{-4}$ | $3 * 10^{3}$ |
| 5 | 4 | x | x | | | x | | | | | x | 3105 | 0.095 | $2.7 * 10^{5}$ | $3.3 * 10^{-4}$ | $1.1 * 10^{3}$ |
| 6 | 4 | x | | | | x | | | | x | x | 7581 | 0.23 | $6.7 * 10^{5}$ | $8.3 * 10^{-4}$ | 450 |
| 7 | 4 | | x | x | | x | | | | | x | 1384 | 0.043 | $6.7 * 10^{4}$ | $8.3 * 10^{-5}$ | $4.5 * 10^{3}$ |
| 8 | 5 | x | x | x | | x | | | | | x | 7659 | 0.235 | $4 * 10^{6}$ | $4.9 * 10^{-3}$ | 75 |
| 9 | 5 | x | x | | x | x | x | | | | | 5215 | 0.16 | $2.8 * 10^{5}$ | $3.4 * 10^{-4}$ | $1 * 10^{3}$ |
| 10 | 5 | x | x | | | x | x | | | x | | 12870 | 0.40 | $8 * 10^{5}$ | $9.9 * 10^{-4}$ | 380 |
| 11 | 5 | x | x | | | x | | | | x | x | 10402 | 0.32 | $5.4 * 10^{6}$ | $6.7 * 10^{-3}$ | 55 |
| 12 | 10 | x | x | x | x | x | x | x | x | x | x | 24802 | 0.76 | $33 * 10^{9}$ | 0.99 | 1 |

Size = Number of columns that make the quasi-identifier, A1 = Age, A2 = Work class, A3 = Education, A4 = Marital status, A5 = Occupation, A6 = Relationship, A7 = Race, A8 = Sex, A9 = Hours per week, A10 = Native country, $S$ = Number of singletons in the current table, $F_1$= Fraction of singletons using the table itself = S/32561, $F_2$=Fraction of singletons using Figure 4.1 and $n = 3 * 10^8$ for US population, k-Anon= Anonymity parameter for the published database = $n/D$.

Figure 4.3: *Quasi-Identifiers on the Adult Dataset*

For example, for row 2, where **A1** and **A9** are the attributes of projection, $\mathbf{S} = 986$ is returned by the following SQL statement in MS Access:

```
SELECT A1, A9 FROM T
GROUP BY A1, A9
HAVING count(*)=1
```

$\mathbf{F_1}$ is the fraction of rows uniquely identified, given by $\mathbf{S}/32561$ where $\mathbf{S}$ is the number of singletons while 32561 represents the total number of rows in the database table. For row 2, $\mathbf{F_1} = 0.03$. Some previous definitions of quasi-identifiers [XM06] measured a quasi-identifier as a set of columns that have a large fraction of unique rows. Thus, $\mathbf{F_1}$ is used as a measure of quasiness. This does not model the external table present with the adversary. For example, by this definition, **A1** and **A9** would together be a 0.03-quasi-identifier.

$\mathbf{D}$ is the product of the domain sizes of the attributes marked 'x' in the row. By Multiple Domain Assumption, it is the size of the distinct values space for that combination of columns. For example, for row 3, $\mathbf{D} = 60 * 5 * 2 = 600$.

$\mathbf{F_2}$ captures the notion of quasiness as proposed in Section 4.2. It is given by $f(D/n)$ shown in Figure 4.1. Here, $D$ is set to be equal to the value from column $\mathbf{D}$, and $n = 3 * 10^8$, the size of US population. Please recall that, by Theorems 4.2.1 and 4.2.2, $f(D/n) = D/en$ for $D < n$ and $e^{-n/D}$ for $D \geq n$. For all but the last row of the table, $\mathbf{D} < 3 * 10^8$, hence $\mathbf{F_2} = \frac{\mathbf{D}}{2.7*3*10^8}$, for the last row $\mathbf{F_2} = e^{-3*10^8/\mathbf{D}}$.

**k-Anon** is approximately the probabilistic $k$-anonymity obtained from the published database. Based on the result of Theorem 4.3.1, it is set to $n/\mathbf{D}$, where $n = 3 * 10^8$, the size of the US population. When $\mathbf{D}$ exceeds n, it is set to 1.

Suppose we are allowed to publish a set of columns with the condition that all 0.2-quasi-identifiers are to be suppressed. If we only consider the entries of the table and look at those projections where at least 0.2 fraction of the rows are unique, then the projections indicated by rows numbered $6, 8, 10, 11$ and 12 cannot be published. This is because their $\mathbf{F_1}$ values exceed 0.2.

In fact, our real worry is that $> 0.2$ fraction of the rows should not get uniquely identified after taking an external join with the universal table $\mathcal{U}$. Then, only row 12 qualifies as a possible 0.2-quasi-identifier as only its $\mathbf{F_2}$ value exceeds 0.2. Note that, from Theorems 4.2.1 and 4.2.2, there is a minuscule chance of false negatives, that is, rows $1 - 11$ are unlikely to be 0.2-quasi-identifiers.

Row 12 needs a closer look since 0.99 is only an upper bound on the expected fraction of unique rows. It may be noticed that many combinations are rare and do not occur. In our example, two attributes **A9** and **A10** are special. **A9** may be represented with only 5 distinct values since the exact hours per week of an individual may not be known and **A10** is not uniformly distributed. Such a case by case analysis of the different attributes may bring down the distinct values, **D**, and hence the fraction of distinct rows. Thus, it can help improve the estimate of quasiness, say, from a 0.99 fraction to (probably) a fraction lower than 0.2. In such a case, row 12 would be a false positive.

### 4.5.2 Anonymity Algorithms

We implemented sampling based approximate quantile algorithm (from Section 4.4.1) as a technique in a commercial data masking tool. Our technique required 400 lines of code to be added to the tool. The tool was run on an Oracle database containing $250,000$ rows of a table from a real bank, which was a customer of the tool vendor. The database table was about 1GB in size and had 261 columns. We also repeated our experiments on the public use microdata sample (PUMS) [Bur] provided by the U.S. Census Bureau. This dataset was given in a flat file format as input to the data masking tool. The experiments were run on a machine with 2.66GHz processor and 504 MB of RAM running Microsoft Windows XP with Service Pack 2.

**Scaling with the Dataset Size**

We studied how the running time of the quantile algorithm for masking a single column changes as the number of rows in the database table is varied. We measured the time required to mask various fractions of the table, the entirety of which contains $250,000$ rows. The time required to mask this single numeric column with $k = 10,000$ anonymity (so that there are 25 different quantiles to which the data is approximated) increased linearly to a total of about 10 seconds for the entire column. A straight line with almost exactly identical slope and coordinates was obtained for the PUMS [Bur] dataset.

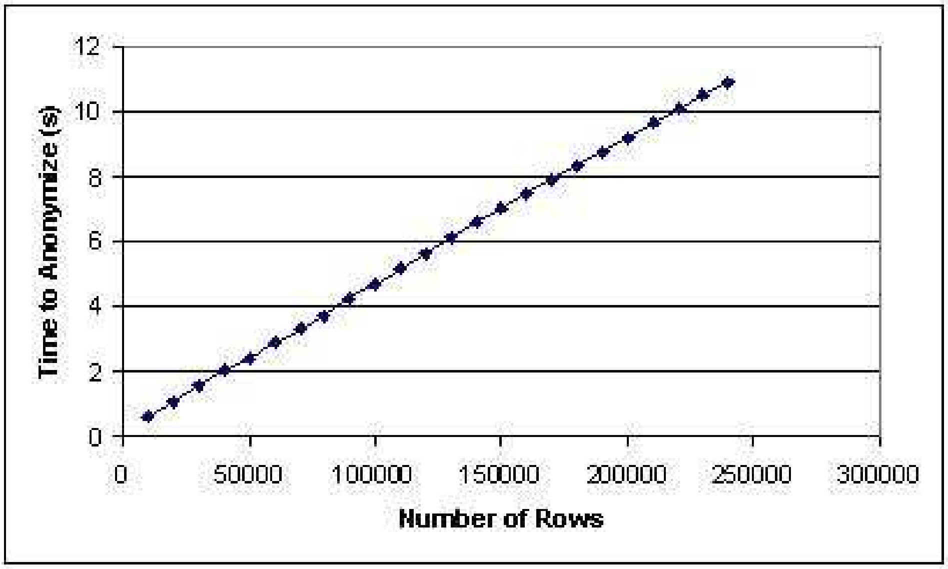


Figure 4.4: *Time taken for varying number of rows.*

**Scaling with the Number of Columns Masked**

We studied how the running time of the quantile algorithm for masking multiple columns varies as the number of columns to be masked is varied. For this experiment too, we used the table with $250,000$ rows and 261 columns. As each column is independently anonymized, the time taken increases linearly as the number of columns being anonymized increases. Previous algorithms [LDR05b] had an exponential increase in the time taken for anonymization as the number of columns increased as the lattice created was exponential in the number of columns being anonymized.

The time taken to anonymize 10 columns of data with $250,000$ rows was approximately 100 seconds. This is almost an order of magnitude improvement over the previous algorithm [LDR05b]. The results on the PUMS dataset were similar.

**Scaling with the Anonymity Parameter**

The implemented algorithm does a binary scan over all buckets to find the bucket closest to each data item. The time required to anonymize a data value, therefore, logarithmically increases as the number of buckets increases (or the value $k$ of anonymity

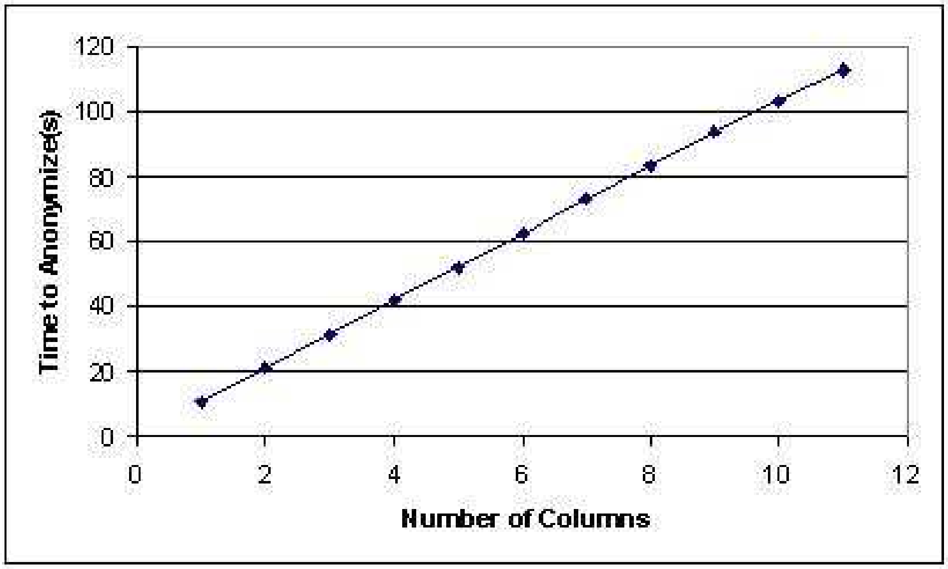


Figure 4.5: *Time taken for varying number of columns.*

parameter decreases). If $b$ is the number of buckets and $n$ the number of rows, then the time to anonymize is $nlog(b)$. The time taken to read $n$ rows from disk is $nC$ where C is a large constant. The total time taken is, therefore, $n(C + \log b)$ where $C \gg \log(b)$. This explains the shape of the curve in Figure 4.6. Here $nC \approx 10$ seconds and the $log(b)$ term explains the slight increase from 0 to 500 buckets.

**Tradeoff between Privacy and Utility**

We studied how the error introduced in a column as a result of $k$-anonymization varies with the anonymity parameter $k$. Let $x_i$ be the original value of the $i^{th}$ row. Let $x_i^{'}$ be its value after $k$-anonymization. Then $(x_i^{'} - x_i)^2$ is the error introduced for row $i$ as a result of $k$-anonymization. The total error introduced over $n$ rows is $Error = \sum_{i=1}^{n}(x_i^{'} - x_i)^2$. Let $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$. If all $x_i^{'}$ are constrained to be identical (corresponding to anonymity with a single bucket), then $\bar{x}$ gives the minimum error according to the above metric, i.e. it gives $MinError = Min_x \sum_{i=1}^{n}(x - x_i)^2 = \sum_{i=1}^{n}(\bar{x} - x_i)^2$. We, therefore, normalize the error as Error/MinError.

The curve is plotted in Figure 4.7 where the normalized error is plotted on the $y$-axis while the number of buckets, $b = \frac{n}{k}$, is plotted on the $x$-axis. An almost identical curve was obtained for the PUMS dataset. The curve very closely follows the curve

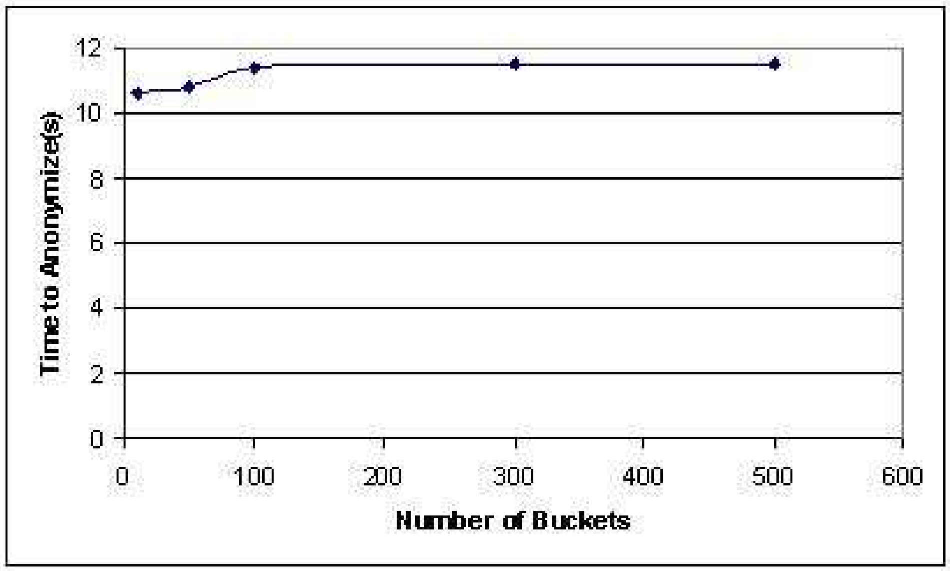


Figure 4.6: *Time taken for varying number of buckets.*

$\frac{1}{b^2}$. This could be proven analytically.

Thus, for given $n$ and $k$, we find that the identity disclosure risk is $< 1/k$ (for "join" class of attacks) and the error introduced in data is $\propto k^2/n^2$. We may, therefore, boldly quantify the privacy provided by $k$-anonymization as $p = 1-1/k$ and the utility retained as $u = 1 - k^2/n^2$ implying the following privacy-utility trade-off equation.

$$(1-p)^2(1-u) = 1/n^2 \text{ (a constant).}$$

Note that, the fact that we used sum square errors, instead of sums of absolute values of errors explains the square term above.

## 4.6 Related Work

One of the earliest definitions of quasi-identifier can be found in Dalenius [Dal86]. [Swe02b, Swe02a] and [LDR05b] use a similar definition.

Samarati and Sweeney formulated the $k$-anonymity framework and suggested

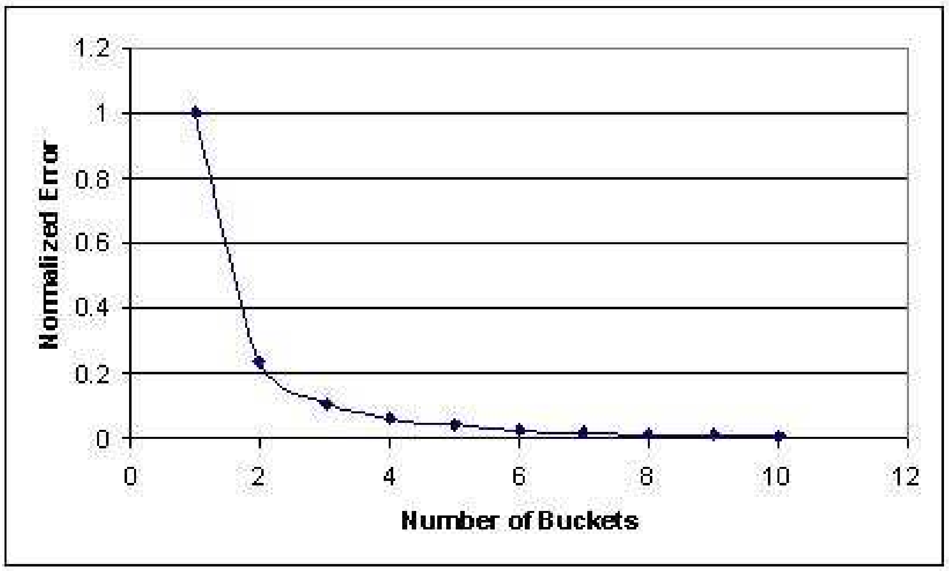


Figure 4.7: *Tradeoff between privacy and utility.*

mechanisms for $k$-anonymization using the ideas of generalization and suppression [SS98, Swe02b, Swe02a]. Subsequent work has shown some NP-hardness results [MW04, AFK+05a, AFK+06] and that has inspired many interesting heuristics and approximation algorithms [Iye02, Win02, MW04, BA05, AFK+05a, LDR05b, MKGV06, AFK+06]. All of this work assumes that quasi-identifier attribute sets are known based on specific knowledge domain.

The basic theme of $k$-anonymity model is to *hide* an individual in a crowd of size $k$ or more. A similar intuition is pursued by Chawla et al in [CDM+05] who, in fact, manage to convert it into a precise mathematical statement. They not only give definition of privacy and its compromise for statistical databases, but also provide a method for describing and comparing the privacy offered by specific sanitization techniques. They also give a formal definition of an *isolating* adversary whose goal is to single out someone from the crowd with the help of some *auxiliary* information $z$. This work is further extended in [CDMT05] where Chawla et al study privacy-preserving histogram transformations that provide substantial utility.

There is a wide consensus that privacy is a corporate responsibility [IBM]. In order to help and ensure corporations fulfill this responsibility, governments all over the world have passed multiple privacy acts and laws, for example, Gramm-Leach-Bliley (GLB)Act [GLB], Sarbanes-Oxley (SOX) Act [SOX], Health Insurance Portability and Accountability Act (HIPAA) [HIP] are some such well known U.S. privacy acts. In fact, HIPAA recommends the following *safe-harbor* method of de-identification in which it provides clear guidelines for sanitizing quasi-identifiers including date types, `Zipcode`, etc. For $20,000$ anonymity, HIPAA advises to retain essentially only the State information in `Zipcode` and year information in `Date of Birth` which is quite inline with what we concluded in Examples 6, 7 and 8 based on our analysis. The de-identification excerpt from the HIPAA law is provided in Appendix 4.8.

## 4.7 Conclusions

In this chapter, we provided the first formalism and a practical technique to identify a quasi-identifier. Along the way we discovered an interesting connection between whether a set of columns forms a quasi-identifier and the number of distinct values assumed by the combination of the columns.

Then we defined a new notion of anonymity called as probabilistic anonymity where in we insist that each row of the anonymized dataset should match with at least $k$ or more rows of the universal table $\mathcal{U}$ along a quasi-identifier. We observed that this new notion of anonymity is similar to the existent $k$-anonymity notion in terms of privacy guarantees and is sufficiently strong for many real life scenarios involving oblivious adversaries. Building on our earlier work, we found an interesting connection between the number of distinct values taken by a combination of columns and the anonymity it can offer. This allowed us to find an ideal amount of generalization or suppression to apply to different columns in order to achieve probabilistic anonymity. We worked through many examples and showed that our analysis can be used to make a published database conform to privacy acts like HIPAA.

In order to achieve the probabilistic anonymity, we observed that one needs to solve multiple 1-dimensional $k$-anonymity problems. We proposed many efficient and scalable algorithms for achieving 1-dimensional anonymity. Our algorithms are optimal in a sense that they minimally distort data and retain much of its utility.

## 4.8 De-identification required for HIPAA

*"The following identifiers of the individual or of relatives, employers, or household members of the individual must be removed to achieve the "safe harbor" method of de-identification: (A) Names; (B) All geographic subdivisions smaller than a State, including street address, city, county, precinct, zip code, and their equivalent geocodes, except for the initial three digits of a zip code if, according to the current publicly available data from the Bureau of Census (1) the geographic units formed by combining all zip codes with the same three initial digits contains more than 20,000 people; and (2) the initial three digits of a zip code for all such geographic units containing 20,000 or fewer people is changed to 000; (C) All elements of dates (except year) for dates directly related to the individual, including birth date, admission date, discharge date, date of death; and all ages over 89 and all elements of dates (including year) indicative of such age, except that such ages and elements may be aggregated into a single category of age 90 or older; (D) Telephone numbers; (E) Fax numbers; (F) Electronic mail addresses: (G) Social security numbers; (H) Medical record numbers; (I) Health plan beneficiary numbers; (J) Account numbers; (K) Certificate/license numbers; (L) Vehicle identifiers and serial numbers, including license plate numbers; (M) Device identifiers and serial numbers; (N) Web Universal Resource Locators (URLs); (O) Internet Protocol (IP) address numbers; (P) Biometric identifiers, including finger and voice prints; (Q) Full face photographic images and any comparable images; and (R) any other unique identifying number, characteristic, or code, except as permitted for re-identification purposes provided certain conditions are met. In addition to the removal of the above-stated identifiers, the covered entity may not have actual knowledge that the remaining information could be used alone or in combination with any other information to identify an individual who is subject of the information. 45 C.F.R. §164.514(b). "*

# Chapter 5

# Masketeer

The results in this chapter appear in [DLP+06].

## 5.1 Introduction

Advances in storage, networks, and hardware technology have resulted in an explosion of data and given rise to multiple sources of overlapping data. This, combined with general apathy towards privacy issues while designing systems and processes, leads to frequent breaches in personal identity and data security. What makes this worse is that many of these breaches are committed by the legitimate users of the data. Major countries like the U.S., Japan, Canada, Australia and EU have come up with strict data distribution laws which demand their organizations to implement proper data security measures that respect personal privacy and prohibit dissemination of raw data outside the country.

Since companies are not able to provide real data, they often resort to completely random data. It is obvious that such a data would offer complete privacy, but would have very low utility. This has serious implications for an IT services companies, since application development and testing environments rely on realistic test data to verify that the applications provide the functionality and reliability they were designed to deliver. It is always desirable that the test data is *similar* to, if not the same as, the production data. Hence, deploying proven tools that make de-identifying production data easy, meaningful and cost-effective is essential.

Data masking methods came into existence to permit the legitimate use of data and avoid misuse. In this Chapter, we consider various such techniques to be able to come up with a comprehensive solution for data privacy requirements. We present the data masking product MASKETEER™ (developed at TCS), which implements these techniques for providing maximum privacy for data while maintaining good utility.

## 5.2 Data Masking

Masking the production data is simply *the process of systematically removing or transforming data elements that could be used to gain additional knowledge about the sensitive information*. The objective of data masking is to maximize data utility in such a way that the masked data should have the same characteristics as the original data and at the same time minimize disclosure risks, that is, reduce the ability to identify an individual and reduce the ability to predict the value of confidential attributes.

The need for data masking is, in fact, ubiquitous. The contract software development is just one such striking example. Data masking plays a key role when a certain version of privately held data has to be made public. Here goal is to keep the identities of the individuals who are the subjects of the data secret, and yet allow the legitimate users to make perfect use of the released data. This problem is very common in the health sector and financial sectors.

### 5.2.1 Approaches

Over the years, statisticians, cryptographers and computer scientists have developed many models and techniques to address the trade-off between data privacy and its utility. We present here techniques that are robust, practical, and have simple quantification of privacy/utility. We explain them using Figures 5.1 and 5.2.

*Randomization*: In this approach, a data-element is replaced by a randomly chosen value from a given range or a dataset. The **Name** column of Figure 5.1 is replaced by randomly chosen names from a dataset of `English Names` in Figure 5.2. This

| **SSN** | **Name** | **Gender** | **Age** | **Zipcode** | **Balance** |
|---|---|---|---|---|---|
| 101 | Alice | F | 31 | 94305 | 100 |
| 102 | Bob | M | 31 | 94308 | 24 |
| 103 | Carol | F | 32 | 94308 | 35 |
| 104 | David | M | 22 | 94125 | 85 |
| 105 | Evelyn | F | 34 | 90428 | 12 |
| 106 | Frank | M | 18 | 92405 | 73 |
| 107 | George | M | 35 | 94308 | 57 |

Figure 5.1: *Original Database table*

| **SSN** | **Name** | **Gender** | **Age** | **Zipcode** | **Balance** |
|---|---|---|---|---|---|
| 501 | Jane | F | 31 | 9430* | 110 |
| 438 | Kurt | M | 31 | 9430* | 30 |
| 107 | Lance | M | 31 | 9430* | 45 |
| 745 | Molly | F | 20 | 94*** | 75 |
| 885 | Nancy | F | 34 | 9**** | 25 |
| 990 | Oscar | M | 20 | 94*** | 60 |
| 210 | Philip | M | 34 | 9**** | 52 |

Figure 5.2: *Masked Database table*

technique provides strong identity protection.

*Encryption*: In this approach, the data-element $X$ is replaced by its image $h(X)$ where $h$ is a suitable hash function. Ideally, one would want the hash function $h$ to be collision-free, non-idempotent and one-way. Unfortunately no such $h$ is provably known. But in practice, `MD5`, `SHA-1`, or Discrete Log based functions are useful. In the above example, encryption is applied to the **SSN** column. Encryption techniques are often efficient, but they provide low data utility by destroying semantics readily.

*Shuffling*: Shuffling randomly permutes the data-elements in a column. Thus, it can easily destroy relations between the columns. Shuffling is applied to the **Gender** column in the above example.

*Perturbation*: Another popular approach is to use perturbation techniques in order to hide the exact values, for example, adding noise to data and its numerous improvements. Here it is possible to capture the richness of data, say, with the covariance matrices . A simple perturbation technique could be addition of Gaussian noise to the input data. Let $X$ be the input column. Then, the resultant $Y$ would be $Y = X + e$, here $e$ is the Gaussian noise taken from a standard distribution. Perturbation is applied to the **Balance** column in the above example.

Perturbation techniques, capable of providing high data utility and low disclosure risk, may require some pre-processing of the data to yield parameter values. Otherwise, they are fairly efficient. They are not very suitable if one wants to draw inferences with 100% confidence.

*$k$-Anonymity*: De-identifying the data by masking key attributes like **SSN** and **Name** may not protect identities since linking such a masked database with a publicly available database on non-key attributes like **Gender**, **Date of Birth** and **Zipcode** can uniquely identify an individual [Swe02b].

A table provides $k$-anonymity [Swe02b] if any attempt to link the identifying columns by external joins results in $k$ or more matches. It means that each row in the table is forced to be same as at least $k - 1$ other rows in the potentially identifying attributes. Thus, identification of an individual by external join is with a probability of at most $1/k$. $k$-Anonymity is achieved by blocking all the dissimilar

| **Technique** | **Data Utility** | **Identity Disclosure Risk** | **Value Disclosure Risk** | **Scalability** |
|---|---|---|---|---|
| Randomization | Low | **Low** | **Low** | **High** |
| Encryption | Low | Medium | Medium | **High** |
| Shuffling | Medium | **Low** | High | Medium |
| Perturbation | Medium | **Low** | Medium | Medium |
| $k$-Anonymity | **High** | **Low** | High | Low |

Figure 5.3: *Techniques Overview*

values (suppression) or by replacing them with a less specific common consistent value (generalization). Hence $k$-anonymity is a trade off between data utility and privacy. A higher value of $k$ enforces a stricter privacy but more data loss. It is important to note that the data loss happens mainly for the potentially identifying attributes only, and the sensitive information (for example, say, medical condition of patients or cash balance) may appear as it is.

In the above example, we have applied 2-Anonymity to **Age** and **Zipcode**. The first three, the 4th and 6th, and the 5th and 7th rows in Figure 5.2 are identical in these two columns. Non identical values in **Age** are replaced by their average, while those in **Zipcode** are substituted by *. Hence anybody trying to link some known **Age** and **Zipcode** values with those in this table to find the **Balance** value would essentially be confused with *two* balance values. Thus, we are guaranteed 50% privacy.

$k$-Anonymity provides high data utility since it generalizes or suppresses only the quasi-identifiers. It also quantifies the identity disclosure risk at $1/k$. But the optimal $k$-anonymity is known to be NP-hard [AFK$^+$05a]. The known algorithms are either $O(k)$ approximations [AFK$^+$05a] or super-linear [AFK$^+$06] or require time exponential in the number of columns [LDR05b] thus making them inefficient or expensive. Our product uses the efficient algorithm outlined in [LT06].

A comparative overview of the different techniques is given in Figure 5.3. It is clear from this table that no single technique by itself can provide low disclosure risk, high data utility and work for high data volumes. But good news is that these techniques seem to complement each other. So their right combination may generate *good* data for us. A recent Forrester report [Yuh06] also advocates the same.

## 5.3 Constraints

We chose the above techniques because they are privacy-preserving and amenable to efficient implementations that scale well. Also appropriate choice of parameters for these techniques can help us retain much of the characteristics and patterns from the original data. In order to further improve the utility aspect in the masked data, we note that one has to account for the following database related considerations as well. They bring in serious engineering challenges.

*Syntactic Constraints*: The masked values should be within the limits specified in the database schema.

*Uniqueness Constraints*: For attributes that are marked as unique or as primary keys of the table, the masked values should be unique.

*Relational Integrity*: In order to support RDBMS, it is important to make sure that relational integrity constraints are satisfied, that is, the masked data-elements are propagated from the parent table to child tables.

*Business Constraints*: Much of the known business logic is encoded in company databases by linking multiple columns through some arithmetical or logical operations. The masked data should also satisfy these constraints.

## 5.4 MASKETEER™

There has been a lot of other products in the market that address a similar problem [Cam, Mas, Van, Sof]. Our data-masker software, is to the best of our knowledge, the one that provides all different masking techniques.

### 5.4.1 Key Features

MASKETEER™ is a platform independent, user friendly, highly extensible and easily portable data masking solution.

1. *Masking Techniques*: MASKETEER™ comes with the following different techniques providing mathematical robustness guarantee for masking purposes: randomization, encryption, shuffling, perturbation, and $k$- anonymity.

   If deemed necessary, users can code and easily plug-in their own masking techniques through its extensible framework.

2. *Compliance Assistance*: MASKETEER™ has proprietary algorithms to identify and optimally mask quasi-identifiers for the given database. This not only provides $k$-anonymity [Swe02b] and increased protection against identity thefts, but also helps in meeting HIPAA like compliance [HIP].

3. *Operational Modes*: MASKETEER™ is available in both client-server and a separate stand-alone application mode.

   In case of the client-server mode of operation, multiple users can connect to the MASKETEER™ server and carry out specific masking tasks remotely. Different roles and rights can be assigned to different users.

4. *Data Integrity*: Integrity constraints such as primary or foreign key and unique constraints are satisfied while masking the data. If referential integrity is not defined as a part of the database schema, user can specify it through the constraint editor. Similarly business constraints can also be specified on the masked data using another constraint editor.

5. *Enterprise Masking*: Most companies have overlapping data spread across different types of databases on various platforms. Often, they would need to mask the data in a consistent fashion even in such heterogeneous environs.

   MASKETEER™ allows users to specify cross database integrity constraints. It generates consistent masked data for all mentioned databases.

6. *Convenient Data Export*: MASKETEER™ does not modify the production data. In fact, it stores the masked data in flat files. These flat files can be

loaded into *any* target database. An export utility is also provided for moving the masked data from the flat files into the target database. Thus, shipping/uploading of masked data is made convenient.

7. *Masked Data Validation*: MASKETEER™ provides visual comparative validation of original and masked data. It also includes different statistical tests for assessing the quality of masking. In fact, users can define their own tests as well.

8. *Easy Customization* MASKETEER™ facilitates the plug-in of user defined masking techniques. The user has to just program the masking algorithm using the interface exposed by MASKETEER™ . The fetching and export of data is handled by MASKETEER™ itself. Similar interfaces to add user defined validation tests on masked data are provided.

9. *Robustness*: In case of failure, MASKETEER™ resumes the masking process from the point of failure. This feature is particularly useful while masking large size databases.

10. *Performance*: On an average, MASKETEER™ can mask 5GB/hour on a 2GHz Pentium machine having 512MB RAM.

11. *Database Support*: MASKETEER™ supports most of the RDBMS using, say, JDBC drivers for connectivity. For example, it supports DB2, MS Access, Oracle, MS SQL Server, Sybase, VSAM, ISAM, and also flat files. Its extensible framework allows users to plug-in other databases as well.

As a service offering, this product has already enabled more offshoring of IT projects. As a product, it has been used by major banks, insurance companies and health care providers not only for their IT engagements. It is helping these clients secure their test environments without compromising software quality. It is also playing a crucial role in their privacy compliance initiatives.

Evaluation copies of MASKETEER™ , are available at masketeer@tcs.com.

# Part II

# Auditing Query Logs for Privacy Compliance

# Chapter 6

# Auditing Batches of SQL Queries

The results in this chapter appear in [MNT07].

## 6.1 Introduction

Auditing is the process of inspecting past actions to determine whether they were in conformance with official policies. In the context of database systems with data disclosure policies, auditing queries is the process of inspecting queries that have been answered in the past and determining whether these answers could have been pieced together by a user to infer private information about an individual or a group of individuals.

More formally, given a set of forbidden views, $\mathcal{V} = \{V_1, \ldots, V_k\}$, of a database that must be kept confidential according to the data disclosure policies, a batch of queries, $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, that have been posed over this database, and a system-defined notion of suspiciousness, S, the task of an auditor is to determine whether $\mathcal{Q}$ is suspicious with respect to $\mathcal{V}$. In all the notions of suspiciousness that we study in this chapter a batch of queries is suspicious with respect to a set of forbidden views if it is suspicious with respect to any one view. Therefore we focus on the auditing problem for a single forbidden view.

Throughout this chapter, we also restrict ourselves to the class of "duplicate-preserving" *select-project-join* (SPJ) queries. Here by duplicate-preserving we mean that the `SELECT` clauses do not contain `distinct` in the select list and multi-set semantics is used in projection. Since by default all the queries that we consider will be duplicate-preserving, we drop this qualifier from here on. Furthermore, we assume that all foreign key relationships are known and the universal table obtained by joining along these foreign keys is polynomial in the size of the input tables.

The need for some sort of an audit mechanism in database management systems is clear. For example, an individual on receiving targeted health advertisements might suspect his health-care provider of having leaked private information from his medical records to interested parties. If the provider's privacy policy stipulates that it does not release patient data to external parties, it would be in the best interests of the provider to be able to demonstrate compliance with this policy.

It is in this context that the authors of [ABF$^+$04] introduced an auditing framework for checking whether any one query that had been posed in the past accessed/revealed some specified private data. In their approach, *audit expressions* are formulated to specify forbidden parts of the data that one would like to ensure were not wrongfully disclosed. The audit component then returns all suspicious queries that accessed this data during their execution. For reasons that will become evident later, we call the notion of suspiciousness used here *semantic suspiciousness*.

In general, however, it need not be any single query on its own that is the cause of a disclosure. Instead, the results of a few different queries in conjunction might enable a user to infer private data and we therefore extend the definition of semantic suspiciousness to a batch of queries in Section 6.3. We discover that an extension of the approach used in [ABF$^+$04] would suffice for auditing the class of duplicate-preserving SPJ queries. We call this auditor a *semantic auditor*.

A drawback of our approach to semantic auditing is that it requires that candidate queries actually be run against the database. A natural question to ask, therefore, is whether this could be avoided. For this purpose, in Section 6.4, we formulate the notion of *strong syntactic suspiciousness* wherein suspiciousness of queries is determined independently of the underlying database instance. We show that it is in fact NP-hard to audit a batch of queries under this definition, even when we restrict ourselves to the class of conjunctive select-project queries. We therefore define a relaxed notion of *weak syntactic suspiciousness*, and provide a polynomial time auditing algorithm for batches of "duplicate-preserving" conjunctive queries under this definition. Note that such an auditor would only be more conservative, in that it would certainly detect query batches that are strongly suspicious as well.

This chapter is thus an exploration of auditing schemes for different notions of suspiciousness. The new definitions of suspiciousness that we introduce fall in between those introduced in [MS04, MG06] at one end and [ABF+04] at the other, both in terms of their privacy guarantees as well as the tractability of auditing with respect to these definitions for certain classes of queries. We illustrate these relationships in Section 6.5. Further, we draw an interesting connection to another database mechanism for controlling access to data — namely fine-grained access control studied in [Mot89, RS00, RS01, RSD99, RMSR04].

In the next section we give a brief overview of relevant results in these areas of auditing and access control.

## 6.2 Related Work

**Auditing Aggregate Queries:** The problem of auditing queries has been extensively studied in the context of statistical databases [DJL79, Rei79, Chi86, KPR00, KMN05, NMK+06]. Statistical databases allow users to retrieve only aggregate statistics over subsets of its data. In this chapter we consider only SPJ queries and our work is orthogonal to the body of work on statistical databases.

**Perfect Privacy:** In [MS04, MG06] the authors consider the problem of ensuring "perfect privacy": as a database system reveals various views of its data, does it disclose any information *at all* about a view that was required to be kept confidential. The work here can also be cast in the auditing framework proposed in this chapter — the secret view corresponds to the forbidden view/audit expression in our scenario, and the views of the data that were released correspond to queries that were answered. The notion of information disclosure used in [MS04, MG06] is, however, very strict and results in the following definition of suspiciousness:

**Definition 6.1 (Critical Tuple)** A tuple $t$ belonging to the domain of all possible tuples in the database, is critical for a query $Q$, if there exists any possible database instance $I$ for which $Q(I - \{t\}) \neq Q(I)$, i.e., $t$ is critical for $Q$ if there exists some instance for which dropping $t$ makes a difference to the result of $Q$. □

**Definition 6.2 (Suspiciousness of a Query Batch Under Perfect Privacy)** A batch of SQL queries $\mathcal{Q}$ is suspicious with respect to a secret view $\mathcal{V}$ according to the perfect privacy definition of security, if and only if there exists some critical tuple common to both $\mathcal{V}$ and to all the queries in $\mathcal{Q}$. □

This definition is grounded in a precise theory of information disclosure for databases where tuples are drawn independently from some probability distribution. The result, however, is a very weak definition of suspiciousness, resulting in very strong privacy guarantees and causing many seemingly innocuous queries to be marked as suspicious. Consider, for example, a database containing the names and phone numbers of patients in a hospital and imagine that we wish to keep secret all the phone numbers listed in this database. A query asking for the names of all the patients in the hospital would be considered suspicious with respect to the secret view even though not a single phone number would have been revealed by this query. This is because every possible tuple in the domain of all tuples is critical to both the query and the secret view. The idea is that simply by revealing information about the size of the database, the query revealed some small amount of information about the phone numbers column and therefore should be considered suspicious.

**Auditing SQL Queries:** In [ABF$^+$04], the authors study the problem of determining whether any single SQL query in the query log accessed forbidden information. Here the data being subject to a disclosure review is specified very simply through an audit expression that closely resembles a SQL query:

```
AUDIT audit list
FROM table list
WHERE condition list
```

The audit expression can be viewed as an SPJ query, specifying a certain view of the database that it wishes to ascertain was not disclosed. It essentially identifies the tuples of interest from the cross-product of tables in the `FROM` clause via predicates in the `WHERE` clause. The audit expression thus asks for all queries that accessed *all* the `audit list` columns for *any* of these tuples. We illustrate this approach with some examples from [ABF$^+$04]. Consider the audit expression:

```
AUDIT p.disease
FROM Patients p
WHERE p.zipcode = 94305
```

This expression asks for all queries that accessed the disease column of any patient living in the zipcode 94305. All such queries will be considered suspicious. Now consider the SQL query:

```
SELECT p.zipcode
FROM Patients p
WHERE p.disease = ‘diabetes’
```

If any patient who has diabetes lives in zipcode 94305, this SQL query will be considered suspicious with respect to the above audit expression. This is because, in answering the query, the disease column of a patient living in zipcode 94305 was accessed. On the other hand, this SQL query would not be suspicious with respect to the audit expression given below:

```
AUDIT p.zipcode
FROM Patients p
WHERE p.disease = ‘hypertension’
```

This is because this audit expression is only interested in checking if the zip-code of any patient with hypertension was revealed. But what if a patient has both diabetes and hypertension? Although this patient’s address would be revealed by the SQL query, the fact that he had hypertension was not relevant to the query and so it is reasonable to deem the SQL query unsuspicious. Note that in doing so, we take queries at their face value, assuming away background knowledge. For instance, a user might know that most patients with diabetes also have hypertension and thus his query ought to be considered suspicious. But the assumption made here is that users do not use external information to formulate queries so as to deduce information without detection.

We now formalize what it means for a query to be suspicious with respect to an audit expression. Consider an SPJ query of the form $Q = \pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}))$ and an audit expression of the form $A = \pi_{C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S}))$. Here $\mathcal{T}$ is the cross-product of tables common to both the audit expression and the query and $\mathcal{R}$ and $\mathcal{S}$ are the crossproducts of other tables in their `FROM` clauses. $C_{Q(resp.\ A)}$ are the columns that are projected out in $Q$ (resp. $A$) and $P_{Q(resp.\ A)}$ are the predicates of $Q$ (resp. $A$). We also use $C_Q^*$ to denote all the column names that appear anywhere in the query $Q$.

**Definition 6.3 (Candidate Query)** A query $Q$ is a candidate query with respect to an audit expression $A$, if $Q$ accesses all the columns that $A$ specifies in its `audit` `list`, i.e. $C_Q^* \supseteq C_A$. □

**Definition 6.4 (Indispensable Tuple)** A tuple $t \in \mathcal{T}$ is indispensable to a query $Q$ if the presence or absence of $t$ makes a difference to the result of $Q$, i.e. $\pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R})) \neq \pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} - \{t\} \times \mathcal{R}))$. □

**Definition 6.5 (Suspicious Query)** A candidate query $Q$ is suspicious with respect to an audit expression $A$, if they share an indispensable tuple. □

The idea is that an indispensable tuple for the audit expression would be one of the forbidden tuples being subjected to a disclosure review. So if any one of these tuples is also an indispensable tuple for a candidate query in the query log, then that query would have accessed all the columns of the `audit list` for the forbidden tuple and should therefore be considered suspicious. .

For the class of SPJ queries, the condition of sharing an indispensable tuple translates to the following: a candidate query $Q$ is suspicious with respect to an audit expression $A$ if and only if $\sigma_{P_A}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R} \times \mathcal{S})) \neq \emptyset$. So now the audit process is simple: for every candidate query, $Q$, in the query log if the result of the running the query $\sigma_{P_A}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R} \times \mathcal{S}))$ is non-empty, then $Q$ is marked as suspicious.

Since the notion of suspiciousness of a query derives meaning from a particular world view which is the current state of the database, we call it *semantic suspiciousness*. This is in contrast with *syntactic suspiciousness* that we will define later where suspiciousness of a query depends entirely on its structure and not on the current database state. We next briefly describe the notion of query validity described in access control work.

**Fine-Grained Access Control:** In database access control literature, the problem studied is essentially the dual of the auditing problem: Given a set of "authorization views", $\mathcal{U} = \{U_1, \ldots, U_k\}$ for a user and a definition of validity, $\mathcal{V}$, what queries posed by the user are valid with respect to the authorization views and therefore safe to answer? Here the authorization views correspond to information that the user is allowed to access and they can be specified via SQL queries similar to audit expressions. One notion of validity considered in particular in [Mot89, RS00, RS01, RSD99, RMSR04] is that of unconditional validity of a query:

**Definition 6.6 (Unconditionally Valid Query)** Given a set of authorization views, a query $Q$ is said to be unconditionally valid if there is a query $Q'$ that can be written using only the instantiated authorization views, and is equivalent to $Q$, that is $Q'$ produces the same result as $Q$ on all possible database instances. □

We will see in Section 6.5 how the notion of syntactic suspiciousness of a query is tightly connected to the notion of unconditional validity.

## 6.3 Semantic Auditing for a Batch of SQL Queries

The work most closely related to ours and that we build on is that of [ABF+04] where semantic suspiciousness was considered for single SQL queries. In general, however, no one query in isolation may access all the columns of the `audit list` of an audit expression, instead a few queries together may cause sensitive information to be disclosed. For example, the audit expression might be

```
AUDIT p.name, p.disease
FROM Patients p
WHERE p.zipcode = 94305
```

Here the data that needs to be kept secret is the association between names and diseases of patients living in the zipcode 94305. Now consider the SQL queries:

```
SELECT p.zipcode
FROM Patients p
WHERE p.disease = ‘diabetes’
```

```
SELECT p.name
FROM Patients p
WHERE p.zipcode = 94305
```

Note that neither of these queries on their own reveal the association between name and disease of any individual living in zipcode 94305, however the combination of the queries does reveal something. For instance, if there is only one patient in zipcode 94305 and this patient has diabetes, then these two queries have revealed the name, disease association for that individual. In general, there are subtle ways in which the results of queries could be combined to reveal information.

One simplifying assumption that we make here is that the adversary is very powerful and knows exactly which tuples in the results of two queries join together, i.e., we assume that in each query he implicitly also selects the key column. The resulting definition of suspiciousness is conservative — it may at times label as suspicious batches of queries whose results could not have been easily joined in reality.

**Definition 6.7 (Semantically Suspicious Query Batch)** A batch of queries, $\mathcal{Q}$ is said to be semantically suspicious with respect to an audit expression $A$ if there is some subset of queries $\mathcal{Q}' \subseteq \mathcal{Q}$ such that (1) a tuple $t \in \mathcal{T}$ is indispensable to both $A$ and every query in $\mathcal{Q}'$ and (2) the queries in $\mathcal{Q}'$ together access all the columns of the `audit list` in $A$. Here $\mathcal{T}$ is the cross product of all the tables common to $A$ and every query in $\mathcal{Q}'$. $\square$

Note that, as in [ABF$^+$04], we require the query to access *all* the columns of the `audit list` in ordered to be even considered for suspiciousness testing. This is motivated by findings such as those in [Swe00] where attributes such as date-of-birth, gender or zipcode on their own are not selective, however in conjunction can be used to uniquely identify individuals. In such a case, revealing the diseases of all females in a certain zipcode would be harmless enough, however revealing this information in conjunction with date-of-birth information could cause significant privacy leaks. The much stronger notion of perfect privacy [MS04, MG06], which does not require this would mark as suspicious many innocuous views as explained in Section 6.2.

This definition of suspiciousness is a natural extension of the definition in [ABF$^+$04] and as in that case, lends itself to an auditing approach where a query is executed over the database for every query in the query batch.

**Theorem 6.3.1** *There is a polynomial time algorithm to test semantic suspiciousness of a batch of SQL queries $\mathcal{Q}$ with respect to an audit expression $A$.*

**Proof:** We first run the query $\pi_*(\sigma_{P_A}(\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ on the given database instance. On the resulting view, we now run every single query $Q \in \mathcal{Q}$. Each time one of the tuples, $t$, in this view satisfies the predicates of $Q$, the portion of $t$ that comes from $T$ must be indispensable to both $Q$ and $A$. This is because $Q$ is a duplicate-preserving query: every input tuple that satisfies its predicates will have a corresponding output tuple. We mark as "accessed" all the cells of $t$ that were accessed by $Q$ during its execution. If at the end of this process, there exists a tuple in the view all of whose `audit list` columns are marked as accessed, then $\mathcal{Q}$ is semantically suspicious with respect to $A$. To determine exactly which set of queries from the batch were involved in the disclosure, we can maintain with each marked cell, the set of queries that accessed it. The cross-product of all the query sets for each cell of each disclosed row gives us all the query sets that were involved in the disclosure. Under the assumption that all joins (along foreign keys) are polynomial in the size of the input tables, this is a polynomial time algorithm. □

A natural question to ask is whether executing every single query in $\mathcal{Q}$ against the forbidden view specified by $A$ could be avoided. This leads us directly to the next section.

## 6.4 Syntactic Auditing for a Batch of SQL Queries

In this section we define notions of suspiciousness that are independent of the actual data in the database.

### 6.4.1 Strong Syntactic Suspiciousness

**Definition 6.8 (Strong Syntactic Suspiciousness of a Query Batch)** A query batch is said to be strongly syntactically suspicious with respect to an audit expression if there is some possible instantiation of database tables for which it is semantically suspicious. □

Note that since the current database tables form an instantiation, it follows that if a set of queries is semantically suspicious with respect to an audit expression $A$, then it is also strongly syntactically suspicious with respect to $A$.

Unlike semantic auditing, however, the hope is that strong syntactic auditing would require the auditor to only analyze the structure of queries in the query log, not the answers to the queries on the actual database tables since suspiciousness is independent of the underlying database instantiation. Unfortunately, our result here is negative, even if we restrict ourselves to just the class of conjunctive select-project queries without any joins.

**Theorem 6.4.1** *Testing whether a batch of conjunctive select-project queries is strongly syntactically suspicious with respect to an audit expression is NP-hard.*

**Proof:** We provide a reduction from 3-SAT. Consider a 3-SAT formula $(x_{11} \vee x_{12} \vee x_{13}) \wedge (x_{21} \vee x_{22} \vee x_{23}) \wedge \dots (x_{m1} \vee x_{m2} \vee x_{m3})$. We now create a set of queries and an audit expression such that the queries are syntactically suspicious with respect to the audit expression if and only if the above 3-SAT formula is satisfiable.

Let $y_1, \dots y_n$ be the variables that appear in the clauses of the 3-SAT formula.

For each variable $y_i$ create a column $Y_i$ that can take on two possible values 0 or 1. For the $j^{th}$ clause $(x_{j1} \vee x_{j2} \vee x_{j3})$ create a column $X_j$ that can take on only one value. Let literal $y_i$ occur in clauses $i_1, i_2, \ldots i_k$ and literal $\bar{y}_i$ occur in clauses $\bar{i_1}, \bar{i_2}, \ldots \bar{i_m}$. We then create $n$ query pairs corresponding to each variable, where the $i$th pair looks like:

$Q_i^+$: `SELECT` $X_{i_1}$, $X_{i_2}$, $X_{i_k}$, $Y_i$
`FROM T`
`WHERE` $Y_i = 1$

and

$Q_i^-$: `SELECT` $X_{\bar{i_1}}$, $X_{\bar{i_2}}$, $X_{\bar{i_m}}$, $Y_i$
`FROM T`
`WHERE` $Y_i = 0$

The audit expression is

`AUDIT` $X_1$, $X_2$, $X_3$, $\ldots$, $X_m$
`FROM` $T$

Now if the 3-SAT formula is satisfiable, then this batch of queries is syntactically suspicious with respect to the audit expression: Consider the queries $Q_i^+$ for every $y_i$ that is set to true in the satisfying assignment and $Q_i^-$ for every $y_i$ that is set to false. This subset of queries and the audit expression all share an indispensable tuple - namely the tuple with $Y_i = 0$ for every $y_i$ that is set to false and $Y_i = 1$ for every $y_i$ that is set to true. Moreover, since every clause is satisfied, this subset of queries together selects all the columns that are in the `audit list` of the audit expression.

Similarly, if the batch of queries is syntactically suspicious with respect to the audit expression, then the 3-SAT formula must be satisfiable. Some subset of the queries must share an indispensable tuple with the audit expression. It cannot be the case that both $Q_i^+$ and $Q_i^-$ are included in this subset for no one tuple can be indispensable to both $Q_i^+$ and $Q_i^-$ as their selection predicates are contradictory. For each query in the subset of the form $Q_i^+$, we set $y_i$ to be true and for each query of the form $Q_i^-$, we set $y_i$ to be false. For all the $y_i$s that are not set in the process, we set them arbitrarily to 0 or 1. Since the select columns of this subset of queries cover all the columns in the `audit list`, this ensures that every clause is set to true.

Thus we've shown that the 3-SAT formula is satisfiable if and only if the above query batch is suspicious with respect to the audit expression. □

Weakening the definition of suspiciousness, however, enables us to make some positive claims, while providing auditors that would only be more conservative than strong syntactic auditors, i.e., all queries batches marked as strongly suspicious would also be classified as weakly suspicious.

### 6.4.2 Weak Syntactic Suspiciousness

**Definition 6.9 (Weak Syntactic Suspiciousness of a Query Batch)** A batch of SPJ queries $\mathcal{Q}$ is weakly syntactically suspicious with respect to an audit expression $A$, if there exists some subset of the queries $\mathcal{Q}' \subseteq \mathcal{Q}$ and some instantiation of database tables $I$ such that (1) a tuple $t \in \mathcal{T}$ is indispensable to both $A$ and every query in $\mathcal{Q}'$ in the context of $I$ and (2) the queries in $\mathcal{Q}'$ together access *at least one* of the columns of the `audit list` in $A$. Here $\mathcal{T}$ is the cross product of all the tables in $I$ common to both $A$ and every query in $\mathcal{Q}'$. □

Note that to arrive at this definition, we simply weakened condition (2) in the definition of strong syntactic suspiciousness and thus a query batch that is strongly syntactically suspicious will also be weakly syntactically suspicious. Note also that weakening this condition enables us to come up with the following decomposability result.

**Theorem 6.4.2** *A batch of SPJ queries is weakly syntactically suspicious with respect to an audit expression if and only if one of the SPJ queries in the batch is weakly syntactically suspicious with respect to the audit expression.*

This enables us to test suspiciousness of a batch simply by testing suspiciousness of every individual query on its own. Still testing suspiciousness of an arbitrary SPJ query under this definition is NP-hard. The intuition is that an arbitrary 3-SAT formula could be embedded in the where clause predicates of the query.

**Theorem 6.4.3** *Testing weak syntactic suspiciousness of an arbitrary SPJ query Q with respect to an audit expression A is NP-hard.*

**Proof:** Consider a 3-SAT formula $(x_{11} \vee x_{12} \vee x_{13}) \wedge (x_{21} \vee x_{22} \vee x_{23}) \wedge \ldots (x_{m1} \vee x_{m2} \vee x_{m3})$ with $n$ variables $y_1, \ldots, y_n$ and $m$ clauses. Let the table $T$ contain $n$ binary attributes $Y_1, \ldots, Y_n$ corresponding to these variables. In the following, we use the predicate $X_{ij} = 1$ to mean $Y_k = 1$ if $y_k$ is the $j$th literal of the $i$th clause and $Y_k = 0$ if $\bar{y_k}$ is the $j$th literal of the $i$th clause. Consider the query

Q: `SELECT` $Y_1, Y_2, \ldots, Y_n$
`FROM T`
`WHERE` $X_{11} = 1 \vee X_{12} = 1 \vee X_{13} = 1 \quad \wedge$
$X_{21} = 1 \vee X_{22} = 1 \vee X_{23} = 1 \quad \wedge$
$\vdots$
$X_{m1} = 1 \vee X_{m2} = 1 \vee X_{m3} = 1$

and the audit expression

A: `AUDIT` $Y_1, Y_2, \ldots, Y_n$
`FROM T`

Now $Q$ is weakly syntactically suspicious with respect to $A$ if and only the 3-SAT formula is satisfiable. This is because every tuple in the domain of possible tuples that is indispensable to $Q$ would correspond to a satisfying assignment of the 3-SAT formula. □

The natural question to ask is whether there are classes of queries for which weak syntactic auditing can be efficiently done. We restrict ourselves to the class of conjunctive

SPJ queries and show that it is possible to develop a polynomial time algorithm for detecting weakly suspicious query batches. In this chapter we only provide a sketch of the auditing algorithm for such queries.

**Theorem 6.4.4** *There exists a polynomial time auditing algorithm for determining if a batch of conjunctive queries is weakly syntactically suspicious with respect to an audit expression.*

*Proof Sketch:* First, due to Theorem 6.4.2 it suffices to check suspiciousness of each query in the batch individually. If any one query is suspicious, we can mark the entire batch as suspicious. Second, we only check suspiciousness of queries that access at least one of the `audit list` columns of the audit expression. Now for such a candidate conjunctive query, suspiciousness is tested in polynomial time by testing compatibility of the predicates in the `WHERE` clauses of the candidate query and the audit expression, i.e., could there be any tuple in the domain of possible tuples of $\mathcal{T} \times \mathcal{R} \times \mathcal{S}$ that could satisfy all the predicates in both the candidate query and the audit expression. If such a tuple could exist, then the query is weakly syntactically suspicious with respect to the audit expression. This follows from the fact that the query is duplicate-preserving thereby ensuring that every input tuple that satisfies the `WHERE` clause predicates will have a corresponding output tuple and will therefore be indispensable to the query. Consider, for now, predicates over numerical attributes. These predicates could be between two attributes or between an attribute and a constant. Group attributes in to equivalence classes based on equality predicates between attributes. Then a pass over the inequality predicates between attributes enables us to establish relationships between these equivalence classes. If at any point a strict inequality is expressed over attributes that belong to the same equivalence class, we immediately know that the set of predicates is incompatible and can stop. Similarly other such inconsistencies can be checked for. Once predicates between attributes have been processed, the predicates between attributes and constants can be used to derive upper and lower bounds on the values of attributes in the equivalence classes. The predicates are compatible only if the upper and lower bounds of all attributes in a particular equivalence class have a common intersection and so on, and this can be checked in polynomial time.

In general, any of the query classes that can be audited under the perfect privacy notion of suspiciousness, can also be audited under weak syntactic suspiciousness — the only additional check that needs to be done is whether the queries access some of the `audit list` columns. This means that techniques in [MG06] can be used for weakly syntactically auditing different classes of queries.

## 6.5 Auditing and Access Control

As we have seen, previous work in [MS04, MG06, ABF$^+$04] can be cast in the auditing framework proposed in this chapter. The differences lie in the definitions of suspiciousness used. If $A$ and $B$ are two different notions of suspiciousness, then we use the notation $A \leq B$ to indicate that $A$ is a weaker definition than $B$, i.e., for a given set of forbidden views on a database, an auditor that audits with respect to $A$ will always mark as suspicious every batch of queries that are marked as suspicious with respect to $B$. Then it is clear that the following relationship holds:

**Theorem 6.5.1** *Perfect Privacy (PP) $\leq$ Weak Syntactic Suspiciousness (WSS) $\leq$ Strong Syntactic Suspiciousness (SSS) $\leq$ Semantic Suspiciousness (SEM).*

The table below summarizes some properties of auditors that audit with respect to these definitions of suspiciousness. Recall that the hardness results are for the default of duplicate-preserving queries. Also the polynomial time result for semantic auditing holds under the assumption that foreign key joins are polynomial in the size of the input tables.

| | PP | WSS | SSS | SEM |
|---|---|---|---|---|
| Online Auditing | Yes | Yes | Yes | No |
| SPJ | NP-hard | NP-hard | NP-hard | P |
| Conjunctive | P | P | NP-hard | P |

Here, the first row asks the question whether it would be meaningful to use auditors that audit with respect to these definitions of suspiciousness in an online fashion. In online auditing, as queries arrive, the task of the auditor is to determine whether the current query in conjunction with all previously answered queries would be suspicious with respect to the forbidden views. If so, the query must be denied, else it can be answered. If semantic suspiciousness is used as the definition of suspiciousness, online auditing would not make much sense as the act of denying a query itself might leak information about the forbidden view. For instance, if the forbidden view is specified as

```
AUDIT p.name, p.disease
FROM Patients p
WHERE p.zipcode = 94305
```

Now suppose the following query is permitted and it returns only one disease.

```
SELECT p.disease
FROM Patients p
WHERE p.zipcode = 94305
```

But the following query is denied

```
SELECT p.name
From Patients p
WHERE p.zipcode = 94305 and p.name = ‘Alex’
```

The only reason that this could happen is that Alex who lives in 94305 is actually in the database and his disease was the one returned by first query. Even though the second query was denied, the name-disease association of an individual in the forbidden view was revealed. Thus online auditing with respect to the semantic notion of suspiciousness does not make sense. In general, as shown in [KMN05], denials that are based on the actual data in a database can leak considerable information. Auditors that audit according to the other definitions of suspiciousness would, however, be independent of the true state of the database and could thus be used in an online fashion as well. Any decision to deny a query by such auditors could always be *simulated* by a user and thus denials would not leak information.

As we saw in Section 6.2, another closely related problem to auditing is that of database access control, where the problem studied is almost the dual of the auditing problem. We now formalize this connection.

**Theorem 6.5.2** *Given a forbidden view, $V$, there exists an authorization view, $U$, such that a query is suspicious with respect to $V$ under the perfect privacy notion of suspiciousness if and only if it is not unconditionally valid with respect to $U$.*

**Proof:** The proof has two parts. Firstly suppose query $Q = \pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}))$ is suspicious with respect to view $V$, represented by the audit expression $\pi_{C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S}))$ under the perfect privacy notion of suspiciousness. Consider the view $U = \pi_*(\sigma_{\bar{P_A}}(\mathcal{T} \times \mathcal{S} \times \mathcal{R}))$. We will first show $Q$ is not unconditionally valid with respect to $U$. By Definition 6.2 $\exists t \in (\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ such that $P_A(t) = 1$ and $P_Q(t) = 1$. Consider the database instance $\mathcal{I}$ consisting of the single tuple $t$ in $(\mathcal{T} \times \mathcal{R} \times \mathcal{S})$. Then as $P_Q(t) = 1$, the $Q(\mathcal{I})$ is non-empty. However, $\bar{P_A}(t) = 0$ so the view $U$ does not contain the tuple $t$, and hence the query $Q$ rewritten using view $U$ is empty for this instance. Thus as the answers differ on this instance, $Q$ is not unconditionally valid with respect to $U$.

For the other direction, suppose $Q$ not suspicious with respect to $V$. As they never share a critical tuple, for all instances $\mathcal{I}$ $\forall t \in (\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ such that $P_Q(t) = 1 \leftrightarrow P_A(t) = 0$, i.e. $P_Q \rightarrow \bar{P_A}$. The query. $Q$ can be rewritten so that the selection conditions $P_Q$ and $\bar{P_A}$ are both applied on the view $U$ before projecting columns $C_Q$. Thus $Q$ is unconditionally valid with respect to $U$.

□

**Theorem 6.5.3** *Given a conjunctive forbidden view, $V$, such that there is no intersection between the audit list columns and the where clause predicate columns, there* *exist a set of authorization views, $U$, such that a conjunctive query with non-trivial predicates (define non-trivial beforehand) is suspicious with respect to $V$ under the weak syntactic notion of suspiciousness if and only if it is not unconditionally valid with respect to $U$.*

**Proof:** Suppose $Q$ is not unconditionally valid with respect to $U$. Then there for some instance $\mathcal{I}$ $\exists t \in (\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ such that $P_Q(t) = 1$ and $\bar{P_A}(t) = 0$, i.e. $P_A(t) = 1$. Now, view $V$ and query $Q$ share the critical tuple $t$, so $Q$ is suspicious with respect to the view $V$ under the notion of perfect privacy.

□

**Theorem 6.5.4** *Given a forbidden view over a database, and a weak syntactic auditor, there exist a set of authorization views over the database such that an SPJ query is suspicious with respect to the forbidden view if and only if it is unconditionally valid with respect to the authorization views.*

**Proof:** The intuition behind this result is that the forbidden view implicitly defines a set of authorization views that are essentially its complement. We will show the result here for the case where all the tuples in the database tables have unique tuple identifiers, but the result holds even if this is not true (although the corresponding set of authorization views would be different).

Let the forbidden view over the database be represented by the audit expression $\pi_{C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S}))$. Then define the authorization views over the database as follows $\pi_*(\sigma_{\bar{P_A}}(\mathcal{T} \times \mathcal{S} \times \mathcal{U}))$ and $\pi_{*\setminus C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S} \times \mathcal{U}))$ where $\mathcal{U}$ represents the cross product of the remaining base tables of the database and $\bar{P_A}$ is the complement of the selection predicates of the forbidden view. We will use the notation $P(t) = 1$ to denote that tuple $t$ satisfies predicates $P$.

Now suppose a query $\pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}))$ is considered weakly syntactically suspicious with respect to the audit expression. This means that $C_Q^* \cap C_A \neq \emptyset$ and there exists a tuple $t$ in the domain of all possible tuples in $\mathcal{T} \times \mathcal{R} \times \mathcal{S}$ such that

$P_A(t) = 1$ and $P_Q(t) = 1$. Note, however, that for no such tuple where $P_A(t) = 1$ are any of the columns in $C_A$ included in any of the authorization views. This means that there must be at least one column in $C_Q^*$ for the tuple $t$ that is not included in any of the authorization views. Therefore there can be no $Q'$ expressed over the authorization views that is equivalent to $Q$. Thus if the query is weakly syntactically suspicious, it is not unconditionally valid with respect to the authorization views.

Similarly, now suppose query $Q$ is not weakly syntactically suspicious. This means that either $C_Q^* \cap C_A = \emptyset$ or there exists no tuple $t$ in the domain of all possible tuples in $\mathcal{T} \times \mathcal{R} \times \mathcal{S}$ such that $P_A(t) = 1$ and $P_Q(t) = 1$. If the latter is true, then all tuples that satisfy $P_Q(t) = 1$ must satisfy $\bar{P_A}(t) = 1$ and such tuples can be returned by predicates expressed over view 1. If the former is true, then since none of the columns in $C_Q^*$ appear in $C_A$, these columns will be queriable for all tuples in the two authorization views and a query can be written over the authorization views that is equivalent to the query $Q$ over the entire database for all possible database instantiations. To ensure that multiplicities of tuples in the result are maintained, the query can group by the tuple identifiers. Thus every query that is not weakly syntactically suspicious is also unconditionally valid with respect to the defined authorization views. $\square$

This result is interesting as it ties together work in auditing and work in the database access control literature. Mechanisms that are used for detecting validity of an SPJ query could now also be used for detecting suspiciousness of SPJ queries. Ideally a database management system would try to incorporate both mechanisms - a database access control mechanism that gives users access to various parts of the data, thereby providing utility, and an auditing mechanism to detect privacy breaches. An interesting avenue for future work would be to see how both these mechanisms could be combined in a system to work together. Checking for consistency in such a system would then be an interesting question, i.e., for a set of forbidden views, a notion of suspiciousness, a set of authorization views and a notion of validity, is it the case that every suspicious query is invalid and every valid query is not suspicious. Another interesting question to ask would be for a given level of privacy (forbidden views and definition of suspiciousness), what is the maximum utility one could have while maintaining consistency and vice versa. We are able to answer this question for weak syntactic suspiciousness and unconditional validity, but for other definitions, the question remains wide open.

## 6.6 Conclusions

In this chapter we introduced a framework for auditing queries and several different notions of suspiciousness that differed in their privacy guarantees as well as the tractability of auditing under them for different classes of queries. We tied in our work with existing auditing mechanisms and also drew an interesting connection to the area of database access control. Extending the tractability results to other classes of queries would be an interesting avenue for future work as would investigating further, the connection to database access control mechanisms.

# Part III

# Distributed Architecture for Privacy

# Chapter 7

# Distributed Architectures for Secure Database Services

The results in this chapter appear in [ABG$^+$05]. Algorithms for an implementation of ideas in this chapter are presented in [FGGM$^+$07].

## 7.1 Introduction

The database community is witnessing the emergence of two recent trends set on a collision course. On the one hand, the outsourcing of data management has become increasingly attractive for many organizations [HIM02]; the use of an external database service promises reliable data storage at a low cost, eliminating the need for expensive in-house data-management infrastructure, e.g., [Mor02]. On the other hand, escalating concerns about data privacy, recent governmental legislation [sb102], as well as high-profile instances of database theft [O'B04], have sparked keen interest in enabling secure data storage.

The two trends described above are in direct conflict with each other. A client using a database service needs to trust the service provider with potentially sensitive data, leaving the door open for damaging leaks of private information. Consequently, there has been much recent interest in a so-called *Secure Database Service* – a DBMS that provides reliable storage and efficient query execution, while not knowing the *contents* of the database [KC04]. Such a service also helps the service provider by limiting their liability in case of break-ins into their system – if the service providers do not know the contents of the database, neither will a hacker who breaks into the system.

Existing proposals for secure database services have typically been founded on encryption [HILM02, HH04, AKSX04]. Data is encrypted on the (trusted) client side before being stored in the (untrusted) external database. Observe that there is always a trivial way to answer all database queries: the client can fetch the entire database from the server, decrypt it, and execute the query on this decrypted database. Of course, such an approach is far too expensive to be practical. Instead, the hope is that queries can be transformed by the client to execute directly on the encrypted data; the results of such transformed queries could be post-processed by the client to obtain the final results.

Unfortunately, such hopes are often dashed by the privacy-efficiency trade-off of encryption. Weak encryption functions that allow efficient queries leak far too much information and thus do not preserve data privacy [KC04]. On the other hand, stronger encryption functions often necessitate resorting to Plan A for queries – fetching the entire database from the server – which is simply too expensive. Moreover, despite increasing processor speeds, encryption and decryption are not exactly cheap, especially when performed over data at fine granularity.

We propose a new approach to enabling a secure database service. The key idea is to allow the client to partition its data across *two*, (and more generally, *any number of*) logically independent database systems that cannot communicate with each other. The partitioning of data is performed in such a fashion as to ensure that the exposure of the contents of any *one* database does not result in a violation of privacy. The client executes queries by transmitting appropriate sub-queries to each database, and then piecing together the results at the client side.

The use of such a distributed database for obtaining secure database services offers many advantages, among which are the following:

**Untrusted Service Providers** The client does not have to trust the administrators of either database to guarantee privacy. So long as an adversary does not gain access to *both* databases, data privacy is fully protected. If the client were to obtain database services from two different vendors, the chances of an adversary breaking into both systems is reduced greatly. Furthermore, "insider" attacks at one of the vendors do not compromise the security of the system as a whole.

**Provable Privacy** The presence of two databases enables the efficient encoding of sensitive attributes in an information-theoretically secure fashion. To illustrate, consider a sensitive fixed-length numerical attribute, such as a credit-card number. We may represent a credit card number $c$, by storing $c$ XORed with a random number $r$ in one database, and storing $r$ in the other database. The set of bits used to represent the credit-card number in either database is completely random, thus providing perfect privacy. However, we may recover the number merely by XORing the values stored in the two databases, which is more efficient than using expensive encryption and decryption functions.

**Efficient Queries** The presence of multiple databases enables the storage of many attribute values in unencrypted form. Typically, the exposure of a *set* of attribute values corresponding to a tuple may result in a privacy violation, while the exposure of only some subset of it may be harmless. For example, revealing an individual's name and her credit card number may be a serious privacy violation. However, exposing the name alone, or the credit card number alone, may not be a big deal [sb102]. In such cases, we may place individuals' names in one database, while storing their credit-card number in the other, avoiding having to encrypt either attribute. A consequence is that queries involving both names and credit-card numbers may be executed far more efficiently than if the attributes had been encrypted.

The rest of this paper is organized as follows. In Section 7.2, we present a general architecture for the use of multiple databases in preserving privacy, describing the space of techniques available for partitioning data and the trade-offs involved. In Section 7.3, we define a specific notion of privacy based on hiding sets of attribute values, and consider how to achieve such privacy using a subset of the available partitioning techniques. Section 7.4 expands upon this framework and describes how queries may be transformed, optimized and executed in a privacy-preserving fashion. Section 7.5 discusses how one may design the database schema in order to minimize

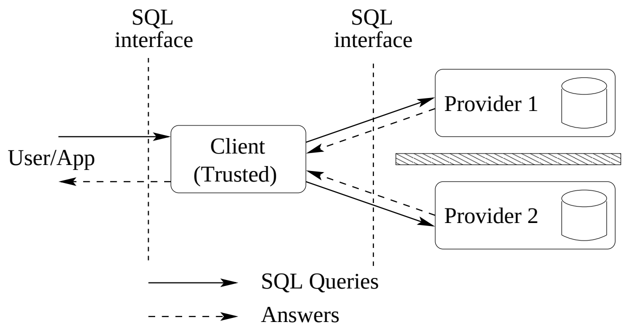


Figure 7.1: *The System Architecture*

the execution cost of a given query workload, while obeying the constraints imposed by the needs of data privacy.

## 7.2 General Architecture

The general architecture of a distributed secure database service, as illustrated in Figure 7.1, consists of a trusted client as well as two or more servers that provide a database service. The servers provide reliable content storage and data management but are not trusted by the client to preserve content privacy.

The client wants to out-source the (high) costs of managing permanent storage to the service providers; hence, we assume that the client does not store any persistent data. However, the client has access to cheap hardware – providing processing power as well as temporary storage – which is used to provide three pieces of functionality:

**1. Offer a DBMS Front-End** The client exports a standard DBMS front-end to client-side applications, supporting standard SQL APIs.

**2. Reformulate and Optimize Queries** The queries received by the client need to be translated into appropriate SQL sub-queries to be sent to the servers; such translation may involve limited forms of query-optimization logic, as we discuss later in the paper.

**3. Post-process Query Results** The sub-queries are sent to the servers (using a standard SQL API), and the results are gathered and post-processed before being returned in a suitable form to the client-side application.

We note that all three pieces of functionality are fairly cheap, at least if the amount of post-processing required for queries is limited, and can be performed using inexpensive hardware, without the need for expensive data management infrastructure or personnel.

**Security Model** As mentioned earlier, the client does not trust either server to preserve data privacy. Each server is honest, but potentially curious: the server may actively monitor all the data that it stores, as well as the queries it receives from the client, in the hope of breaching privacy; it does not, however, act maliciously by providing erroneous service to the client or by altering the stored data.

The client maintains separate, permanent channels of communication to each server. We do not require communication to be encrypted; however, we assume that no eavesdropper is capable of listening in on *both* communication channels. The two servers are assumed to be unable to communicate directly with each other (depicted by the "wall" between them in Figure 7.1) and, in fact, need not even be aware of each other's existence.

Note that the client side is assumed to be completely trusted and secure. There would not be much point in developing a secure database service if a hacker can simply penetrate the client side and transparently access the database. Preventing client-side breaches is a traditional security problem unrelated to privacy-preserving data storage, and we do not concern ourselves with this problem here.

## 7.2.1 Relation Decomposition

We now consider different techniques to partition data across the two servers in the distributed architecture described above. Say the client needs to support queries over the "universal" relation $R(A_1, A_2, \ldots, A_n)$. Since the client itself possesses no permanent storage, the contents of relation $R$ need to be decomposed and stored across the two servers. We require that the decomposition be both *lossless* and *privacy preserving*.

A lossless decomposition is simply one in which it is possible to reconstruct the original relation $R$ using only the contents in the two servers $S_1$ and $S_2$. The exact manner in which such a reconstruction is performed is flexible, and may involve not only traditional relational operators such as joins and unions, but also other user-defined functions, as we discuss shortly. We also require the decomposition to be privacy preserving: the contents stored at server $S_1$ or $S_2$ must not, in themselves, reveal any private information about $R$. We postpone our discussion of what constitutes private information to the next section.

Traditional relation decomposition in distributed databases is of two types:

**Horizontal Fragmentation** Each tuple of the relation $R$ is stored at $S_1$ or $S_2$. Thus, server $S_1$ contains a relation $R_1$, and $S_2$ contains a relation $R_2$ such that $R = R_1 \cup R_2$.

**Vertical Fragmentation** The attributes of relation $R$ are partitioned across $S_1$ and $S_2$. The key attributes are stored at both sites to ensure lossless decomposition. Optionally, other attributes may also be replicated at both sites in order to improve query performance. If the relations at $S_1$ and $S_2$ are $R_1$ and $R_2$ respectively, then $R = R_1 \bowtie R_2$, where $\bowtie$ refers to the natural join on all common attributes.

We believe that horizontal fragmentation is of limited use in enabling privacy-preserving decomposition. For example, a company might potentially store its American sales records as $R_1$ and its European records as $R_2$ to prevent an adversary from gathering statistics about overall sales, thus providing a crude form of privacy. In this paper, we will focus on vertical fragmentation which appears to hold much more promise.

We now discuss a variety of extensions to vertical fragmentation which all aid in making the decomposition privacy preserving.

**Unique Tuple IDs** Vertical partitioning requires a key to be present in both databases in order to ensure lossless decomposition. Since key attributes may themselves be private (and can therefore not be stored in the clear), we may introduce a unique tuple ID for each tuple and replicate this tuple ID alone across the two sites. (This concept is not new. Tuple IDs have been considered as an alternative to key replication to lower update costs in distributed databases [OV99].)

There are a variety of ways to generate unique tuple IDs when inserting new tuples. Tuple IDs could simply be sequence numbers generated independently at the two servers with each new tuple insertion; so long as the client ensures that tuple insertions are atomic, both servers will automatically generate the same sequence number for corresponding tuples. Alternatively, the client could generate random numbers as tuple IDs, potentially performing a query to a server to make sure that the tuple ID does not already exist.

**Semantic Attribute Decomposition** It may be useful to split an attribute $A$ into two separate, but related, attributes $A_1$ and $A_2$, in order to exploit privacy constraints that may apply to $A_1$ but not to $A_2$. To illustrate, consider an attribute representing people's telephone numbers. The area code of a person's number may be sufficiently harmless to be considered public information, but the phone number, in its entirety, is information subject to misuse and should be kept private. In such a case, we may decompose the phone-number attribute into two: a private attribute $A_1$ representing the last seven digits of the phone number, and a public attribute $A_2$ representing the first three digits.

We can immediately see the benefits of such attribute decomposition. Selection queries based on phone numbers, or queries that perform aggregation when grouping by area code, could benefit greatly from the availability of attribute $A_2$. In contrast, in the absence of $A_2$, and if the phone numbers were completely hidden (e.g., by encryption), query processing becomes more expensive.

**Attribute Encoding** It may be necessary to encode an attribute value across both databases so that neither database can deduce the value. For example, consider an attribute that needs to be kept private, say the employee salary. We may encode the salary attribute $A$ as two separate attributes $A_1$ and $A_2$, to be stored in the two databases. The encoding of a salary value $a$ as two separate values $a_1$ and $a_2$ may be performed in different fashions, three of which we outline here:

1. **One-time Pad:** $a_1 = a \bigoplus r, a_2 = r$, where $r$ is a random value;

2. **Deterministic Encryption:** $a_1 = E(a, k), a_2 = k$, where $E$ is a deterministic encryption function such as AES or RSA;

3. **Random Addition:** $a_1 = a + r, a_2 = r$, where $r$ is a random number drawn from a domain much larger than that of $a$.

In all the above cases, observe that we may reconstruct the original value of $a$ using the values $a_1$ and $a_2$. The three different encoding schemes above offer different trade-offs between privacy and efficiency.

The first scheme offers true information-theoretic privacy, since both $a_1$ and $a_2$ consist of random bits that reveal no information about $a$[1]. It also offers fast reconstruction of $a$ from $a_1$ and $a_2$, since only a XOR is required. However, such an encoding rules out any hope of "pushing down" a selection condition on attribute $A$; such conditions can be evaluated only on the client side, after fetching all corresponding $a_1$ and $a_2$ values from the servers and reconstructing value $a$.

The second scheme offers no benefits over the first if the key $k$ is chosen to be an independent random value for each tuple. However, one could use the same key $k$ for *all* tuples, i.e., all values for attribute $A_2$ are equal to the same key $k$. In such a case, we may be able to execute selection conditions on $A$ more efficiently by pushing down a condition on $A_1$. Consider the selection condition $\sigma_{A=v}$. We may evaluate this condition efficiently as follows, assuming key $k$ is stored at $S_2$:

1. Fetch key $k$ from database $S_2$.

2. Send selection condition $\sigma_{A_1=E(v,k)}$ to database $S_1$.

3. Obtain matching tuples from $S_1$.

However, a drawback of such an encoding scheme is a loss in privacy. For example, if two different tuples have the same value in attribute $A$, they will also possess the same encrypted value in attribute $A_1$, thus allowing database $S_1$ to deduce that the two tuples correspond to individuals with identical salaries. A second drawback of the scheme is that it requires encryption and decryption of attribute values at the client side which may be computationally expensive. Finally, such an encryption scheme

[1]We assume that attribute $A$ is of fixed length. Variable-length attributes may leak information about the length of the value unless encoded as fixed-length.

does not help if the selection condition is a range predicate on the attribute rather than an equality predicate.

The third scheme outlined above is useful when handling queries that perform aggregation on attribute $A$. For example, a query that requires the average salary of employees may be answered by obtaining the average of attribute $A_1$ from database $S_1$, and subtracting out the average of attribute $A_2$ from database $S_2$. The price paid for this efficiency is once again a compromise of true privacy: in theory, it is possible for database $S_1$ to guess whether the salary value in a particular tuple is higher than that in another tuple.

**Adding Noise** Another technique for enabling privacy is the addition of "noise" tuples to both databases $S_1$ and $S_2$ in order to improve privacy. Recall that the actual relation $R$ is constructed by the natural join of $R_1$ and $R_2$. We may thus add "dangling tuples" to both $R_1$ and $R_2$ without compromising the lossless decomposition property. The addition of such noise may help provide privacy, say by guaranteeing that the probability of any set of attribute values being part of a "real" tuple is less than a desired constant.

## 7.3 Defining the Privacy Requirements and Achieving It

So far, we have not stated the exact requirements of data privacy. There are many different formulations of data privacy, some of which are harder to achieve than others, e.g., [AS00, Swe02b]. We introduce one particular definition that we believe is appropriate for the database-service context. We refer the reader to Appendix 7.8 to see how our definition captures privacy requirements imposed in legislation such as California bill SB1386 [sb102].

Our privacy requirements are specified as a set of *privacy constraints* $\mathcal{P}$, expressed on the schema of relation $R$[2]. Each privacy constraint is represented by a subset, say

[2]Other notions of privacy, such as k-anonymity [Swe02b], are defined on the actual relation instances and may be harder to enforce in an efficient fashion.

$P$, of the attributes of $R$, and informally means the following:

Let $R$ be decomposed into $R_1$ and $R_2$, and let an adversary have access to the entire contents of either $R_1$ or $R_2$. For every tuple in $R$, the value of at least one of the attributes in $P$ must be completely opaque to the adversary, i.e., the adversary should be unable to infer anything about the value of that attribute. Note that it is permissible for the values of *some* attributes in $P$ to be open, so long as there is at least one attribute completely hidden from the adversary.

We illustrate this definition by an example. Consider a company desiring to store relation $R$ consisting of the following attributes of employees: *Name, Date of Birth (DoB), Gender, Zipcode, Position, Salary, Email, Telephone*. The company may have the following considerations about privacy:

1. *Telephone* and *Email* are sensitive information subject to misuse, even on their own. Therefore both these attributes form singleton privacy constraints and cannot be stored in the clear under any circumstances.

2. *Salary*, *Position* and *DoB* are considered private details of individuals, and so cannot be stored together with an individual's name in the clear. Therefore, the sets {*Name, Salary*}, {*Name, Position*} and {*Name, DoB*} are all privacy constraints.

3. The set of attributes {*DoB, Gender, Zipcode*} can help identify a person in conjunction with other publicly available data. Since we already stated that {*Name, DoB*} is a privacy constraint, we also need to add {*DoB, Gender, Zipcode*} as a privacy constraint.

4. We may also want to prevent an adversary from learning sensitive association rules, for example, between position and salary, or between age and salary. Therefore, we may add two privacy constraints: {*Position, Salary*}, {*Salary, DoB*}.

What does it mean to not be able to "infer the value" of an attribute $A$? We have left this definition intentionally vague to accommodate the requirements of different applications. On one end, we may require true information-theoretic privacy – the adversary must be unable to gain any information about the value of the attribute from examining the contents of either $R_1$ or $R_2$. We may also settle for weaker forms of privacy, such as the one provided by encoding an attribute using encryption or random addition. Neither of these schemes provides true information-theoretic privacy as above, but may be sufficient in practice. In this paper, we will restrict ourselves to the stricter notion of privacy, noting the advantages of the weaker forms where appropriate.

### 7.3.1 Obtaining Privacy via Decompositions

Let us consider how we might decompose data, using the methodologies outlined in Section 7.2, into two relations $R_1$ and $R_2$ so as to obey a given set of privacy constraints. We will restrict ourselves to the case where $R_1$ and $R_2$ are obtained by vertical fragmentation of $R$, fragmented by unique tuple IDs, with some of the attributes possibly being encoded. (We ignore semantic attribute decomposition, as well as the addition of noise tuples. The former may be assumed to have been applied beforehand, while the latter is not useful for obeying our privacy constraints.)

We abuse notation by allowing $R$ to refer to the *set* of attributes in the relation. We may then denote a decomposition of $R$ as $\mathcal{D}(R) = \langle R_1, R_2, E \rangle$, where $R_1$ and $R_2$ are the sets of attributes in the two fragments, and $E$ refers to the set of attributes that are encoded (using one of the schemes outlined in Section 7.2.1). Note that $R_1 \cup R_2 = R$, $E \subseteq R_1$, and $E \subseteq R_2$, since encoded attributes are stored in both fragments. We denote the privacy constraints $\mathcal{P}$ as a set of subsets of $R$, i.e., $\mathcal{P} \subseteq 2^R$.

From our definition of privacy constraints, we may state the following requirement for a privacy-preserving decomposition:

> *The decomposition $\mathcal{D}(R)$ is said to obey the privacy constraints $\mathcal{P}$ if, for every $P \in \mathcal{P}$, $P \nsubseteq (R_1 - E)$ and $P \nsubseteq (R_2 - E)$.*

To understand how to obtain such a decomposition, we observe that each privacy constraint $P$ may be obeyed in two ways:

1. Ensure that $P$ is not contained in either $R_1$ or $R_2$, using vertical fragmentation. For example, the privacy constraint {*Name, Salary*} may be obeyed by placing *Name* in $R_1$ and *Salary* in $R_2$.

2. Encode at least one of the attributes in $P$. For example, a different way to obey the privacy constraint {*Name, Salary*} would be to use, say, a one-time pad to encode *Salary* across $R_1$ and $R_2$. Observe that such encoding is the only way to obey the privacy constraint on singleton sets.

**Example** Let us return to our earlier example and see how we may find a decomposition satisfying all the privacy constraints. We observe that both *Email* and *Telephone* are singleton privacy constraints; the only way to tackle them is to encode both these attributes. The constraints specified in items (2) and (3) may be tackled by vertical fragmentation of the attributes, e.g., $R_1$*(ID, Name, Gender, Zipcode)*, and $R_2$*(ID, Position, Salary, DoB)*, with *Email* and *Telephone* being stored in both $R_1$ and $R_2$.

Such a partitioning satisfies the privacy constraints outlined in item (2) since *Name* is in $R_1$ while *Salary*, *Position* and *DoB* are in $R_2$. It also satisfies the constraint in item (3), since *DoB* is separated from *Gender* and *Zipcode*. However, we are still stuck with the constraints in item (4) which dictate that *Salary* cannot be stored with either *Position* or *DoB*. We cannot fix the problem by moving *Salary* to $R_1$ since that would violate the constraint of item (2) by placing *Name* and *Salary* together.

The solution is to resort to encoding *Salary* across both databases. Thus, the resulting decomposition is $R_1$ ={*ID, Name, Gender, Zipcode, Salary, Email, Telephone*}, $R_2$ = {*ID, Position, DoB, Salary, Email, Telephone*} and $E$ = {*Salary, Email, Telephone*}. Such a decomposition obeys all the stated privacy constraints.

**Identifying the Best Decomposition** It is clear by now that it is always possible to obtain a decomposition of attributes that obeys all the privacy constraints – in the worst case, we could encode all the attributes to obey all possible privacy constraints. A key question that remains is: What is the *best* decomposition to use, where "best" refers to the decomposition that minimizes the cost of the workload being executed against the database?

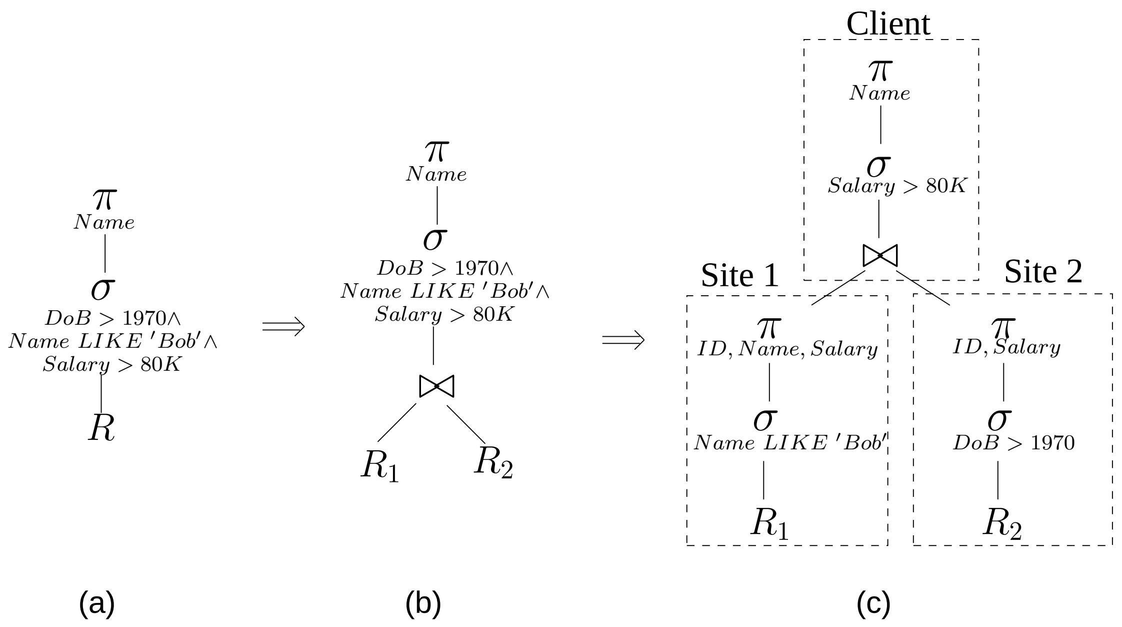


Figure 7.2: *Example of Query Reformulation and Optimization*

An answer to the above question requires us to understand two issues. First, we need to know how an arbitrary query on the original relation $R$ is transformed, optimized and executed using the two fragments $R_1$ and $R_2$. We will address this issue in the next section. We will then consider how to exploit this knowledge in formulating an optimization problem to find the best decomposition in Section 7.5.

## 7.4 Query Reformulation, Optimization and Execution

In this section, we discuss how a SQL query on relation $R$ is reformulated and optimized by the client as sub-queries on $R_1$ and $R_2$, and how the results are combined to produce the answer to the original query. For the most part, it turns out that simple generalizations of standard database optimization techniques suffice to solve our problems. In Sections 7.4.1 and 7.4.2, we explain how to repurpose the well-understood distributed-database optimization techniques [OV99] for use in our context. We discuss the privacy implications of query execution in Section 7.4.3 and present some open issues in query optimization in Section 7.4.4.

### 7.4.1 Query Reformulation

Query reformulation is straightforward and identical to that in distributed databases. Consider a typical conjunctive query that applies a conjunction of selection conditions $C$ to $R$, groups the results using a set of attributes $G$, and applies an aggregation function to a set of attributes $A$. We may translate the logical query plan of this query into a query on $R_1$ and $R_2$ by the following simple expedient: replace $R$ by $R_1 \bowtie R_2$ (with the understanding that the $\bowtie$ operation also replaces encoded pairs of attribute values with the unencoded value). When the query involves a self-join on $R$, we simply replace each occurrence of $R$ by the above-mentioned join[3].

Figure 7.2 shows how a typical query involving selections and projections on $R$ (part (a) of figure) is reformulated as a join query on $R_1$ and $R_2$ (part (b) of figure).

### 7.4.2 Query Optimization

The trivial query plan for answering a query reformulated in the above fashion is as follows: Fetch $R_1$ from $S_1$, fetch $R_2$ from $S_2$, execute all plan operators locally at the client. Of course, such a plan is extremely expensive, as it requires reading and transmitting entire relations across the network.

**Optimizing the Logical Query Plan** The logical query plan is first improved, just as in traditional query optimization, by "pushing down" selection conditions, with minor modifications to account for attribute fragmentation. Consider a selection condition $c$:

- If $c$ is of the form $\langle \mathit{Attr} \rangle$ $\langle \mathit{op} \rangle$ $\langle \mathit{value} \rangle$, and $\langle \mathit{Attr} \rangle$ has not undergone attribute fragmentation, condition $c$ may be pushed down to $R_1$ or $R_2$, whichever contains $\langle \mathit{Attr} \rangle$. (In case $\langle \mathit{Attr} \rangle$ is replicated on both relations, the condition may be pushed down to both.)

[3]Note that since $R$ is considered to be the universal relation, all joins are self-joins.

- If $c$ is of the form $\langle \mathit{Attr1} \rangle$ $\langle \mathit{op} \rangle$ $\langle \mathit{Attr2} \rangle$, and both $\langle \mathit{Attr1} \rangle$ and $\langle \mathit{Attr2} \rangle$ are unfragmented and present in either $R_1$ or $R_2$, the condition may be pushed down to the appropriate relation.

Similarly, projections may also be pushed down to both relations, taking care not to project out tuple IDs necessary for the join. Group-by clauses and aggregates may also pushed down, provided all attributes mentioned in the Group-by and aggregate are unfragmented and present together in either $R_1$ or $R_2$. Self-joins of $R$ with itself, translated into four-way joins involving two copies each of $R_1$ and $R_2$, may be rearranged to form a "bushy" join tree where the two $R_1$ copies, and the two $R_2$ copies, are joined first.

Figure 7.2(c) shows the pushing down of selections and projections to alter the logical query plan. We assume attributes are fragmented as in the example of Section 7.3: $R_1$ contains *Name*, while *DoB* is in $R_2$ and *Salary* is encoded across both relations. Thus, the condition on *Name* is pushed down to $R_1$, while the condition on *DoB* is pushed down to $R_2$. The selection on *Salary* cannot be pushed down since *Salary* is encoded. In addition, we may push down projections as shown in the figure.

**Choosing the Physical Query Plan** Having optimized the logical query plan, the physical plan needs to be chosen, determining how the query execution is partitioned across the two servers and the client. The basic partitioning of the query plan is straightforward: all operators present above the top-most join have to be executed on the client side; all operators underneath the join and above $R_i$ are executed by a sub-query at server $S_i$ for $i = 1, 2$ (shown by the dashed boxes in Figure 7.2).

In the ideal case, it may be possible to push all operators to one of $R_1$ or $R_2$, eliminating the need for a join. Otherwise, we are left with a choice in deciding how to perform the join.

The first option is to send sub-queries to both $S_1$ and $S_2$ in parallel, and join the results at the client. The second option is to send only one of the two sub-queries, say to server $S_1$; the tuple IDs of the results obtained from $S_1$ are then used to perform a *semi-join* with the content on server $S_2$, in addition to applying the $S_2$-subquery to filter $R_2$.

To illustrate with the example of Figure 7.2(c), consider the following two sub-queries:

$Q_1$: `SELECT Name, ID, Salary FROM` $R_1$ `WHERE (Name LIKE Bob)`

$Q_2$: `SELECT ID, Salary FROM` $R_2$ `WHERE (DoB > 1970)`

There are then three options available to the client for executing the query:

1. Send sub-query $Q_1$ to $S_1$, send $Q_2$ to $S_2$, join the results on *ID* at the client, and apply the selection on *Salary*.

2. Send sub-query $Q_1$ to $S_1$. Apply $\pi_{ID}$ to the results of $Q_1$; call the resulting set $\Lambda$. Send $S_2$ the query $Q_3$: `SELECT ID, Position, Salary FROM` $R_2$ `WHERE (ID IN` $\Lambda$`) AND (DoB > 1970)`. Join the results of $Q_3$ with the results of $Q_1$.

3. Send sub-query $Q_2$ to $S_2$. Apply $\pi_{ID}$ to the results of $Q_2$ and rewrite $Q_1$ in an analogous fashion to the previous case.

The first plan may be expensive, since it requires a lot of data to be transmitted from site $S_2$. The second plan is potentially a lot more efficient, since only tuples that match the condition `Name LIKE Bob` are ever transmitted from both $S_1$ and $S_2$. However, there may be a greater delay in obtaining query answers; the results from $S_1$ need to be obtained before a query is sent to $S_2$. In our example, the third plan is unlikely to be efficient unless the company consists only of old people.

We illustrate query optimization with two more examples.

**Example 2** Consider the query:

`SELECT SUM(Salary) FROM R`

Say *Salary* is encoded using *Random Addition* instead of by a one-time pad. In this case, the client may simultaneously issue two subqueries:

$Q_1$: `SELECT SUM(Salary) FROM` $R_1$

$Q_2$: `SELECT SUM(Salary) FROM` $R_2$

The client then computes the difference between the results of $Q_1$ and $Q_2$ as the answer.

**Example 3** Consider the query:

`SELECT Name FROM R`

`WHERE DoB>` 1970 `AND Gender=M`

The client first issues the following sub-query to $S_2$:

$Q_2$: `SELECT ID FROM` $R_2$ `WHERE DoB>` 1970

After it gets the ID list $\Lambda$ from $Q_2$ , it sends $S_1$ the query: `SELECT Name FROM` $R_1$ `WHERE Gender=M AND ID in` $\Lambda$. Note that the alternative plan – sending sub-queries in parallel to both $S_1$ and $S_2$ – may be more efficient if the condition `DoB>` 1970 is highly unselective.

### 7.4.3 Query Execution and Data Privacy

One question that may arise from our discussion of query execution is: Is it possible for an adversary monitoring activity at either $S_1$ or $S_2$ to breach privacy by viewing the *queries* received at either of the two databases?

We claim that the answer is ‘No’. Observe that when using simple joins as the query plan, the sub-queries sent to $S_1$ and $S_2$ are dependent only on the original query and not on the data stored at either location; thus, the sub-queries do not form a “covert channel” by which information about the content at $S_1$ could potentially be transmitted to $S_2$ or vice versa.

However, when semi-joins are used, we observe that the query sent to $S_2$ is influenced by the results of the sub-query sent to $S_1$. Therefore, a legitimate concern might be that the sub-query sent to $S_2$ leaks information about the content stored at $S_1$. We avoid privacy breaches in this case by ensuring that only *tuple IDs* are carried over from the results of $S_1$ into the query sent to $S_2$. Since knowledge of some tuple IDs being present in $S_1$ does not help an adversary at $S_2$ in inferring anything about other attribute values, such a query execution plan continues to preserve privacy.

### 7.4.4 Discussion

The above discussion does not by any means exhaust all query-optimization issues in the two-server context. We now list some areas for future work, with preliminary observations about potential solutions.

**Maintaining Statistics for Optimization** Our discussion of the space of query plans implicitly assumed that the client has sufficient database statistics at hand, and a sufficiently good cost model, for it to choose the best possible plan for the query. More work is required to validate both the above assumptions.

For example, one question is to understand where the client obtains its statistics from. Statistics on the individual relations $R_1$ and $R_2$ could be obtained directly from the servers $S_1$ and $S_2$. The statistics may be cached on the client side in order to avoid having to fetch them from the servers for each optimization. There may potentially be statistics, e.g., multi-dimensional histograms, that require knowledge of both relations $R_1$ and $R_2$ in order to be maintained. If necessary, such statistics could conceivably be maintained on the client side and may be constructed by means of appropriate SQL sub-queries sent to the two servers.

**Supporting Other Decomposition Techniques** Our discussion of query optimization so far has only covered the case of vertical fragmentation with attribute encoding using one-time pads or random addition. It is possible to optimize the query plans further when performing attribute encoding by deterministic encryption, or when using semantic attribute decomposition.

For example, when an attribute is encrypted by a deterministic encryption function, it is possible to push down selection conditions of the form $\langle attr \rangle = \langle const \rangle$, by obtaining the encryption key from one database and encrypting $\langle const \rangle$ with this key before pushing down the condition.

When an attribute is decomposed by semantic decomposition, the resulting functional dependencies across the decomposed attributes may potentially be used to push down additional selection conditions. To illustrate, consider a *PhoneNumber (PN)* attribute which is decomposed into *AreaCode (AC)* and *LocalNumber(LN)*. A selection condition of the form $\sigma_{PN=5551234567}$ could still be pushed down partially as $\sigma_{AC=555}$ and $\sigma_{LN=1234567}$. (Note that the original condition cannot be eliminated, and still needs to be applied as a filter at the end.) Such rewriting depends on the nature of the semantic decomposition; the automatic application of such rewriting therefore requires support for specifying the relationship between attributes in a simple fashion.

## 7.5 Identifying the Optimal Decomposition

Having seen how queries may be executed over a decomposed database, our next task at hand is to identify the *best* decomposition that minimizes query costs. Say, a workload $W$ consisting of the actual queries to be executed on $R$ is available. We may then think of the following brute-force approach:

For each possible decomposition of $R$ that obeys the privacy constraints $\mathcal{P}$:

- Optimize each query in $W$ for that decomposition of $R$, and
- Estimate the total cost of executing all queries in $W$ using the optimized query plans.

We may then select that decomposition which offers the lowest overall query cost. Observe that such an approach could be prohibitively expensive, since there may be an extremely large number of legitimate decompositions to consider, against each of which we need to evaluate the cost of executing all queries in the workload.

To work around this difficulty, we attempt to capture the effects of different decompositions on query costs in a more structured fashion, so that we may efficiently prune the space of all decompositions without actually having to evaluate each decomposition independently. A standard framework to capture the costs of different decompositions, for a given workload $W$, is the notion of the *affinity matrix* [OV99] $M$, which we adopt and generalize as follows:

1. The entry $M_{ij}$ represents the "cost" of placing the unencoded attributes $i$ and $j$ in different fragments.
2. The entry $M_{ii}$ represents the "cost" of encoding attribute $i$ across both fragments.

We assume that the cost of a decomposition may be expressed simply by a linear combination of entries in the affinity matrix. Let $R = \{A_1, A_2, \dots A_n\}$ represents the original set of $n$ attributes, and consider a decomposition of $\mathcal{D}(R) = \langle R_1, R_2, E\rangle$. Then, we assume that the cost of this decomposition $C(\mathcal{D})$ is $\sum_{i\in(R_1-E),j\in(R_2-E)} M_{ij}$

$+ \sum_{i \in E} M_{ii}$. ( For simplicity, we do not consider replicating any unencoded attribute, other than the tupleID, at both sites.)

In other words, we add up all matrix entries corresponding to pairs of attributes that are separated by fragmentation, as well as diagonal entries corresponding to encoded attributes, and consider this sum to be the cost of the decomposition.

Given this simple model of the cost of decompositions, we may now define an optimization problem to identify the best decomposition:

*Given a set of privacy constraints $\mathcal{P} \subseteq 2^R$ and an affinity matrix $M$, find a decomposition $\mathcal{D}(R) = \langle R_1, R_2, E \rangle$ such that*

*(a) $\mathcal{D}$ obeys all privacy constraints in $\mathcal{P}$, and*

*(c) $\sum_{i,j:i \in (R_1 - E), j \in (R_2 - E)} M_{ij} + \sum_{i \in E} M_i$ is minimized.*

We are left with two questions:

- How is the affinity matrix $M$ generated from a knowledge of the query workload?
- How can we solve the optimization problem?

We address the first question in Appendix 7.9, where we present heuristics for generating the affinity matrix. We discuss the second question next.

## 7.5.1 Solving the Optimization Problem

We may define our optimization problem as the following hypergraph-coloring problem:

> We are given a complete graph $G(R)$, with both vertex and edge weights defined by the affinity matrix $M$. (Diagonal entries stand for vertex weights.) We are also given a set of privacy constraints $\mathcal{P} \subseteq 2^R$, representing a hypergraph $H(R, \mathcal{P})$ on the same vertices. We require a 2-coloring of the vertices in $R$ such that (a) no hypergraph edge in $H$ is monochromatic, and (b) the weight of bichromatic graph edges in $G$ is minimized. The twist is that we are allowed to delete any vertex in $R$ (and all hyperedges in $\mathcal{P}$ that contain the vertex) by paying a price equal to the vertex weight.

Observe that coloring a vertex is equivalent to placing it in one of the two partitions. Deleting the vertex is equivalent to encoding the attribute; so, all privacy constraints associated with that attribute are satisfied by the vertex deletion. Also observe that vertex deletion may be necessary, since it is not always possible to 2-color a hypergraph.

The above problem is very hard to solve – even if we remove the feature of vertex deletion (allowing encoding), say by guaranteeing that the hypergraph is 2-colorable. In fact, much more restrictive special cases are NP-hard, even to approximate, as the following result shows:

> It is NP-hard to color a 2-colorable, 4-uniform hypergraph using only $c$ colors for any constant c [GHS00].

In other words, even if all privacy constraints were 4-tuples of attributes, and it is known that there exists a partitioning of attributes into two sets that satisfies all constraints, it is NP-hard to partition the attributes into *any* fixed number of sets, let alone two, to satisfy all the constraints!

Given the hardness of the hypergraph-coloring problem, we consider three different heuristics to solve our optimization problem. All our heuristics utilize the following two solution components:

**Approximate Min-Cuts** If we were to ignore the privacy constraints for a moment, observe that the resulting problem is to two-color the vertices to minimize the weight of bichromatic edges in $G(R)$; this is equivalent to finding the min-cut in $G$ (assuming that at least one vertex needs to be of each of the two colors). This problem can be solved optimally in polynomial time, but we will be interested in a slightly more general version: We will require *all* cuts of the graph that have a weight within a specified constant factor of the min-cut.

Intuitively, we want to produce a lot of cuts that are near-optimal in terms of their quality, and we will choose among the cuts to pick one that helps satisfy the most privacy constraints. This approximate min-cut problem can still be solved efficiently in polynomial time using an algorithm based on edge contraction [KS96]. (Note that this also implies that the number of cuts produced by the algorithm is only polynomially large.)

**Approximate Weighted Set Cover** Our second component uses a well-known tool in order to tackle the satisfaction of the privacy constraints via vertex deletion. Let us ignore the coloring problem and consider the following problem instead: Find the minimum weight choice of vertices to delete so that all hypergraph edges are removed, i.e., each set $P \in \mathcal{P}$ loses at least one vertex.

This is the *minimum weighted set-cover* problem, and the best known solution is to use the following greedy strategy: keep deleting the vertex that has the lowest cost per additional set that it covers, until all sets are covered. This greedy strategy offers a $(1 + \log |\mathcal{P}|)$-approximation to the optimal solution [Joh73, Chv79] and will be used by us in our heuristics.

We now present three different heuristics which utilize the above two components:

**Heuristic 1** Our first heuristic is to solve the optimization problem in three phases:

1. Ignore fragmentation, and delete vertices to cover all the constraints using **Approximate Weighted Set Cover**. Call the set of deleted vertices $E$.

2. Consider the remaining vertices, and use **Approximate Min-Cuts** to find different 2-colorings of the vertices, all of which approximately minimize the weight of the bichromatic edges in $G$.

3. For each of the 2-colorings obtained in step (2): Find all deleted vertices that are present only in bichromatic hyperedges, and consider "rolling back" their deletion, and coloring them instead, to obtain a better solution.

4. Choose the best of (a) the solution from step (3) for each of the 2-colorings, and (b) the decomposition $\langle R - E, E, E \rangle$.

In the first step, we cover all the privacy constraints by ensuring that at least one attribute in each constraint is encoded. Note that this step leads us directly to one possible decomposition: place all deleted vertices in both fragments, and all the remaining vertices in one of the two fragments. Call this decomposition $\mathcal{D}_1$.

In the next two steps, we attempt to improve on $\mathcal{D}_1$ by avoiding encrypting all these attributes, hoping to use fragmentation to cover some of the constraints instead. To this end, we find different approximate min-cuts in step (2), each of which fragments the attributes differently. In each fragmentation, we try to roll back some of the attribute encoding (vertex deletion) that had earlier been necessary to cover some constraints, but is no longer needed thanks to the fragmentation satisfying the constraints instead.

Finally, we compare the quality of the different solutions obtained from step (3), with the basic solution $\mathcal{D}_1$ obtained directly from step 1, and select the best of the lot. Note that the entire heuristic runs in polynomial time, because the number of different cuts considered is only a polynomial function of $|R|$.

**Heuristic 2** Our second heuristic reverses the order in which fragmentation (2-coloring) and encoding (deletion) are attempted. We first apply **Approximate Min-Cuts** to the original graph $G(R)$ to obtain a set of possible cuts. For each such cut, we perform the following steps:

(a) Some of the privacy constraints are already satisfied by the fragmentation; we therefore delete these constraints from $\mathcal{P}$,

(b) We apply **Approximate Weighted Set Cover** to the modified $\mathcal{P}$, deleting vertices until all constraints are satisfied.

Finally, we may once again compare the solutions obtained from each cut and select the best one.

**Heuristic 3** The third approach we consider is to interleave the execution of our approximate min-cut and set-cover components, instead of just using one after the other. We start with some 2-coloring obtained by running **Approximate Min-cuts**. We then repeat the following steps until all constraints are satisfied:

1. Use **Approximate Set Cover** to greedily select *one* vertex to delete. (Note that we only delete one vertex, instead of deleting as many as necessary to satisfy all constraints.)

2. Having deleted this vertex, re-run **Approximate Min-Cuts** and attempt to find a 2-coloring that satisfies even more constraints than the current coloring.

(If we can't find such a coloring, retain the current coloring.)

Observe that the above heuristic uses many more invocations of the min-cut algorithm in order to recompute colorings after each vertex deletion. To obtain some intuition as to why this heuristic is useful, consider some vertex $v$ which has high-weight graph edges to some of its neighbors $v_1, v_2, \dots v_k$. A min-cut on the original graph will tend to force $v$ together with all these neighbors (i.e., all these vertices will have the same color) since the edges from $v$ to $v_1, v_2, \dots v_k$ are all of high weight. However, once $v$ is deleted, the nature of the coloring may change dramatically; the different vertices $v_1, v_2, \dots v_k$ may no longer need to have the same color, which opens the door for colorings that can satisfy many more privacy constraints (specifically, constraints that can be satisfied by separating some of the vertices in $v_1, v_2, \dots v_k$).

### 7.5.2 Discussion

There are many open questions surrounding the above decomposition problem. One question is to understand the relative performance of our different design heuristics on different relation schemata and privacy constraints. Another is to develop better theoretical approaches to the optimization problem. Formulating the optimization problem itself is based on a number of heuristics (discussed in Appendix 7.9) which are also open to improvement. The scope of the optimization problem may also be expanded in a number of different directions.

For example, we could allow attributes to be replicated across partitions, trying to exploit such replication to lower query costs. In the terminology of the optimization problem, vertices are allowed to take on both colors. Edges in $G$ emanating from a vertex $v$ with two colors will not be considered bichromatic; however, hyperedges involving $v$ will need to be bichromatic even when ignoring $v$.

Another extension is to deal with constraints imposed by functional dependencies, normal forms and multiple relations. For example, we may want our decomposition to be dependency-preserving, which dictates that functional dependencies should not be destroyed by data partitioning. Different partitioning schemes may have different impacts on the cost of checking various constraints. Factoring these issues into the optimization problem is a subject for future work.

Finally, expanding the definition of the optimization problem to accommodate the space of different encoding schemes for each attribute is also an area as yet unexplored.

## 7.6 Related Work

**Secure Database Services** As discussed in the introduction, the outsourcing of data management has motivated the model where a DBMS provides reliable storage and efficient query execution, while not knowing the contents of the database [HIM02]. Schemes proposed so far for this model encrypt data on the client side and then store the encrypted database on the server side [HILM02, HH04, AKSX04]. However, in order to achieve efficient query processing, all the above schemes only provide very weak notions of data privacy. In fact a server that is secure under formal cryptographic notions can be proved to be hopelessly inefficient for data processing [KC04]. Our architecture of using multiple servers helps to achieve both efficiency and provable privacy together.

**Trusted Computing** With trusted computing [TCG03], a tamper-proof secure co-processor could be installed on the server side, which allows executing a function while hiding the function from the server. Using trusted tamper-proof hardware for enabling secure database services has been proposed in [KC04]. However, such a scheme could involve significant computational overhead due to repeated encryption and decryption at the tuple level. Understanding the role of tamper-proof hardware in our architecture remains a subject of future work.

**Secure Multi-party Computation** Secure multi-party computation [Yao86, GMW87] discusses how to compute the output of a function whose inputs are stored at different parties, such that each party learns only the function output and nothing about the inputs of the other parties. In our context, there are two parties – the server and the client – with the server's input being encrypted data, the client's input being the encryption key, and the function being the desired query. In principle, the client and the server could then engage in a one-sided secure computation protocol to compute the function output that is revealed only to the client. However, “in principle” is the operative phrase, as the excessive communication overhead involved makes this approach even more inefficient than the trivial scheme in which the client fetches the entire database from the server. More efficient specialized secure multi-party computation techniques have been studied recently[LP00, FNP04, AMP04]. However all of this work is to enable different organizations to securely analyze their combined data, rather than the client-server model we are interested in.

**Privacy-preserving Data Mining** Different approaches for privacy-preserving data mining studied recently include: (1) *perturbation* techniques [AS00, AA01, EGS03, DN03, DN04] (2) *query restriction/auditing* [CO82, DJL79, KPR00] (3) *k-anonymity* [Swe02b, MW04, AFK$^+$05a]. However, research here is motivated by the need to ensure individual privacy while at the same time allowing the inference of higher-granularity patterns from the data. Our problem is rather different in nature, and the above techniques are not directly relevant in our context.

**Access Control** Access control is used to control which parts of data can be accessed by different users. Several models have been proposed for specifying and enforcing access control in databases [CFMS95]. Access control does not solve the problem of maintaining an untrusted storage server as even the the administrator or an insider having complete control over the data at the server is not trusted by the client in our model.

## 7.7 Conclusions

We have introduced a new distributed architecture for enabling privacy-preserving outsourced storage of data. We demonstrated different techniques that could be used to decompose data, and explained how queries may be optimized and executed in this distributed system. We introduced a definition of privacy based on hiding sets of attribute values, demonstrated how our decomposition techniques help in achieving privacy, and considered the problem of identifying the best privacy-preserving decomposition. Given the increasing instances of database outsourcing, as well as the increasing prominence of privacy concerns as well as regulations, we expect that our architecture will prove useful both in ensuring compliance with laws and in reducing the risk of privacy breaches.

We have built a prototype having algorithms for data partitioning, query decomposition, and execution [FGGM+07].

## 7.8 Addendum: Extract from California SB 1386

The California Senate Bill SB 1386, which went into effect on July 1, 2003, defines what constitutes personal information of individuals, and mandates various procedures to be followed by state agencies and businesses in California in case of a breach of data security in that organization. We present below its definition of personal information, observing how it is captured by our definition of privacy constraints (the italics are ours):

> For purposes of this section, "personal information" means an individual's first name or first initial and last name in combination with any one or more of the following data elements, *when either the name or the data elements are not encrypted*:
>
> (1) Social security number.
>
> (2) Driver's license number or California Identification Card number.
>
> (3) Account number, credit or debit card number, in combination with any required security code, access code, or password that would permit access to an individual's financial account.
>
> For purposes of this section, "personal information" does not include publicly available information that is lawfully made available to the general public from federal, state, or local government records. [sb102]

## 7.9 Addendum: Computing the Affinity Matrix

Let us revisit the definition of the affinity matrix to examine its semantics: the entry $M_{ij}$ is required to represent the "cost" of placing attributes $i$ and $j$ in different partitions of a decomposition, while the entry $M_{ii}$ represents the "cost" of encoding attribute $i$, with the overall cost of decomposition being expressed as a linear combination of these entries.

Note that it is likely impossible to obtain a matrix that accurately captures the costs of all decompositions. The costs of partitioning different pairs of attributes are unlikely to be independent; placing attributes $i$ and $j$ in different partitions may have different effects on query costs, depending on how the other attributes are placed and encoded. Our objective is to come up with simple heuristics to obtain a matrix that does a reasonable job of capturing the *relative* costs of different decompositions.

Similar matrices are used to capture costs in other contexts too, e.g., the allocation problem in distributed databases [OV99]. Our problem is complicated somewhat by the fact that we need to account for the effects of attribute encoding on query costs, as well as the interactions between relation fragmentation and encoding.

**A First Cut** As a first cut, we may consider the following simple way to populate the affinity matrix from a given query workload, along the lines of earlier approaches [OV99]:

- $M_{ij}$ is set to be the number of queries that reference both attributes $i$ and $j$.
- $M_{ii}$ is set to be the number of queries involving attribute $i$.

Of course this simple heuristic ignores many issues: different queries in the workload may have different costs and should not be weighted equally; the effect of partitioning attributes $i$ and $j$ may depend on how $i$ and $j$ are used in the query, e.g., in a selection condition, in the projection list, etc.; the cost of encoding an attribute may be very different from that of partitioning two attributes, so that counting both on the same scale may be a poor approximation.

In order to improve on this first cut, we dig deeper to understand the effects of fragmentation and encoding on query costs.

### 7.9.1 The Effects of Fragmentation

Let us consider a query that involves attributes $i$ and $j$ and evaluate the effect of a fragmentation that separates $i$ and $j$ on the query. We may make the following observations:

- If $i$ and $j$ are the only attributes referenced in the query, the fragmentation forces the query to touch both databases, and increases the communication cost for the query; the extra communication cost is proportional to the number of tuples satisfying the most selective conditions on one of the two attributes.
- If attributes other than $i$ and $j$ are involved in the query, it is possible that the query may have to touch both databases even if $i$ and $j$ were held together, since the separation of other attributes may be the culprit. Therefore, the query cost that may be attributed to $M_{ij}$ should be only a fraction of the query overhead caused by fragmentation.
- If $i$ or $j$ is part of a GROUP BY clause, fragmentation makes it impossible to apply the GROUP BY, making the query overhead very high.

Using the above observations, we may devise a scheme to populate the matrix entries $M_{ij}$ for $i \neq q$. Each entry $M_{ij}$ is computed as a sum of "contributions" from each query that references both $i$ and $j$. The contribution of a query $Q$ to $M_{ij}$, for any pair $i$ and $j$ referenced in $Q$, is a measure of the fraction of extra tuple fetches from disk, and transmissions across the network, that are induced by the partitioning of $i$ and $j$. We define this contribution as follows: ( Let $s_i$ be the selectivity of $Q$, ignoring all conditions involving $i$, and $s_j$ be its selectivity ignoring all conditions involving $j$.)

- If $Q$ involves either $i$ or $j$ in a GROUP BY, or a selection condition, the contribution of $Q$ to $M_{ij}$ is set to $\min(s_i, s_j)$.
- If $Q$ involves $i$ and $j$ only in the projection list, the contribution of $Q$ to $M_{ij}$ is set to $\min(s_i, s_j)/n$ where $n$ is the number of attributes referenced in $Q$'s projection list.

Note that the approach above requires the estimation of query selectivities; this may be performed using standard database techniques, i.e., using a combination of selectivity estimates from histograms, independence assumptions, and *ad hoc* guesses about the selectivity of predicates [SAC+79].

### 7.9.2 The Effects of Encoding

Let us now consider the effects of encoding attributes on query costs. We make the following observations about the effects of encoding attribute $i$ on a query $Q$: (We will assume that encoding is performed using one-time pads or random addition.)

- If $Q$ contains a selection condition involving $i$, the condition cannot be pushed down; the overhead due to this is proportional to the selectivity of the query ignoring the conditions on $i$.
- If $Q$ involves $i$ only in the projection list, there may be additional overhead equal to the cost of fetching $i$ from both sides.
- If $Q$ involves $i$ in the GROUP BY clause, grouping cannot be pushed down, and may cause additional overhead.
- If $Q$ involves $i$ only as an attribute to be aggregated, the use of Random Addition for encoding ensures that the overhead of encoding is low.

From these observations, we use the following rules to determine the contributions of $Q$ to $M_{ii}$: (Again, we let $s_i$ be the selectivity of the query ignoring predicates involving $i$.)

- If $i$ is in a selection condition or a GROUP BY clause, the contribution to $M_{ii}$ is set to $s_i$.
- Else, if $i$ is in the projection list, the contribution to $M_{ii}$ is set to $1/n$, where $n$ is the total number of attributes referenced by $Q$.

# Chapter 8

# Conclusions

This thesis addresses three problems.

Data sanitization for outsourcing for research or software development was covered in chapters  2, 3, 4, 5. These chapters provide not only models and theory for these problems but also algorithms and implementations as well as a contribution to commercial software in chapter 5. One area of future research involves refining the current notions of privacy. The notion of privacy is quite clearly defined in cryptography, for example, semantic security [GM82]. But semantically secure encrypted data can provably not be used without decryption, i.e. this encrypted data has no utility. Several middle grounds were proposed for this problem in many recent directions of research  [EGS03, AST05, Dwo06, AA01]. An interesting direction of research is to find out whether a unifying view of privacy can be obtained. Or maybe two views, one for categorical or discrete data and the other for numeric or continuous data. This would be a wonderful contribution to research in this area. The main contribution of MASKETEER™ was that it brought together many of the techniques for data sanitization together in one tool. The user of the tool therefore can choose what kind of sanitization to apply to the various tools – randomization, perturbation, shuffling, encryption and $k$-anonymity. Maybe this is the way research and development in this area should proceed: provide all the techniques to the user so that the user would have a choice on what to techniques to use for the dataset in question.

We provide models and theory for auditing SQL query logs in chapter 6. However we are removed from practice in this chapter. This is not to say there are no tools for auditing logs today: see for example [Log, Aud00]. A direction of research in this area would be to bring theory and practice closer and develop practical algorithms to check semantic or syntactic suspiciousness.

In chapter 7 we propose solutions to distribute data for data privacy. We already have built a simple implementation for this [FGGM+07]. This presently handles simple select-project-join SQL queries and we are presently extending it to group-by queries. An interesting direction for future research would be to extend this to handle general SQL queries which may have nested sub-queries.

Outsourcing data management has been presently considered by some products in the market today. This may be very close to the hardware layer as in Amazon Elastic Compute Cloud [Ama], or closer to applications as in Salesforce [Sal] or Google applications for your domain [Goo]. None of these solutions consider privacy as a primary pillar as in our proposal. It would be an interesting direction of research whether privacy can be built into these architectures as a primary pillar.

Data privacy thus seems an interesting direction of research both from a fundamental contributions point of view as well as building software/hardware products. This thesis hopefully broadens the horizon of both theory and practice in the field of data privacy, but we do hope this is just the beginning of a lot more of work in this area.

# Bibliography

[AA01] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving datamining algorithms. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2001.

[ABF$^+$04] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant. Auditing compliance with a hippocratic database. In *Proceedings of the International Conference on Very Large Data Bases*, September 2004.

[ABG$^+$05] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *Conference on Innovative Data Systems Research*, 2005.

[AES03] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

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

[AFK$^+$05b] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-Anonymity. *Journal of Privacy Technology*, 20051120001, 2005. Earlier version appeared in Proc. of the Intl. Conf. on Database Theory (ICDT 2005).

[AFK$^+$06] G. Aggarwal, T. Feder, K. Kenthapadi, R. Panigrahy, D. Thomas, and A. Zhu. Clustering for privacy. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2006.

[Agg05] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 2005 International Conference on Very Large Data Bases*, pages 901–909, 2005.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.

[Ama] Amazon. Amazon elastic compute cloud. `aws.amazon.com/ec2`.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Privacy preserving computation of the k-th ranked element. In *EUROCRYPT*, 2004.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.

[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, May 2000.

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

[Aud00] SQL Server 2000 Auditing, 2000. Available from URL: `https://www.microsoft.com/technet/security/prodtech/sqlserver/sql2kaud.mspx`.

[AW89] Nabil R. Adam and John C. Wortmann. Security control methods for statistical databases: A comparative study. In *ACM Computing Surveys, Vol21, No 4*, December 1989.

[BA05] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the International Conference on Data Engineering*, pages 217–228, 2005.

[Bau06] Katrina Baum. First estimates from the national crime victimization survey: Identity theft, 2004. *Bureau of Justice Statistics Bulletin*, April 2006. Available from URL: `http://www.ojp.usdoj.gov/bjs/pub/pdf/it04.pdf`.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. Available from URL: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[Bro00] Michelle Brown. Identity theft victim stories: Verbal testimony by michelle brown, July 2000. Privacy Rights ClearingHouse. Available from URL: `http://www.privacyrights.org/cases/victim9.htm`.

[Bur] U.S. Census Bureau. Public use microdata sample (PUMS). http://www.census.gov/acs/www/Products/PUMS/.

[Cam] Camouflage. http://www.datamasking.com/.

[CDM$^+$05] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *2nd Theory of Cryptography Conference (TCC)*, pages 363–385, 2005.

[CDMT05] Shuchi Chawla, Cynthia Dwork, Frank McSherry, and Kunal Talwar. On the utility of privacy-preserving histograms. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[Cen] US Census. Accuracy of the US census data. Available from URL: `http://www.census.gov/acs/www/UseData/Accuracy/Accuracy1.htm`.

[CFMS95] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Principles of Distributed Database Systems*. Addison Wesley, 1995.

[CGGM03] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[Che52] H. Chernoff. Asymptotic efficiency for tests based on the sums of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[Chi86] F. Chin. Security problems on inference control for sum, max, and min queries. *J. ACM*, pages 451–464, 1986.

[Chv79] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[CKL$^+$03] C. Clifton, M. Kantarcioglu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, January 2003.

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

[CO82] F. Chin and G. Ozsoyoglu. Auditing and inference control in statistical databases. In *IEEE TSE, 8(6)*, 1982.

[Dal86] T. Dalenius. Finding a needle in a haystack or identifying anonymous census records. In *Journal of Official Statistics (2)*, pages 329–336, 1986.

[DJL79] D. Dobkin, A. Jones, and R. Lipton. Secure databases: Protection against user influence. In *ACM TODS, 4(1)*, 1979.

[DLP$^+$06] Prasenjit Das, Sachin Lodha, Nikhil Patwardhan, Sharada Sundaram, and Dilys Thomas. Data privacy using MASKETEER. Research Report, Tata Consultancy Services, Pune, India, 2006.

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

[DN04] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Proc. CRYPTO*, 2004.

[Dwo06] Cynthia Dwork. Differential privacy. pages 1–12, 2006.

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the ACM Symposium on Principles of Database Systems*, June 2003.

[ESAG02] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.

[FGGM$^+$07] Tomas Feder, Vignesh Ganapathy, Hector Garcia-Molina, Rajeev Motwani, and Dilys Thomas. Algorithms for distributing data, parititioning and executing queries for a secure database. 2007.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[FT03] J. D. Ferrer and V. Torra. Disclosure risk assesment in statistical microdata protection via advanced record linkage. In *Statistics and Computing*, pages 343–354, 2003.

[GHS00] V. Guruswami, J. Hastad, and M. Sudan. Hardness of approximate hypergraph coloring. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[Gib01] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proceedings of the International Conference on Very Large Data Bases*, pages 541–550, 2001.

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

[GL] G. Golub and C. V. Loan. Matrix computations.

[GLB] GLB. Gramm-Leach-Bliley Act. Available from URL: `http://www.ftc.gov/privacy/privacyinitiatives/glbact.html`.

[GM82] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. 1982.

[GMM00] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 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 – a completeness theorem for protocols with a honest majority. In *Proceedings of the 1987 Annual ACM Symp. on Theory of Computing*, 1987.

[Goo] Google. Google apps for your domain. `http://www.google.com/a/`.

[HFH99] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proc. of the 1st ACM Conference on Electronic Commerce*, pages 78–86, Denver, Colorado, November 1999.

[HH04] S. Mehrotra H. Hacigumus, B. Iyer. Efficient execution of aggregation queries over encrypted relational databases. In *Proc. DASFAA*, 2004.

[HILM02] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.

[HIM02] H. Hacigumus, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proceedings of the International Conference on Data Engineering*, 2002.

[HIP] HIPAA. Health Information Portability and Accountability Act. Available from URL: `http://www.hhs.gov/ocr/hipaa/`.

[HK71] K. Hoffman and R. Kunze. Linear algebra. Prentice-Hall Inc, 1971.

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

[HT98] C. A. J. Hurkens and S. R. Tiourine. Model and methods for the microdata protection problem. In *Journal of Official Statistics*, 1998.

[IBM] IBM. Privacy is good for business. Available from URL: `http://www-306.ibm.com/innovation/us/customerloyalty/harriet_pearson_interview.shtml`.

[Iye02] V. Iyengar. Transforming data to satisfy privacy constraints. In *8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 279–288, 2002.

[JBIP93] G. Kortsarz J. Bar-Ilan and D. Peleg. How to allocate network centers. In *Journal of Algorithms*, pages 385–415, 1993.

[Joh73] David S. Johnson. Approximation algorithms for combinatorial problems. In *Proc. 5th annual ACM Symposium on Theory of Computing(STOC)*, 1973.

[Jr.] R. A. Moore Jr. Controlled data-swapping techniques for masking public use microdata sets. In *US Bureau of Census, Report*.

[JV99] K. Jain and V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the*

*Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[KC04] Murat Kantarcioglu and Chris Clifton. Security issues in querying encrypted data. Technical Report TR-04-013, Purdue University, 2004.

[KM00] D. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. In *Proceedings of the 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 ACM Symposium on Principles of Database Systems*, June 2005.

[KPR00] Jon M. Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. Auditing boolean attributes. In *Symposium on Principles of Database Systems*, pages 86–91, 2000.

[KS96] David Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, July 1996.

[KS00] S. Khuller and Y. Sussmann. The capacitated k-center problem. In *SIAM Journal on Discrete Mathematics*, pages 403–418, 2000.

[LCL85] Chong K. Liew, Uinam J. Choi, and Chung J. Liew. A data distortion by probability distribution. *ACM Transactions on Database Systems*, 10(3):395–411, 1985.

[LDR05a] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2005.

[LDR05b] Kristin Lefevre, David J. Dewitt, and Raghu Ramakrishnan. Incognito: efficient full domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60, 2005.

[Log] ApexSQL Log. Available from URL: `http://www.apexsql.com/sql_tools_log.asp`.

[LP00] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

[LT06] Sachin Lodha and Dilys Thomas. Probabilistic anonymity. Research Report, Tata Consultancy Services, Pune, India, 2006.

[Mas] Data Masker. http://www.datamasker.com/.

[MG06] A. Machanavajjhala and J. Gehrke. On the efficiency of checking perfect privacy. In *PODS*, 2006.

[MKGV06] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the International Conference on Data Engineering*, page 24, 2006.

[MNT07] Rajeev Motwani, Shubha Nabar, and Dilys Thomas. Auditing batches of SQL queries. In *PDM workshop with ICDE*, 2007.

[Mor02] J.P. Morgan signs outsourcing deal with IBM. ComputerWorld, Dec 30, 2002.

[Mot89] A. Motro. An access authorization model for relational databases based on algebraic manipulation of view definitions. In *ICDE*, 1989.

[MP78] I. Munro and M. Paterson. Selection and sorting with limited storage. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 253–258, 1978.

[MRL99] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 251–262, 1999.

[MS04] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, 2004.

[MW04] A. Meyerson and R. Williams. On the complexity of optimal *k*-anonymity. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 223–228, June 2004.

[NMK$^+$06] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards robustness in query auditing. In *VLDB*, 2006.

[O'B04] Robert O'Barrow Jr. Advertiser charged in massive database theft. The Washington Post, July 22, 2004.

[OV99] M. Tamer Ozsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 2nd edition, 1999.

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

[RH02] S. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the International Conference on Very Large Data Bases*, 2002.

[RMSR04] S. Rizvi, A. Medelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD*, 2004.

[RS00] A. Rosenthal and E. Sciore. View security as the basis for data warehouse security. In *International Workshop on Design and Management of Data Warehouses*, 2000.

[RS01] A. Rosenthal and E. Sciore. Administering permissions for distributed data: Factoring and automated inference. In *IFIP 11.3 Working Conference in Database Security*, 2001.

[RSD99] A. Rosenthal, E. Sciore, and V. Doshi. Security administration for federations, warehouses, and other derived data. In *IFIP WG11.3 Conference on Database Security*, 1999.

[Rud87] Walter Rudin. *Real and Complex Analysis.* McGraw-Hill, 1987.

[SAC$^+$79] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In *Proc. SIGMOD*, pages 23–34, 1979.

[Sal] Salesforce. On-demand customer relationship management. `http://www.salesforce.com/`.

[sb102] *California Senate Bill SB 1386*, September 2002.

[Sof] Princeton Softech. http://www.princetonsoftech.com/Solutions/ PCIStandards.asp.

[SOX] SOX. Sarbanes-Oxley Act. Available from URL: `http://www.sec.gov/about/laws/soa2002.pdf`.

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

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

[Swe02a] L. Sweeney. Achieving $k$-anonymity privacy protection using generalization and suppresion. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[Swe02b] L. Sweeney. $k$-Anonymity: A model for preserving privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[TCG03] TCG TPM specification version 1.2. https://www.trustedcomputinggroup.org, Nov 2003.

[Tim97] Time. *The Death of Privacy*, August 1997.

[Van] Data Vantage. http://www.datavantage.com/.

[Vaz04] Vijay Vazirani. *Approximation Algorithms.* Springer, 2004.

[Vit85] Jeff Vitter. Random sampling with a reservoir. *ACM Transaction on Mathematical Software*, pages 37–57, 1985.

[War65] S.L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Assoc.*, 60(309):63–69, March 1965.

[Win02] W. Winkler. Using simulated annealing for k-anonymity. *Research Report 2002-07, US Census Bureau Statistical Research Division*, November 2002.

[WJW] Lingyu Wang, Sushil Jajodia, and Duminda Wijesekera. Securing OLAP data cubes against privacy breaches. In *In Proc. of the 2004 IEEE Symposium on Security and Privacy.*

[WWJ04] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. Cardinality-based inference control in data cubes. In *Journal of Computer Security*, 2004.

[XM06] Ying Xu and Rajeev Motwani. Random sampling based algorithms for efficient semi-key discovery, 2006. Available from URL: `http://theory.stanford.edu/~xuying/papers/minkey_vldb.pdf`.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *Proceedings of the 1986 Annual IEEE Symposium on Foundations of Computer Science*, 1986.

[Yuh06] Noel Yuhanna. Protecting private data with data masking, March 2006. Available from URL: `http://www.forrester.com`.