PRIVACY PRESERVING INFORMATION SHARING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Alexandre Valentinovich Evfimievski

August 2004

PRIVACY PRESERVING INFORMATION SHARING

Alexandre Valentinovich Evfimievski, Ph.D.

Cornell University 2004

Modern business creates an increasing need for sharing, querying and mining information across autonomous enterprises while maintaining privacy of their own data records. The capability of preserving privacy in query processing algorithms can be demonstrated in two ways: through statistics and through cryptography. Statistical approach evaluates disclosure by its effect on an adversary's probability assumptions regarding privacy-sensitive data properties, while cryptographic approach gives comparative lower bounds on the computational complexity of learning these properties.

This dissertation presents results in both approaches. First, it considers the setup with one central server and a large number of clients connected only to the server, each client having a private data record. The server wants to generate an aggregate model of clients' data, and the clients want to limit disclosure of their individual records. Before sending to the server, each client hides its record using randomization, i.e. replaces the record with another one drawn from a certain distribution that depends on the original record. Disclosure is limited statistically by providing guarantees against "privacy breaches": situations when the randomized record significantly alters the server's probability for answer "yes" to some sensitive question about the original record. Privacy preserving mining of association rules is used as a concrete application for the method, with private records being small sets of items. More generally, a novel upper bound on privacy breaches is given, which at once covers all questions about an individual client's

record, and which works regardless of the client's data distribution. The bound is easy to use with many different types of randomization.

Second, the dissertation proposes a paradigm of minimal information sharing across several private databases, and instantiates it by developing cryptographic protocols for intersection, equijoin, intersection size, and equijoin size queries over two tables owned by two enterprises. Given a database query spanning multiple private databases, the paradigm suggests to compute the answer to the query while revealing minimal additional information apart from the query result. The protocols for intersection and equijoin are constructed using commutative encryption as well as Boolean circuits, and compared. The use of protocols is illustrated by applications.

BIOGRAPHICAL SKETCH

Alexandre Valentinovich Evfimievski entered the Department of Mathematics and Mechanics of Moscow State University, Russia, in 1992, and graduated with excellence in 1997. At Moscow State University, he studied computational complexity and theory of algorithms, the title of his diploma paper was "A Probabilistic Algorithm for Updating Files over a Communication Link." He entered the Department of Computer Science of Cornell University in 1998 and graduated in 2004. At Cornell, he worked on privacy preserving data mining and information sharing. In the summers of 2001 and 2002, he worked as a summer intern at IBM Almaden Research Center in San Jose, California. In his research on privacy at Cornell and at IBM he collaborated with Prof. Johannes Gehrke (his advisor at Cornell), Dr. Ramakrishnan Srikant (his mentor at IBM) and Dr. Rakesh Agrawal (his manager at IBM).

To my parents, Valentin Pavlovich and Zinaida Vasilyevna.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to Professor Johannes Gehrke, my scientific advisor at Cornell University, and acknowledge his contributions into this work as well as into my education. Johannes introduced me into the area of data mining at his course in advanced database systems, guided through the first steps of studying the unknown, taught how to successfully present our research to others, and was my role model as an active and industrious scientist. Our meetings and discussions were, in many ways, the driving force of my progress. It was due to his recommendation that I was admitted as a summer intern to IBM Almaden Research Center, where I commenced the work on privacy preserving information sharing.

During my two summers at IBM Almaden and at other periods when we worked together, Dr. Ramakrishnan Srikant was my mentor, teacher, and friend. He was ready to spend time working with me whenever I needed help, even if it meant for him going home late in the evening. At our daily meetings he went into all the technical details, pointing out my errors and suggesting solutions. He taught me how to write papers so that they get accepted to prestigious conferences, and how to prepare for the subsequent conference talks. He and my manager Dr. Rakesh Agrawal gave me research problems which could be successfully studied and published and which ultimately lead me to this dissertation. Their kindness and willingness to help, as well as their productivity, provide the direction for my personal development.

Any progress I made in solving mathematical problems is entirely due to the knowledge and training received in the Department of Mathematics and Mechanics at Moscow State University, under the supervision of my Moscow advisor Professor Nikolai Vereshchagin and Alexander Shen, and due to the courses taught there by many brightest mathematicians while they received miniscule salaries and suffered through the difficult transitional period of Russian economy. Their commitment to science and our education is beyond evaluation.

My special thanks and acknowledgement go to my committee members at Cornell, Prof. Anil Nerode and Prof. Jon Kleinberg, as well as to Prof. Jayavel Shanmugasundaram, Prof. Shai Ben-David, and all other faculty members who contributed to my progress. I also acknowledge the help of my friends Alin Dobra (who told me about sketches and kept my computer working), Cristian Bucila, Alexei Kopylov, Yannis Vetsikas, and all those who helped me in various ways.

Finally, I would like to thank my parents and relatives for their constant support, by word and by action.

My study and research at Cornell University would have been impossible without generous financial support from the University in the form of teaching and research assistantships.

The research in this dissertation was supported in part by NSF Grants IIS-0084762 and IIS-0121175, the Cornell Information Assurance Institute, and by gifts from Microsoft and Intel.

Parts of this dissertation are based on publications [69, 68, 3] © ACM 2002, 2003, and on publication [70] © Elsevier Ltd., 2003.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 The Problem of Preserving Privacy

The explosive progress in networking, storage, and processor technologies is resulting in an unprecedented amount of digitization of information. It is estimated that the amount of information in the world is doubling every 20 months [129]. In concert with this dramatic and escalating increase in digital data, concerns about privacy of personal information have emerged globally [60, 65, 129, 161]. Privacy issues are further exacerbated now that the internet makes it easy for the new data to be automatically collected and added to databases [27, 41, 42, 165, 166, 167]. Datasets containing sensitive records about thousands or millions of individuals and businesses become themselves a valuable asset that can be sold by one company to another, used for purposes different from what respondents originally expected (such as advertising), or exploited for unfair bias or discrimination of some respondents over the others (asking different prices, refusing a job etc.). The concerns over massive collection of data are naturally extending to analytic tools applied to data. Data mining, with its promise to efficiently discover valuable, non-obvious information from large databases, is particularly vulnerable to misuse [37, 63, 129, 160]. Information integration of sensitive datasets across enterprises also naturally leads to privacy issues. Although it has long been an area of active database research [31, 43, 62, 89, 168], the literature has so far tacitly assumed that the information in each database can be freely shared. However, there is now an increasing need for computing queries across databases belonging to autonomous entities in such a way that no more information than necessary is revealed from each database to the other databases. This need is driven by several trends:

- **End-to-end Integration**: E-business on demand requires end-to-end integration of information systems, from the supply chain to the customer-facing systems. This integration occurs across autonomous enterprises, so full disclosure of information in each database is undesirable.
- **Outsourcing**: Enterprises are outsourcing tasks that are not part of their core competency. They need to integrate their database systems for purposes such as inventory control.
- **Simultaneously compete and cooperate**: It is becoming common for enterprises to cooperate in certain areas and compete in others, which requires selective information sharing.
- **Security**: Government agencies need to share information for devising effective security measures, both within the same government and across governments. However, an agency cannot indiscriminately open up its database to all other agencies.
- **Privacy**: Privacy legislation and stated privacy policies place limits on information sharing. However, it is still desirable to mine across databases while respecting privacy limits.

One long-standing and well-studied practical problem of sharing private information is the collection, research and publication of demographic and socio-economical data by the government (see Section 2.1). Many countries' governments conduct periodic nationwide censuses as well as various polls over samples of respondents. The collected information is used for planning and forecasting economical developments, and is needed by many companies and researchers outside the government. However, legislature, such as the United States Privacy Act of 1974 [140, 141], places limits on disclosure of identifiable individual records, and the respondents' willingness to participate depends on their privacy being preserved. So, the collecting agencies search for methods to achieve the best possible compromise between privacy of individuals and precision of statistical query evaluation. Some of their current practices can be found in [36, 71, 173, 64, 172].

Agrawal *et al* [6] argue that the database community has an opportunity to play a central role in the integration into the digitized world of such an essential human freedom as the right to privacy. This can be achieved by re-architecting the future database systems to include responsibility for the privacy of data as a fundamental tenet. Such databases may place greater emphasis on consented sharing rather than on maximizing concurrency, and store, process and release any personal information only according to the purpose for which the information has been collected. This dissertation's results also serve as small steps towards this noble goal.

1.2 Main Research Directions

The science of privacy and disclosure has its roots within the studies of cryptography, statistics, and mathematical logic, which go back to the nineteenth century and earlier. The currently popular methods and paradigms, however, are often thought to have begun in 1948–49 with the works of Claude Shannon on information and secrecy in communication [149, 150]. In [149] Shannon proposed a definition of a secrecy system for encrypting and decrypting messages, some operations over such systems, and the notion of "perfect secrecy". He suggested to represent the knowledge of an adversary by probability distributions over possible private data values: prior distribution before the cryptogram is revealed, and posterior distribution after the adversary sees the cryptogram (but not the key). Perfect secrecy corresponds to the situation where the posterior distribution is identical to the prior (i.e. where the adversary's knowledge does not change) for any possible cryptogram. A way to relax this notion is shown in Section 4.3 of this dissertation. In [150] Shannon and Weaver introduced a measure of information communicated by a random variable, as well as mutual information between two dependent random variables. Mutual information was later used to measure disclosure [2, 53]; Section 4.5 of this dissertation gives a possible modification of its definition to reflect the "worst-case" nature of privacy.

The next big step was the rise of public-key cryptosystems in the end of the 1970s. Here, the assumption is that the adversary's computing capability is limited so that it cannot solve certain mathematical problems, such as factoring a product of two prime numbers or taking a logarithm modulo a prime, for sufficiently large arguments. Among seminal papers are Diffie and Hellman [47] which introduces a private key exchange protocol based on a variant of commutative encryption, and Rivest, Shamir and Adleman [137] which defines a well-known public key encryption algorithm. Another less known paper by the latter three authors [148] from that time gives a simple protocol for secure shuffling of poker cards between two players connected by information link only. It too uses commutative encryption, which is a reversible function $E_k(x)$ that encrypts x with key k so that for any two keys k_1 and k_2 we have $E_{k_1}(x) = E_{k_2}(x)$. Two players Alice and Bob choose their private keys a and b; Alice shuffles the cards, encrypts them using E_a and sends to Bob, who shuffles the cryptograms again and re-encrypts using E_b. Now Bob can pick his hand and send it to Alice for a "blind" E_a-decryption, then get them back, decrypt with E_b and play. The commutativity of E_a and E_b allows to reverse the order of decryption. This dissertation applies commutative encryption for two-party intersection and join query evaluation over private datasets in Chapter 5.

One particularly successful direction of the cryptographic approach is secure multiparty computation: the methodology for converting algorithms distributed across two or more parties into secure protocols that ensure honest behavior of participants or/and privacy of parties' data besides the legitimate results. A major idea in this direction was the definition of oblivious transfer [134, 67]. Its simplest form is the 1 out of 2 oblivious transfer: a two-party operation, where one party has two records r_0 and r_1 and learns nothing, while the other party has a bit b and learns only record r_b (and not r_{1-b}). Using oblivious transfer as a primitive operation, it is possible to convert any function whose arguments are distibuted across multiple parties into a protocol that securely evaluates this function while the parties learn nothing besides the function output [171, 86] (see [128] for a simple two-party conversion). The first step of this conversion is to express the function as a Boolean circuit, so the efficiency of the protocol depends on the size of the best known circuit for the function. More recently, other conversion procedures were proposed, see for example [125].

In secure multiparty computation, two main types of adversarial behavior are considered: semi-honest and malicious [83]. A semi-honest party executes its cryptographic algorithm correctly (and with the correct initial argument), but keeps a record of all information it receives from other parties and tries to learn properties of their private data; a malicious party may execute an algorithm different from the one prescribed by the protocol. Chapter 5 deals with semi-honest adversaries. If all adversaries are semi-honest, security of a multi-party protocol means that, for any party, all the information it sees during the protocol execution does not disclose anything new to the party, other than, of course, the legitimate result. This security can be demonstrated if each party can *simulate* everything it sees while running the protocol, given just the party's input and legitimate output and using its limited computational resources. The simulation does not have to be absolutely precise, but it should be good enough so that to distinguish it from the actual "view" of the protocol is as hard as to violate some well-known cryptographic assumption. The malicious model is substantially more difficult to work with, because, for example, there is no way to ensure that a party gives the correct argument to the protocol. Developing a mechanism of checking a party's input argument for legitimacy, e.g. by associating with it a proof of its origin, is an interesting challenge for future research in this area. Section 2.2 reviewes how secure multiparty computation is used in the framework of privacy preserving information sharing.

While computational complexity theorists were busy developing cryptography, the statistical direction in privacy progressed too, in the framework of *statistical databases*. A statistical database resides at server S within a certain company or organization and keeps private records about thousands of individual clients. Many other companies and organizations want to use this database for various forms of statistical analysis, such as computing averages and correlations over selected data subsets, and they are permitted to do it as long as they cannot identify who sent which record. A typical example is census data collected by the government; the businesses need to analyze regional census data to plan their development, but the government must ensure impossibility of matching a person (household) with his/her private record. A record can sometimes be successfully matched even if all uniquely identifiable attributes (name, address, social security number etc.) are withheld, by looking at outlier values, rare combinations of attributes, or by using background knowledge (the company's own database). One way to maintain privacy is to have all statistical queries sent to S, where these queries will be audited to see how much they reveal about individual entries; another way is to create a masked dataset and make it public. Both methods are not perfect: optimal auditing is a computationally hard problem [105], and it reveals all the companies' queries to the central server; masking the dataset lowers precision of statistical analysis and often introduces bias. Currently, masking is a more popular method, because it is easier to use in practice and it takes the query execution workload off the original server S. See Section 2.1 for more on statistical databases.

Mathematical logic makes its contribution to privacy as well. When a query to the restricted data is complex, or there is a sequence of many different queries, or if the snooping adversary has some background knowledge about the data, or when the privacy restrictions are formulated in the form of a policy written using a formal language — the question of preserving privacy may expand into the problem of finding and measuring dependence between logical systems. Even if we use cryptography or other means to ensure that nothing is disclosed besides what is asked by the query, it is still necessary to verify that the query itself is legitimate. Additional difficulties arise when some queries may *write* new information, and there is a dynamic environment with data being constantly updated and retrieved by different private entities. Such dynamic situations were studied in the literature on database security [108, 29, 114, 20]. In the simplest case, the private data values as well as queries issued to a database receive security levels, such as "top secret", "secret", "confidential" and "unclassified"; a party that is eligible for a lower security level cannot issue a query that accesses a higher-security value [156, 97, 96]. A related approach is to create a number of views for a private database, so that a view made accessible to a party is guaranteed to be safe against disclosure of all restricted data properties [122]. More generally, each data-carrying entity may have a set of privacy and security policies towards other entities, and before an exchange of information all participating parties verify compliance of the query to their policies. The use of the data collected under different policies may be restricted by associating a *purpose* with each record or attribute [6]; then these purposes also participate in evaluating query compliance. Special formal languages were invented to formulate such policies [7, 8]. For complex queries, the language would have to be quite general, which could make the policy verification problem algorithmically undecidable or infeasible. Nevertheless, logical inference engines based on search and heuristics may be used in practice [38, 92]. The amount of time spent on a query or on a protocol step, or the communication pattern across private entities, also need to be considered as a potential source of privacy leaks [143].

Finally, it is important to mention that, besides preventing privacy violations, one may track them down after they occurred. This can be achieved if the stolen data has some sort of a digital watermark of the owner, or even a unique digital fingerprint of the entity legally accessing the data [25, 5, 99, 39].

1.3 Summary and Contributions

The dissertation studies the concept of privacy preserving data mining that has been recently been proposed in response to privacy concerns (Section 1.1) [12, 110]. There have been two broad approaches. The data perturbation approach focuses on individual privacy, and reveals randomized (or otherwise masked) information about each record in exchange for not having to reveal the original records to anyone [2, 12, 69, 138]. In the secure multi-party computation approach, the goal is to evaluate queries and build data models across multiple databases without revealing the individual records in each database to the other databases [110, 101, 163]. Here some results are given in both of the approaches.

The main part of the dissertation consists of three chapters, each originally published as a paper: Chapter 3 as [70], Chapter 4 as [68], and Chapter 5 as [3]. The first two of these chapters concentrate upon the notion of statistical privacy, where knowledge is represented and measured in terms of probability distributions, while Chapter 5 works with computational privacy, where disclosure limitation is proven by reference to cryptographic computational intractability assumptions. Chapters 3 and 4 explore the use of randomization for preserving privacy of individual records while allowing to recover an approximate statistical model of the data with reasonably high precision. In both of them, there is one server and many thousands of respondents submitting their randomized private records to that server; then the server applies a mathematical procedure to recover significant statistical parameters of the original records from the collected randomized data. The server's knowledge about a respondent's record is thought as a probability distribution, prior (before the server learns the randomized record) and posterior (after the randomized record is received). Privacy of the records is evaluated using the notion of "privacy breaches": situations when there is a sensitive property of a private record whose prior probability (as seen by the server) is small, but whose posterior probability becomes large. Chapter 5 considers a secure multi-party computation setting (see overview in Section 2.2) with two servers owned by different, perhaps competing, enterprises, each having a private database. The servers need to evaluate a query jointly over both databases, disclosing to each other as little as possible besides the legitimate query answer. Cryptographic protocols are given for intersection and equijoin between two tables residing each at its server. Below is a more detailed description of contributions made in each of the chapters.

Chapter 3 presents a framework for mining association rules from transactions consisting of categorical items where the data has been randomized to preserve privacy of individual transactions. While it is feasible to recover association rules and preserve privacy using a straightforward "uniform" randomization inspired by the "randomized response" method by Warner [164], the discovered rules can unfortunately be exploited to find privacy breaches. Section 3.2 analyzes the nature of privacy breaches, and Section 3.3 proposes a class of randomization operators that are much more effective than uniform randomization in limiting the breaches. Section 3.4 derives formulae for an unbiased support estimator and its variance, which allow us to recover approximate values of itemset supports and association rule confidences from randomized datasets, and shows how to incorporate these formulae into mining algorithms. Finally, in Section 3.5 we present experimental results that validate the algorithm by applying it on real datasets.

Chapter 4 generalizes and extends the above framework in several directions. While Chapter 3 focuses mainly on the recovery of statistical information from randomized data, Chapter 4 concentrates on providing provable privacy guarantees through randomization. It refines the notion of privacy breaches given in the previous chapter so that it is convenient for use with any randomization, not just for itemsets, and classifies the breaches as "straight" and "inverse" (Section 4.2). A straight breach occurs if some rare property of a respondent's record (e.g., $\mathrm{HasAIDS} = \mathit{true}$) becomes likely when the server sees the randomized response; an inverse breach occurs if something uncertain (e.g., $\mathrm{Sex} = \mathrm{male}$) becomes virtually certain given the response. Section 4.3, the most important in the chapter, defines a condition that depends only on the randomization operator and not on the prior distribution, and which provides an upper bound on all privacy breaches possible under the given randomization. The condition, called "amplification condition," is then applied to the case of itemsets from the previous chapter. As a more complex example for amplification condition, in Section 4.4 we use pseudorandom generators to compress randomized itemsets by orders of magnitude without compromising privacy or support recovery. Finally, Section 4.5 discusses the use of information-like measures to quantify privacy, such as one recently proposed by D. Agrawal and C. Aggarwal [2], and defines "worst-case" information that bounds privacy breaches.

Chapter 5 is the cryptography-related part of the dissertation. It begins by introducing the concept of minimal information sharing across private entities: computing the answer to a query involving several databases so as to release as little as possible besides the answer while still being efficient (Section 5.1). Two motivating applications are given as an illustration. The protocol for secure two-party evaluation of set intersection is defined, together with the necessary cryptographic concepts, in Section 5.2; some additional background is provided also in Section 2.2. Section 5.3 defines the protocol for secure two-party equijoins, which is an extension of the intersection protocol. The protocols are supplemented with proofs of their security. Section 5.4 shows how to modify the intersection protocol for the evaluation of intersection size and join size. The rest of the chapter evaluates the protocols' computation and communication cost and compares it with protocols based on oblivious Boolean circuits.

Chapter 2

Data Mining and Privacy: Background and Overview

2.1 Statistical Databases

2.1.1 General Overview

There has been extensive research in the area of statistical databases motivated by the desire to provide statistical information about selections in a dataset without compromising sensitive individual records, see reviews in [153, 1, 75, 169, 54]. One important long-standing motivation is private data collection by statistical agencies of the governments in different countries (Section 1.1), e.g. census data collection. The typical setting is that there are:

- Many thousands, perhaps millions of data contributors (individuals, households, businesses), each sending a private record;
- One trusted centralized database of a statistical agency; and
- Many businesses and researchers who need to evaluate statistical queries over the database.

The aggregate queries issued to a statistical database often involve a selection criterion, so that aggregation is performed over a certain subset of records. A user may be interested in analyzing data coming from a given geographical region, from people of given sex, age, income or occupation, health records of patients having a given desease, test result or treatment, etc. Within the selected subset, the user either computes a simple aggregate operation such as sum, count, average, correlation, maximum, pth percentile, or applies data mining algorithms to learn a statistical model of the data, for example perform clustering, classification with decision trees, association rule mining, or estimate a distribution function. The users' queries may also constitute private information, since they show what these users are working on. When there are many users running complicated queries, the statistical database server may become a bottleneck if it has to handle all the queries alone.

The proposed techniques can be broadly classified into query restriction and data perturbation. In the query restriction approach, the trusted central server receives queries and evaluates them while constantly checking for possible privacy compromise. Disclosure limitation can be ensured either by systematically restricting or perturbing the query outputs, or by keeping an audit trail of all answered queries and analyzing how they overlap. In general case, finding the optimal tradeoff between query evaluation and query restriction means mapping the scope of logical inference from disclosed information, computationally a very difficult problem. It helps to consider special cases, such as when all queries are linear combinations of private values (still general enough to cover many statistical operations). Let us give an illustration with two recent papers. Dinur and Nissim [48] model a statistical database by a vector of Boolean attribute values $d_1, \ldots, d_n$, with a query being a subset $q \subseteq \{1 \ldots n\}$ to be answered by the sum of values d_i with indices in q, i.e. by $\sum_{i \in q} d_i$. The query outputs are perturbed by adding some random noise; preserving privacy here means preventing the recovery of coordinates d_i. The paper gives a lower bound on the perturbation needed to maintain any reasonable notion of privacy: it shows that if the added noise is asymptotically $o(\sqrt{n})$, a polynomial number of queries can be used to efficiently reconstruct almost the entire database very accurately, by means of a version of a linear programming algorithm. This result is remarkable because it suggests that there is little advantage in perturbing query outputs versus perturbing database values; indeed, if we take some $0 < p < 1/2$ and for each d_i replace it with $1 - d_i$ with probability p, the effect on query outputs will be of order $O(\sqrt{n})$. Situation is different, though, if only a sublinear number of queries is allowed: the paper shows that when the adversary's running time is $O(T(n))$, a very strong privacy protection can be reached by roughly $O(\sqrt{T(n)})$ query perturbation.

For the paper about auditing, consider one by Kleinberg, Papadimitriou and Raghavan [105]. The basic setup here is the same as above: a vector $d_1, \ldots, d_n$ of Boolean private values, and a number of queries $q_1, \ldots, q_m$ with $q_j \subseteq \{1 \ldots n\}$, whose outputs are sums of selected subsets q_j of coordinates. These sums are evaluated and disclosed precisely (with no perturbation), and we consider privacy to be violated if there is d_i whose Boolean value is uniquely defined by the query outputs. The paper proves that, in general, determining if there is a privacy violation is a coNP-complete problem. It stays coNP-complete even if all subsets q_j are required to be range queries over two (or more) numerical attributes; in this case each Boolean value d_i is assumed to be associated with numerical values for these extra attributes. In other words, query auditing problem is computationally intractable in this practically significant case. For range queries over just one numerical attribute, though, it is tractable; and non-optimal auditing (where we may suspect privacy violation even if there is none) is sometimes tractable too.

Thus, auditing and query restriction is an important direction in statistical databases, but naturally formulated problems in this area frequently turn out to be intractable or show no clear advantage so far over data perturbation. Also, with this approach the central server gets the bulk of the workload and learns all the queries made to the data, which severely restricts the practical utility. In consequence, most of the research in statistical database literature concentrated on ways to mask the dataset and then make it public.

2.1.2 Data Perturbation

The most studied approach to preserve privacy in a statistical database, here denoted by D, is to transform it into a new database D' so that businesses and researchers can use D' for (approximate) statistical query evaluation, but not for the recovery of sensitive information such as individual records. Usually D' is chosen to be similar in appearance to D, and in such a way that algorithms for statistical analysis of D also work on D' (and give approximate results). Then we can call D' a *perturbation* of D, and the approach can be called *data perturbation*. Sometimes, however, it is better to choose D' in a way that has no resemblance to D and/or to use very different statistical analysis algorithms to avoid bias and improve precision.

The data perturbation family includes such transformations of D as swapping values between records, replacing the original data by sampled values from the same distribution (imputation), adding or multiplying noise to the values in the database, rounding numerical attributes or coarsening categorical ones (i.e. replacing a value with a taxonomical category that includes this value), aggregating within small sets of records, leaving some values blank (cell suppression) etc. The specifics of the perturbation are determined by the trusted central server which already has the non-perturbed database D.

Typically, database attributes in D are classified into *identifying*, *sensitive*, and other attributes [169]. Identifying attributes constitute the information about private entities available from outside the statistical database (e.g. from phone-books, Internet and other public resources). They can be either *direct*, such as *Name*, *Address*, and *Personal ID*, or *indirect*, such as *Occupation*, *Sex*, *Age*, and *Region of Residence*. Direct identifiers should be excluded from D', but indirect identifiers may be revealed as long as there is sufficient perturbation to prevent positive identification from rare combinations of their values. Sensitive attributes are those whose values are private and should be protected against disclosure. By default, all attributes are either identifying or sensitive. Sometimes the precise value of an attribute (such as *Income*) is sensitive, but its coarse value is identifying, because it can be learned from outside the database.

The goal of data perturbation is to balance disclosure risk and information loss. Disclosure risk measures privacy, and the main emphasis in the literature is put on identity disclosure (*re-identification*), rather than on the disclosure of sensitive values in a record whose source's identity is known. Disclosure risk depends on the assumptions about the intruder's knowledge and behavior. Information loss measures the decrease in usefulness of the data for statistical query evaluation, such as the loss in precision and bias. There are many ways to quantify information loss, depending on the type of the data, the way it is perturbed, and the way it is going to be used.

There are two main types of statistical databases: *microdata* and *tabular data* [169]. Microdata is the basic type; it consists of a series of records, each containing information on an individual unit (a person, a company, etc.). Tabular data is obtained from microdata through aggregation of numerical values called *response variables* defined for each individual record (e.g., some sensitive attribute, or just "1" for frequency counts). Each cell in a table corresponds to each combination of values of categorical *spanning variables*, and gives the subtotal for the response variable aggregated over all records having specified values for the spanning variables. For example, if each record corresponds to a company, the response variable may be *Turnover*, while the spanning variables may be *Activity Type*, *Region Name*, and *Business Size* (small, medium, large); then we have a three-dimensional table. Tables may also have marginal totals.

When tabular data is made public, disclosure of private information may occur when some cells in a table correspond to only a single record (a *population unique* [76]) in the whole population covered by the table. It is also dangerous when there are just two or three or zero records aggregated in a cell, or when the cell value is dominated by one or several records (e.g., by the turnover of several largest companies). For example, if a cell value is dominated by two competing companies, then a researcher from one of the companies, knowing that the other company adds to the same cell, can subtract his/her own company's value and get a good upper bound on the competitor's value. Therefore, when choosing a set of spanning variables to form a table for release, the trusted agency either has to make sure there are no population uniques and no dominating records in any cell, or leave some of the cells blank (the compromising ones as well as some non-compromising ones). Sometimes it is also possible to create a table based on a sample of the population, so that a record that is a "sample unique" is unlikely to be a "population unique" [75].

A common approach to tabular disclosure limitation is cell suppression. For each cell in a table, we evaluate the sensitivity of its value, and if it is above a certain theshold, the value is withheld. The sensitivity depends on the relative contribution of records to the cell value; if one or several records dominate, the sensitivity is high. Once sensitive cells are suppressed, it is often necessary to suppress some extra non-sensitive cells (*secondary* suppression) to avoid the disclosure of positions for sensitive cells in the table, as well as the disclosure of upper and lower bounds on the suppressed values. These bounds can be computed from the table marginals, the other released tables, and the assumptions coming from the nature of the data (such as nonnegativity of cell values), by writing all these constraints as linear (in)equalities and solving a linear programming problem. The choice and number of suppressed cells should be balanced with the loss of information in the table, often expressed as a sum of weights of suppressed cells, convenient for the use of linear and integer programming. Besides suppression, rounding and perturbation is used for tabular data as well. Please see [169, 40, 57] for details of these methods.

The closest statistical database research gets to the subject of this dissertation is in the area of microdata perturbation. Here, the trusted server collects a database containing many individual private records, and then releases a database with the same structure but with perturbed records. Disclosure risk is understood as the risk of re-identification. Some of the most common perturbation methods are:

- Adding/multiplying noise to numerical values, randomly changing a fraction of categorical values [164, 81, 103, 124, 88, 77];
- Subsampling records and imputation (inserting records generated from a statistical model) [142, 136, 135];
- Suppression, rounding and coarsening (generalization) of values in records [146, 145, 95];
- Swapping values between records [103, 123, 78];
- Microaggregation (clustering records and then releasing the averages over each cluster) [44, 79, 52].

A formal unified framework, called *matrix masking*, has been proposed [59] for describing a subset of these methods. In [75] it is defined as follows. Suppose that X is an $n \times p$ matrix representing the microdata for n individuals or organizations on p variables. Then matrix masking of X, which corresponds to the transformed microdata file, is given by matrix M:

$$M = A\,X\,B + C \tag{2.1}$$

The matrix A transforms records, B transforms attributes (variables), and C blurs the entries of X, or more generally of AXB. Several perturbation methods are special cases of (2.1): subsampling of records (delete rows of X), imputation of simulated records (add rows to X), microaggregation (combine rows of X), adding random noise (as matrix C), excluding selected attributes (delete columns of X), releasing just the covariance matrix of X (choose $A = X^T$). Clearly, if M is released in place of X, some information about (A, B, C) also needs to be provided, to make statistical analysis possible. For example, if C represents random noise, then one needs the expectations $\mathbf{E}\,(C)$ and covariances $\mathbf{Cov}\,(C)$. An advantage of matrix representation is that it makes statistical analysis more intuitive by reducing it to matrix algebra. Often, the transformation matrices are selected (given X) so that some commonly-used statistical parameters (means, correlations) do not change or change in a simple way from original to perturbed data. Matrix masking framework is not directly applicable to categorical attributes, but can still be used with matrices containing transitional probabilities. This dissertation provides a somewhat nontrivial example in this direction, see Section 3.3.

A different example is given by Gouweleeuw *et al* [88] in Post Randomization method for categorical attributes. Here, X consists of n records, and each record has p categorical values $\xi_1, \xi_2, \ldots, \xi_p$ where ξ_i ranges from 1 to K_i. These values are jointly randomized (but independently in each record), and in general the perturbation is defined by joint transitional probability

$$P^{v_1 \ldots v_p}_{u_1 \ldots u_p} \;=\; \mathbf{P}\left[x_1 = v_1, \ldots, x_p = v_p \mid \xi_1 = u_1, \ldots, \xi_p = u_p\right] \tag{2.2}$$

Often it is convenient to partition the set of all p attributes into several smaller subsets, and randomize jointly within subsets but independently across them. The paper observes that $P^{v_1 \ldots v_p}_{u_1 \ldots u_p}$ makes a Markov probability matrix P of size $K \times K$, $K = K_1 \ldots K_p$, each entry being a transitional probability (2.2) for some pair of input and output categorical records. If we denote by T_ξ the K-dimensional vector of frequencies for all records in the original dataset and by T_X the frequency vector in the randomized dataset, then

$$\mathbf{E}\,(T_X) \;=\; P^T T_\xi \quad \text{and} \quad \mathbf{E}\,\big((P^T)^{-1} T_X\big) \;=\; T_\xi,$$

which makes $(P^T)^{-1}T_X$ an unbiased estimator of record frequencies in original data given randomized data. Covariances can likewise be computed using matrix algebra, and any independence in randomization allows to simplify the matrices. The safety of this perturbation approach is measured in terms of "expectation ratios": for each combination of values $\vec{v} = (v_1, \ldots, v_p)$ the expectation ratio is

$$ER\,(\vec{v}) \;=\; P_{\vec{v}}^{\vec{v}}\, T_\xi[\vec{v}] \Big/ \sum_{\vec{u} \neq \vec{v}} P_{\vec{u}}^{\vec{v}}\, T_\xi[\vec{u}]$$

Intuitively, the smaller $ER\,(\vec{v})$ is, the more likely a randomized record with values set to $\vec{v}$ is "false" (corresponds to a different original record). Thus safety is verified by ensuring that the expectation ratios are small for all sensitive combinations of values. To choose the transition probabilities in P, the authors rely heavily on the prior knowledge of T_ξ frequencies, making $P^T T_\xi$ close to (or even equal to) T_ξ. Details can be found in [88]. In a very similar categorical setting, this dissertation develops safety measures and an approach to randomization parameters selection which does not use prior frequencies, see Section 4.3.

Substantial progress has been made in the development of statistical analysis for masked microdata, as well as of measures for disclosure risk and harm [107, 112, 32, 154, 117, 50, 162]. Some of the results and measures resemble ones introduced later in privacy preserving data mining literature; the main difference is that in statistical databases much attention is given to re-identification of records. An important example is the definition of *pessimistic risk* of re-identification by Lambert [107], quite close to the notion of privacy breach from Definition 4 (Section 3.2), as follows. Suppose the original dataset D has N (identifiable) records, and the released dataset X is computed from D by sampling n records and perturbing their attributes. Given X, the adversary computes probability $p_{i,j}$ that released record x_i corresponds to original record d_j; probability $1 - \sum_{i=1}^{n} p_{i,j}$ is for the case when record d_j is not sampled to X at all. Then, pessimistic risk is

$$D(X) \;=\; \max_{i=1}^{n} \max_{j=1}^{N} p_{i,j}$$

Thus, if $D(X) < \theta$, then threshold θ is a limit on re-identification privacy breaches. It is assumed that a (rational) adversary tries to match a given original record d_j by choosing x_i with the largest $p_{i,j}$, and fails to match if $1 - \sum_{i=1}^{n} p_{i,j}$ is larger than all of them. False identification (between unrelated records) is also counted by this definition, and paper [107] argues that false identification may harm a respondent. To measure the disclosure of a sensitive categorical attribute, say $d_1[1]$, in some original record d_1, it is suggested to find the best match among perturbed records, say x_i, and then compute probabilities

$$\mathbf{P}\left[d_1[1] = y \;\mid\; x_i\right]$$

for possible values y of $d_1[1]$. These probabilities are estimated from the whole perturbed dataset X by representing them in the form of logistic regression

$$\log \frac{\mathbf{P}\left[d_1[1] = y \;\mid\; x_i\right]}{\mathbf{P}\left[d_1[1] \neq y \;\mid\; x_i\right]} \;=\; \beta_0 + \vec{\beta}^T x_i$$

and assuming that $x_i[1]$ depends only on $d_1[1]$ (through perturbation).

A classical paper by Little [112] presents a general framework for recovery of statistical information from masked data. It assumes that the original (non-perturbed) dataset D is generated by a parameterized probability distribution with density $f_D(D \mid \Theta)$, and that perturbation of D into masked dataset Z occurs according to another probability distribution with density $f_Z(Z \mid D)$. Out of Z, some values are selected for release through the use of the third distribution $f_M(M \mid D, Z)$; some values D_{obs} from $D = (D_{\text{obs}}, D_{\text{mis}})$ and from $Z = (Z_{\text{obs}}, Z_{\text{mis}})$ may also be known to the analyst. Then, the problem of statistical analysis reduces to obtaining the maximum likelihood value for parameter Θ given the observed dataset; in other words, to maximizing the following function:

$$L\left(\Theta \mid M, D_{\text{obs}}, Z_{\text{obs}}\right) = \int f_D(D \mid \Theta)\, f_Z(Z \mid D)\, f_M(M \mid D, Z)\, \mathrm{d}D_{\text{mis}}\, \mathrm{d}Z_{\text{mis}}$$

One possible way to achieve this is by the algorithm of Expectation-Maximization [45, 118] in a way similar to described by Agrawal and Aggarwal [2] for privacy preserving data mining (Section 2.3). The paper then goes on to apply this framework to many different data masking methods.

A completely different, and promising, approach to measuring privacy is suggested in [104]. This paper measures private information in terms of its monetary value, as a form of intellectual property. The cost of each piece of information must be determined in a "fair" way, so as to reflect the contribution of this piece in the overall profit. Two notions of fairness are analysed in [104], both coming from the theory of coalitional games: the *core* and the *Shapley value*. Let S be the set of potential participants in a business, and for every subset $S' \subseteq S$ we know the payoff $v(S')$ that occurs if only the participants in S' actually cooperate. Then the core consists of all possible ways to divide the total payoff $v(S)$ between all participants so that, for all $S' \subseteq S$, the share given to S' is at least $v(S')$. In other words, the core contains all "fair" ways of dividing the total payoff, in the sense that no group of participants has an incentive to secede. The Shapley value is a way to divide the payoff so that each participating agent is awarded an amount equal to the average contribution of this agent to the payoff of the group at the time of his or her arrival, where the average is taken over all arrival orders of the agents. In our case, participating clients disclose certain private information to the server and then benefit from the data model built by the server. The paper analyses the core and the Shapley value for several simplified scenarios involving private information.

2.1.3 Relevance to This Dissertation

The usage of randomization for preserving privacy has been studied extensively in the framework of statistical databases [58, 66, 77, 88, 124]. In their case, however, the server has a complete database with the information from its clients fully known, and it has to make a version of this database public, for others to work with. This dissertation in Chapters 3 and 4 considers a different situation: when the central server is not trusted, and the respondents who provide private data want to hide their individual information from this server. We run the randomization procedure on the client's side, and it must be decided upon before the data is collected. In our setting, the server is the one who needs to learn statistical characteristics. So, our server has to choose randomization parameters *before* it knows the dataset, while for a statistical database, choosing randomization *after* collection gives a wider opportunity to ensure privacy as well as to preserve certain aggregate characteristics (such as averages and covariance matrices), or change them in a predetermined way. Privacy notion is also different: statistical databases stress re-identification threat, while in our case all records are already identified upon arrival at the server, and privacy is about hiding the values they contain.

The most relevant work from the statistical database literature is the work by Warner [164], where he developed the "randomized response" method for survey results. The method deals with a single boolean attribute (e.g., drug addiction). The value of the attribute is retained with probability p and flipped with probability $1 - p$. Warner then derived equations for estimating the true value of queries such as COUNT (Age = 42 & Drug Addiction = Yes). The approach we present in Section 3.2.1 can be viewed as a generalization of Warner's idea.

2.2 Secure Multi-Party Computation

2.2.1 Cryptographic Background

Chapter 5 considers a problem of secure two-party computation of intersections and joins, and uses some notions described in cryptographic literature. There are many excellent surveys and textbooks [84, 85, 83, 87, 157, 116, 147, 119] on cryptography and secure multiparty computation, so here I will not try to repeat them but rather just briefly introduce several most relevant notions.

Secure two-party computation [83, 85, 109] Suppose we have two parties, P_1 and P_2, each having some Boolean data vector: P_1 has x_1 and P_2 has x_2. The parties want to evaluate a function

$$(y_1, y_2) \;=\; f\,(x_1, x_2)$$

so that party P_1 learns y_1 (in addition to x_1), and party P_2 learns y_2 (in addition to x_2); the parties do not want to disclose anything else to each other. Of course, the function f must be computable; in fact, we require it to be computable in time polynomial in the size $|x_1| + |x_2|$ of the arguments. Being "polynomial" can be formalized in two different ways, *uniform* and *non-uniform*. To be polynomial in $|x_1| + |x_2|$ in the uniform sense means that there is a program M for a computer with unbounded memory (e.g. a Turing machine) which computes f correctly for every pair of arguments (x_1, x_2), and the number of steps it makes is bounded by a certain polynomial $p(|x_1| + |x_2|)$. To be polynomial in the non-uniform sense means that there is a *sequence* of Boolean circuits

$C_1, C_2, \ldots$, one circuit for each size $|x_1| + |x_2|$, such that for all arguments

$$\forall n \;\; |x_1| + |x_2| = n \;\Rightarrow\; f(x_1, x_2) = C_n(x_1, x_2)$$

The circuits C_n are required to be of size polynomial of their index n. It is easy to see that uniformly polynomial algorithms are also non-uniformly polynomial; the opposite is false since an arbitrary sequence of circuits can be chosen in an algorithmically undecidable way. So, uniformly polynomial complexity is more restrictive than non-uniform one, and any assumption that something is *not* polynomially computable is stronger when non-uniform. Before giving cryptographic definitions and proofs, we have to specify whether we work with uniform or non-uniform case, and in Chapter 5 the non-uniform case is chosen because it is simpler to work with (domain sizes for equijoin attributes can be fixed).

The two parties, P_1 and P_2, are engaged in a *protocol* if there is a pair of (polynomial-time in $|x_1| + |x_2|$) algorithms, one for each party, and each algorithm can send information through a communication link to the other algorithm. For party $P_i,\ i = 1, 2$, its *view* of the protocol is all information P_i sees thoughout the execution of the protocol, such as P_i's input x_i, output y_i, all information P_i receives though the link from the opposite party, and all random coin tosses P_i makes (if its algorithm has such capability). The protocol correctly computes function f if, with probability exponentially close to one, at the end both parties learn the correct outputs y_1 and y_2. The protocol preserves privacy if neither party P_i can learn anything besides y_i about the other party's data x_{3-i} by applying a polynomial algorithm to P_i's view of the protocol. This notion of privacy corresponds to *semi-honest* adversaries (P_1 and P_2) which are assumed to execute all steps of their algorithm correctly and use their true inputs. In this dissertation we do not consider *malicious* behavior when a party does not follow the algorithm or/and makes up a false input with the purpose to learn extra information about the opponent. Very recently there was a paper by Freedman, Nissim, and Pinkas [80] that considers malicious adversaries in a similar problem.

How to verify that, say, P_1 does not learn anything besides y_1 about P_2's input x_2? Intuitively, we have to show that, given x_1 and y_1, P_1's view of the protocol does not noticeably depend on x_2 (does not contain information about x_2). One way to show it is by creating a special algorithm A_{sim} that simulates P_1's view of the protocol given only x_1 and y_1, and then demonstrating that for any x_2 at P_2, as long as

$$f(x_1, x_2) \;=\; (y_1, \text{smth.}),$$

the real view as seen by P_1 is indistinguishable from the view simulated by A_{sim}. If the algorithms are allowed to toss random coins, then P_1 sees only its own coins and not P_2's coins, and we are checking indistinguishability of the *distribution* for the real and for the simulated view. Now, to require that simulation and real view are statistically indistinguishable (i.e. have distributions whose distance is negligible) is very restrictive for practical purposes, and a weaker notion is used: computational indistinguishability. The distributions are allowed to be statistically different, but they should not be separable by any polynomial-time algorithm. A more formal non-uniform definition of computational indistinguishability is as follows. Suppose we are given a sequence of finite domains $\{\Omega_k\}$ of k-bit numbers: $\Omega_k \subseteq \{0,1\}^k$, and assume that it is easy to identify elements of the Ω_k's. For each domain Ω_k, we have a pair of probability distributions $(\mathcal{D}_k, \mathcal{D}'_k)$ over that domain. We say that these two distributions are *computationally indistinguishable* if, for any family of Boolean circuits $\{C_k(x)\}$ of polynomial size w.r.t. k that define predicates over domains Ω_k (output "yes" or "no"), and for any polynomial $p(k)$, we have

$$\exists K_p \; \forall k \geqslant K_p \quad \mathbf{P}\left[C_k(x) \mid x \sim \mathcal{D}_k\right] \;-\; \mathbf{P}\left[C_k(x) \mid x \sim \mathcal{D}'_k\right] \;<\; \frac{1}{p(k)}.$$

Here $x \sim \mathcal{D}$ means that x is distributed according to $\mathcal{D}$, and **P** $[C(x)]$ is the probability that $C(x)$ returns "yes". In other words, if an adversary is polynomial-time, it cannot discern whether an x was generated by $\mathcal{D}_k$ or by $\mathcal{D}'_k$, with any noticeable probability (for large enough k).

Random oracle assumption [22, 74, 158] In Chapter 5 of this dissertation, we use a hash function known to both parties to encode the intersection or join attribute values. The hashes of different values should "look random," i.e., there should be no order in them nor dependency among them that could be exploited by an adversary to learn private information. To make it more precise, we shall rely on the standard *random oracle model* as follows. Suppose we have two finite sets S and S', and we need a hash function $h : S \to S'$. We assume that, besides our two parties P_1 and P_2, there is another party called "random oracle" that computes h. Every time P_1 or P_2 needs to evaluate $h(s)$ for some $s \in S$, the party sends s to the random oracle; if the oracle had already seen this value s, it returns to the party the same $h(s)$ as it did before, otherwise it picks $h(s)$ uniformly at random in set S' (and writes down, to use if asked this s again). In practice, random oracles are not available, but there are "good-quality" hash functions that may be used as a replacement. The conditions where such replacement is justified is still a subject of research.

Diffie-Hellman assumptions [47, 24] The proof that two distributions (of the real view and of the simulated view of a protocol) are computationally indistinguishable is usually based on reduction from a known computationally "hard" problem. Unfortunately, the situation in computational intractability theory is far from settled: most practically useful computational problems are not known to be intractable, and sometimes not even known to be NP-hard (in fact, evidence seems to be gathering that some computational problems fundamental to cryptography are not intractable [151, 152]). However, several number-theory problems are hypothesized to be hard more than 20 years ago and still stand so today, so they developed enough trust to be practically usable. Among them are the two assumptions used in this dissertation: the Diffie-Hellman assumptions. Suppose we are given a cyclic commutative group $G = \langle g \rangle$ over multiplication; $g \in G$ is one of its generating elements. Intuitively, the Computational Diffie-Hellman (CDH) assumption for (G, g) implies that, for random $1 \leqslant a, b \leqslant |G|$, computing g^{ab} given g^a and g^b is a computationally intractable problem (in polynomial time); the Decisional Diffie-Hellman (DDH) assumption implies that the distribution of (g^a, g^b, g^{ab}) is computationally indistinguishable from (g^a, g^b, g^c) where $1 \leqslant c \leqslant |G|$ is also random. In order to be more formal, let us consider a *sequence* of groups: for each integer $k > 0$, let (G_k, g_k) be a group and its generating element, i.e. $G_k = \langle g_k \rangle$, and where the elements of G_k are efficiently encoded as $n(k)$-bit strings for some $n(k) < \mathrm{const} \cdot k$. We shall say that the (non-uniform) CDH assumption holds for this sequence if, for any sequence of Boolean circuits C_k of size bounded by a polynomial of k which output a G_k-element, and for any polynomial $p(k)$

$$\exists K_p \; \forall k \geqslant K_p \quad \mathbf{P}\left[\, C_k(g_k^a, g_k^b) = g_k^{ab} \mid a, b \in_r \{1, \ldots, |G_k|\} \,\right] \; < \; \frac{1}{p(k)}.$$

We shall say that the (non-uniform) DDH assumption holds for this sequence if, for any sequence of Boolean circuits C_k of size bounded by a polynomial of k which output "yes" or "no," and for any polynomial $p(k)$

$$\begin{aligned} \exists K_p \; \forall k \geqslant K_p \quad & \mathbf{P}\left[\, C_k(g_k^a, g_k^b, g_k^{ab}) \mid a, b \in_r \{1, \ldots, |G_k|\} \,\right] \; - \\ & - \; \mathbf{P}\left[\, C_k(g_k^a, g_k^b, g_k^c) \mid a, b, c \in_r \{1, \ldots, |G_k|\} \,\right] \; < \; \frac{1}{p(k)}. \end{aligned}$$

Sometimes in the literature the CDH and the DDH assumptions are formalized a little differently: instead of (G_k, g_k) being one group and one generator for each k, they have a probability distribution over groups and generators. In practice, however, it is convenient to select one group and one generator in the beginning and not change them (but the initial selection can be done probabilistically, if required).

The groups G commonly used as candidates for the CDH and the DDH assumptions are prime-order subgroups of the multiplicative group $\mathbf{Z}_p^*$ of some residue-class field modulo a prime number p (in this case the prime number $q = |G|$ divides $p - 1$). The order of G should be sufficiently large w.r.t. p, for example $q > p^{1/10}$ [24]. An important special case is when $p = 2q + 1$ for prime q; then G is the subgroup of quadratic residues in $\mathbf{Z}_p^*$. In a cyclic group of prime order, any element other than 1 is a generator.

Oblivious transfer [134, 67, 127] One of the earliest, and most fundamental, of secure two-party operations was the transfer operation. Party P_1 has two n-bit messages (Boolean strings) M_0 and M_1, and party P_2 has a bit $b \in \{0, 1\}$. As the result of the transfer, P_1 learns nothing and P_2 learns message M_b. Let us give an example of a protocol for oblivious transfer, by Naor and Pinkas [127]. Fix prime integers p and q such that q divides $p - 1$, and consider a cyclic order-q subgroup $G_q = \langle g \rangle$ of the multiplicative group $\mathbf{Z}_p^*$ of residues modulo p. By g we denote a generating element of G_q. Suppose that the computational Diffie-Hellman (CDH) assumption holds for g and G_q and let function $H : G_q \times \{0, 1\} \to \{0, 1\}^n$ be a random oracle. Intuitively, we can think of H as a well-mixing hash function that takes elements of G_q and outputs an n-bit random-looking string. The steps of the protocol are as follows:

1. P_1 picks at random $1 \leqslant r, s \leqslant q$, and sends (g^s, g^r) to P_2.

2. P_2 picks at random $1 \leqslant k \leqslant q$ and sends to P_1 either $\xi = g^k$ (if $b = 0$), or $\xi = g^s/g^k$ (if $b = 1$).

3. P_1 receives ξ and sends to P_2 encryptions for the messages:

$$M_0 \rightarrow H(\xi^r, 0) \oplus M_0, \quad M_1 \rightarrow H((g^s/\xi)^r, 1) \oplus M_1.$$

4. P_2 receives $H((g^k)^r, b) \oplus M_b$ for its bit b; it uses g^r and k to compute $H((g^r)^k, b)$ and deciphers message M_b.

An intuitive explanation why this protocol is secure for semi-honest adversaries is the following. Party P_1 receives from P_2 only ξ, which is just a random element of G_q (regardless of b); so P_1 learns nothing. Party P_2 receives (g^s, g^r) and two messages masked by hash values $H((g^k)^r, b)$ and $H((g^s/g^k)^r, \neg b)$. In order to say at least something about a random-oracle hash value $H(x)$, it is necessary to *know* x, because for any other x' hash value $H(x')$ is independently random. But if P_2 could compute both $(g^k)^r$ and $(g^s/g^k)^r$ from (g^s, g^r, k), then P_2 could also compute g^{sr} in violation of the CDH assumption. The formal proof can be found in [127].

This 1-out-of-2 transfer protocol can be easily and efficiently extended into a 1-out-of-N transfer protocol, with N messages at P_1 and a number $0 \leqslant b \leqslant N-1$ for the desired message at P_2. It can be done by replacing P_1's value $g^s = C$ by $N-1$ values $C_1, \ldots, C_{N-1}$ and sending them all to P_2 while keeping $C_1^r, \ldots, C_{N-1}^r$ secret. Moreover, the same r and the $\{C_i\}$ can be reused for many transfers without compromising privacy and further improving efficiency, as long as the hash function is updated after each transfer. Several, say l, parallel 1-out-of-2 transfers can also be combined into one 1-out-of-2^l transfer. These methods lead to the computational complexity bound for oblivious transfers used in Section 5.6.

Yao's algorithm [171, 128] Yao [171] was the first to give a general procedure for two-party secure computation of any function expressed as a Boolean circuit. Suppose there is a function $f(x)$ that takes a Boolean vector x and outputs a Boolean vector y, and let $C(x)$ be a Boolean circuit that computes f. We have two parties P_1 and P_2, party P_1 has the input x while party P_2 has the circuit C. Yao's algorithm gives a secure protocol that allows P_1 to learn the output $y = C(x)$ and nothing else about the circuit, while P_2 learns nothing at all. An excellent explanation of the protocol is given by Naor, Pinkas and Sumner in [128], Section 2.3. This dissertation uses the protocol in Section 5.6 to evaluate the cost of circuit-based computation of intersections and equijoins. The main idea of the protocol is to replace circuit C by a "garbled" circuit, where the wires transmit random hash values instead of bits computed normally. At each wire i, two randomly chosen n-bit hash values H_i^0 and H_i^1 represent 0 and 1; the last bits c_i^0 and c_i^1 of $H_i^0 = (W_i^0, c_i^0)$ and $H_i^1 = (W_i^1, c_i^1)$ must be chosen to be different. There is also a pseudorandom hash function $F_g : \{0,1\}^n \to \{0,1\}^n$, one for each circuit gate g, used for masking hash-value outputs in the gate. Each logical gate g with two input wires i and j and one output wire k has a table with four masked hash values, one for each combination of the two input bits. When the gate receives hash inputs $H_i^{b_i}$ and $H_j^{b_j}$, it looks at their last bits $c_i^{b_i}$ and $c_j^{b_j}$, uses them to determine one of the four table entries $M_g[c_i^{b_i}, c_j^{b_j}]$, then unmasks it to get the output hash value:

$$H_k^{b_k} \;=\; M_g[c_i^{b_i}, c_j^{b_j}] \,\oplus\, F_g(H_i^{b_i}) \,\oplus\, F_g(H_j^{b_j})$$

The protocol begins with P_2 preparing the garbled circuit and sending it to P_1, then engaging with P_1 in $|x|$ 1-out-of-2 oblivious transfers so that P_1 can learn hash values for the input wires that correspond to the bits of x. At the end, P_1 recovers the output bits from their hash values using a provided table. Please see the paper [128] for details. While C's logical gates themselves are hidden from P_1, the connectivity structure of C is nevertheless disclosed; but this is not an issue if P_2 tries to hide its own input to the circuit and not the circuit itself.

Other Work on Oblivious Set Intersection In a paper that was not known to us before we obtained results in Chapter 5 of this dissertation, Huberman, Franklin and Hogg [93] described an algorithm for secure two-party set intersection very similar to ours, in the context of the problem of finding people with common preferences, without revealing the preferences. However, they did not provide a proof of its security. In [126], the authors also consider the problem of finding the intersection of two lists while revealing only the intersection. They present two solutions: the first involves oblivious evaluations of n polynomials of degree n each, where n is the number of elements in the list; the second solution requires oblivious evaluation of n^2 linear polynomials. In the context of databases, n will be quite large. A new paper by Freedman, Nissim, and Pinkas [80], published after our work, considers malicious adversaries in a similar problem.

2.2.2 Other Relevant Directions

Quite naturally, the problem of privacy preserving information sharing has a lot in common with many branches of cryptography, and the task of reviewing them all goes far beyond the scope of this dissertation. Here I would like to point out several areas most closely related to the problems treated in Chapters 3, 4 and 5, and mention key references to the literature. These areas are as follows:

1. Secure multiparty evaluation of functions and algorithms;

2. Private information retrieval;

3. Secure voting protocols;

4. Secure data structures and structure-preserving encryptions;

5. Security of approximate computations in sublinear time.

As we saw above, there exist quite efficient algorithms for secure two-party circuit evaluation. Theoretically, this makes possible to implement any database operation over two private entities securely. However, a number of questions remain: How to efficiently convert common database queries (and other algorithms) into circuits? How to extend the two-party solution to the multi-party case? Can we incorporate other primitives, besides logical gates, into the circuits while still being able to convert the circuits into privacy preserving protocols? Goldreich, Micali and Wigderson [86] were the first to describe the multi-party version of Yao's protocol; however, their protocol has quadratic communication complexity in the number of parties. Since then, there was a number of papers, most notably by Naor and Nissim [125] where oblivious circuits are enhanced with "look-up tables," making it possible to simulate any RAM machine with polylogarithmic blowup.

An important class of database queries is information retrieval: selection of a record or of a small set of records by one party (the receiver, R) out of a large database located at the other party (the sender, S). The selection can be by index or by some property over the attributes. Private information retrieval (for selection by index) means that the receiver party R obtains the i-th record from set of n records held by the sender party S without revealing i to S [34, 35, 106, 28, 82, 155]. With the additional restriction that R should only learn the value of one record, the problem becomes that of symmetric private information retrieval [82]. On the surface, it looks similar to 1-out-of-n oblivious transfer; however, here n is extremely large and it is infeasible to send to R the whole encrypted database. The research focuses on achieving polylogarithmic communication complexity while maintaining privacy guarantees. Recently results started to appear that address more general formulations of information retrieval, such as keyword search [130, 30].

Secure voting is relevant to both privacy preserving association rules and cryptographic protocols for query evaluation. This area deals with massively multi-party protocols, where there are thousands of voters and a number of authorities that count the votes. In the simplest case, each voter submits either 0 or 1, and authorities help each other to compute and publish the sum of all values. At most a certain number k of authorities may be malicious and not follow their protocol correctly. The requirements include eligibility (each eligible voter can cast only one vote), privacy (no k or less authorities can compute any voter's vote), verifiability (a voter can check if his/her vote has been counted and whether the published sum is correct), fairness (no information leaks until everyone has voted), robustness (tolerance to failure of at most k authorities), and sometimes also incoercibility (a voter should be unable to convince anyone how he/she has voted). Common approaches to solving the secure voting problem include using anonymous communication channel and blind signatures, as well as sharing the secret among authorities by using homomorphic encryption or by mixing the votes [139]. It could be interesting to extend this research to the case with only one partially-trusted authority.

Sometimes it is possible to perform certain database operations over encrypted data, so that only features relevant to the permitted operations are seen, and nothing else [14]. This dissertation provides one example (Chapter 5): computing set intersection and join by comparing sets masked by commutative encryption. Such a possibility can be based on a special encryption technique that preserves a given algebraic property of the encrypted value, e.g. homomorphic and commutative encryption [23, 100]. Alternatively, the data may be organized into a secure data structure [121] that permits certain operations, such as binary tree traversal in Merkle trees [120, 159] where each tree node contains the hash value of its children nodes.

Many types of queries to a database can be evaluated approximately without using the entire database, but rather by using a small summary, or a *sketch* of the database. In a non-cryptographic setting, this allows to save a lot of space and running time for query processing, and even makes it possible to run the queries over data streams whose size is unlimited [16, 18, 17, 51]. One example is the computation of scalar products between two attribute vectors in a database (the computation of association rule supports is a special case of this). If there are two vectors residing at two different parties, the standard computation of their scalar product would require linear communication cost and linear space to store the vectors; but if approximate answers are fine, one can project both vectors to a smaller-dimensional random space and use the scalar product of projections for approximation. Once we add cryptography, however, things get more complex, because we have to prove that approximate answers do not disclose more than the precise ones would [72, 90]. Nevertheless, reducing the number of cryptographic operations by working with sketches is an important emerging direction in privacy preserving databases: the gain in speed is too significant. The known cryptographic secure scalar products are not as efficient [19, 55, 163].

The paper [163] has a closer relation to this dissertation. There the authors, Vaidya and Clifton, consider the problem of mining association rules over data that is vertically partitioned across two sources, i.e, for each transaction, some of the items are in one source, and the rest in the other source. They use multi-party computation techniques for scalar products to be able to compute the support of an itemset (when the two subsets that together form the itemset are in different sources), without either source revealing exactly which transactions support a subset of the itemset. Related, but not directly relevant to our current work, is the problem of inducing decision trees over horizontally partitioned training data originating from sources who do not trust each other. In [63], each source first builds a local decision tree over its true data, and then swaps values amongst records in a leaf node of the tree to generate randomized training data. Another approach, presented in [111], does not use randomization, but makes use of cryptographic oblivious functions during tree construction to preserve privacy of two data sources.

2.3 Privacy Preserving Data Mining

2.3.1 Aggregate Information Collection

Suppose that some company needs to construct an aggregate model of its customers' personal data. For example, a retail store wants to know the age and income of its customers who are more likely to buy DVD players or mountain ski equipment; a movie recommendation system would like to learn users' movie preferences in order to make advertisements more targeted; or an on-line business arranges its webpages according to an aggregate model of its website visitors. In all these cases, there is one central server (the company), and many clients (the customers), each having a piece of information. The server collects this information and builds its aggregate model using, for example, a classification algorithm or an algorithm for mining association rules. Often the resulting model no longer contains personally identifiable information, but contains only averages over large groups of clients.

The usual solution to the above problem consists in having all clients send their personal information to the server. However, as we discussed in Section 1.1, many people are becoming increasingly concerned about the privacy of their personal data. They would like to avoid giving out much more about themselves than is required to run their business with the company. If all the company needs is the aggregate model, a solution is preferred that reduces the disclosure of private data while still allowing the server to build the model. One possibility is as follows: before sending its piece of data, each client perturbs it so that some true information is taken away and some false information is introduced. This approach is called *randomization*. Other possibilities include:

- Decreasing precision of the transmitted data by rounding, suppressing certain values, replacing values with intervals, or replacing categorical values by more general categories up the taxonomical hierarchy. These methods are usually considered in a somewhat different framework of statistical databases, see Section 2.1
- Using cryptography and secure multiparty computation techniques, discussed in Section 2.2. Unfortunately, such algorithms are often slow and require at least several non-colluding parties that collect information.

The usage of randomization for preserving privacy has been studied extensively in the framework of statistical databases [58, 66, 77, 88, 124]. In that case, however, the database is randomized when it is already fully known to the central server. This is different from our problem, where the randomization procedure is run on the client's side, and must be decided upon before the data is collected.

2.3.2 Numerical Randomization

Agrawal and Srikant [12] consider the following problem. Suppose we have one central server and N clients C_i, $i = 1, 2, \ldots, N$. Let each client have a numerical attribute x_i. Assume that each x_i is an instance of random variable X_i, where all X_i are independent and identically distributed (i.i.d.). The cumulative distribution function (the same for every X_i) is denoted by F_X. The server wants to learn the function F_X, or its close approximation; this is the aggregate model which the server is allowed to know. However, we would like to limit what the server knows about the actual instances x_i.

The paper proposes the following solution. Each client randomizes its x_i by adding to it a random shift y_i. The shift values y_i are i.i.d. random variables whose distribution F_Y is chosen in advance and is known to the server. Client C_i sends randomized value $z_i = x_i + y_i$ to the server, and the server's task is to approximate function F_X given F_Y and values $z_1, z_2, \ldots, z_N$. How much z_i discloses about x_i is measured in [12] in terms of confidence intervals; $c\%$-confidence intervals $I(z)$ around randomized values z satisfy

$$\forall x \quad \mathbf{P}\left[\, x \in I(Z) \;\mid\; Z = x + Y \,\right] \;\geqslant\; c\%.$$

The minimum size of $I(z)$ measures vulnerability of the private value x to the "attack" by estimating x with confidence $c\%$ given the corresponding randomization $z = x + y$.

An approximation to F_X is computed from $z_1, z_2, \ldots, z_N$ by an iterative algorithm based on Bayes' rule. For densities $f_X = F'_X$ and $f_Y = F'_Y$, the step $j \to j+1$ of the iteration is

$$f_X^{j+1}(a) \;:=\; \frac{1}{N} \sum_{i=1}^{N} \frac{f_Y(z_i - a)\, f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(z_i - z)\, f_X^j(z)\, dz}$$

For efficiency, the density functions f_X^j are approximated by piecewise constant functions over a partition of the attribute domain. In paper [2] it has been shown that a similar formula can be derived using the framework of Expectation-Maximization (EM) algorithms [45, 118], which allows to consider a more general randomization setting. Suppose the server wants to approximate density f_X of nonrandomized (original) attribute distribution by some density from a parametric family $\{f_{X;\Theta}\}$. If the server knew

$x_1, x_2, \ldots, x_N$, it could find a maximum likelihood parameter value $\hat{\Theta}$ by computing

$$\hat{\Theta} \;=\; \operatorname*{argmax}_{\Theta} \, \log \prod_{i=1}^{N} f_{X;\Theta}(x_i)$$

However, the server knows only $z_1, z_2, \ldots, z_N$; so this formula is replaced by an iterative EM procedure:

$$\Theta^{k+1} \;=\; \operatorname*{argmax}_{\Theta} \, \mathbf{E} \left[\log \prod_{i=1}^{N} f_{X;\Theta}(X_i) \;\middle|\; \begin{array}{c} f_X = f_{X;\Theta^k}, \\ \forall i:\; X_i + Y_i = z_i \end{array} \right]$$

A similar approach has been described earlier for statistical databases [112]. Both papers [12, 2] consider two specific examples of additive randomization: with a uniform (on a segment) and a Gaussian density f_Y for the shift distribution.

In [12] it is shown how to use randomized numerical data in classification, namely in building a decision tree [26]. The main problem in decision tree construction is finding the right attribute value split point at each node. The quality of a split point depends on the frequency of records from each class in the subsets to the left and to the right of the split point. The tree building algorithm uses randomized attribute values to estimate these frequencies and choose good split points.

2.3.3 Itemset Randomization

Instead of numerical values, the clients $C_1, C_2, \ldots, C_N$ may have other kinds of private information. One practically important case is when there is a set $\mathcal{I}$ of items (e.g., products offered by some on-line store), each client C_i has a private subset of items, or a transaction, $t_i \subseteq \mathcal{I}$ (e.g., the customer's product preferences or purchases), and the server wants to find frequent association rules (see Section 2.4). Frequencies of rules and itemsets constitute a statistical model of the data that the server is allowed to know; however, the disclosure of individual transactions t_i should be minimized.

Again, as above, it is possible to randomize transactions before sending them from clients to the server, and let the server apply statistical recovery techniques to learn approximate itemset frequencies. A substantial fraction of this dissertation is devoted to studying such itemset randomizations, particularly Chapter 3 and some parts of Chapter 4, previously published as papers [69, 70, 68]. Independently from us and at about the same time, itemset randomization was considered by Rizvi and Haritsa [138]. In their paper, each client's transaction t_i is represented as a Boolean vector with items $a \in \mathcal{I}$ as coordinates and

$$\forall\, a \in \mathcal{I}: \;\; t_i[a] = 1 \;\Leftrightarrow\; a \in t_i.$$

These vectors t are randomized (say, into t') by independently considering each coordinate and flipping it with probability p (and leaving as it is with probability $1 - p$). The recovery of itemset frequencies from randomized data is somewhat similar to ours (Section 3.4), though not as general; variance estimation for the frequencies is only described for one-item sets. Privacy is measured as a function of the following expression, for some $\lambda \in (0, 1)$:

$$\mathcal{R}(p, a) \;=\; \lambda \mathcal{R}_1(p, a) \,+\, (1 - \lambda) \mathcal{R}_0(p, a)$$

where

$$\begin{aligned} \mathcal{R}_b(p, a) \;=\; & \mathbf{P}\,[t'[a] = 1 \mid t[a] = b] \cdot \mathbf{P}\,[t[a] = b \mid t'[a] = 1] \;+ \\ & +\; \mathbf{P}\,[t'[a] = 0 \mid t[a] = b] \cdot \mathbf{P}\,[t[a] = b \mid t'[a] = 0] \end{aligned}$$

This expression depends on p and on the support of item a; to compute one privacy measure, $\mathcal{R}(p, a)$ is averaged across all items in proportion to their supports, or the average support value is used. As noted in the paper, such measure does not take into account privacy breaches (Section 3.2).

There has been a number of extensions to bit-wise randomization techniques concerned with mining efficiency [13], different privacy levels for different items [13, 170], building decision tree classifiers from randomized data [56] etc. However, the question of privacy guarantees seems to have been avoided there as well as in [138]. Paper [13] evaluates "reinterrogation privacy" based on measuring item vs. association privacy breaches (Definition 5), but uses a rather arbitrary averaging metric and does not attempt to prove any guarantees.

2.3.4 Multivariate Numerical Randomization

Consider the setting from Section 2.3.2, but suppose that each client C_i has more than one numerical attribute. In other words, the private record x_i of C_i is actually a vector $\vec{x}_i$. We want a randomization operator $R(\vec{x})$ that would randomize private vectors well enough to limit disclosure of the coordinates of $\vec{x}$ while still permitting statistical analysis of the dataset. A naive approach would be to randomize each coordinate in each $\vec{x}_i$ independently, by adding to $\vec{x}_i$ an i.i.d. random vector $\vec{y}_i$ with independent coordinates, as described above. Unfortunately, in this case we may run into problems: the private vector may often be recovered, with high precision, from the randomized vector by exploiting the dependencies among the original (nonrandomized) coordinates. Denote by $\mathcal{X}$ the matrix with vectors $\vec{x}_i$ as rows, and by $\mathcal{Y}$ the matrix with $\vec{y}_i$ as rows; assume that the number of rows is large enough. Kargupta *et al* [102] demonstrate that we can decompose matrix $(\mathcal{X} + \mathcal{Y})^T(\mathcal{X} + \mathcal{Y})$ (which may be viewed as covariance matrix of "random" vector $\vec{x} + \vec{y}$) into a sum

$$(\mathcal{X} + \mathcal{Y})^T(\mathcal{X} + \mathcal{Y}) \;\approx\; Q_s \Lambda_s Q_s^T \;+\; Q_r \Lambda_r Q_r^T$$

with the left term corresponding to the original data and the right term corresponding to the randomization noise. Here Q_s and Q_r are orthogonal matrices, while Λ_s and Λ_r are diagonal matrices. This decomposition is made by taking the set of eigenvalues of the covariance matrix and setting aside those eigenvalues (and eigenvectors) that are likely induced by the added noise; these eigenvalues belong to a certain interval $[\lambda_{\min}, \lambda_{\max}]$ defined in [102]. Now, by projecting the randomized vectors along the eigenvectors in Q_s we can recover original vectors with surprisingly high precision, thus breaching their privacy. This method works if the private vector distribution has a covariance matrix with only a few dominant eigenvalues (true in many real-life situations).

Another way of breaching privacy in randomized vectors with highly dependent coordinates is through the use of independent component analysis, as described by Liu, Kargupta and Ryan [113].

2.3.5 Multi-Party Data Mining

Besides data perturbation, there is another direction in privacy preserving data mining: executing data mining operations over a database whose parts belong to different private entities. Lindell and Pinkas [111] pointed out that applying generic secure multi-party computation techniques (Section 2.2) is not practical for data mining, since the algorithms used for it are often complex, and the inputs are massive datasets. They called for more specialized solutions, and, as a particular instance, they considered the problem of decision tree learning with the ID3 algorithm [133], a well-known and influential algorithm for this task. A decision tree is used for classification of data records (i.e. for prediction of the value of an unknown categorical attribute); each non-leaf node in the tree contains a test expression involving a known attribute, with as many children-nodes as there are test outcomes, and each leaf contains a prediction value. Given a learning dataset where the class attribute is known, the tree is constructed recursively, from the top down, by choosing the best test expression for each node (or making it a leaf). The best test at a node is chosen so that it maximizes *information gain* about the class attribute achieved by making the test. Let the class attribute C have possible values $1, 2, \ldots, k$; if a population T of N database records contains N_i records with $C = i$, information about C in this population is measured as

$$H_C(T) \;=\; -\sum_{i=1}^{k} \frac{N_i}{N} \log \frac{N_i}{N}$$

Suppose we make a test A that has possible outcomes $a_1, \ldots, a_m$, and let $T(a_j)$ be the subpopulation of records that output a_j to the test. Then the information about C after the test is

$$H_C(T \,|\, A) \;=\; \sum_{j=1}^{m} \frac{|T(a_j)|}{|T|}\, H_C(T(a_j))$$

Information gain is the difference between these two measures:

$$\mathrm{Gain}\,(A) \;=\; H_C(T) \,-\, H_C(T \,|\, A) \tag{2.3}$$

In the privacy preserving setting, there are two parties, each having a learning dataset for the decision tree. In the process of building the tree, the parties have to jointly evaluate information gain (2.3) for various subpopulations of their records, subpopulations that include records at both parties' datasets, without disclosing anything to each party besides the final answer (the test that maximizes information gain). Lindell and Pinkas achieve this by representing all intermediate numbers x as sums of two shares: $x = r + (x - r)$, where r is random. Each party keeps one of the two shares, r or $x - r$, essentially just a random number. Numbers represented by random shares can be used for computations, but the computation results should also be divided into shares, for example

$$y_1 + y_2 \;=\; (x_1 + x_2)\, \log(x_1 + x_2) \quad \text{for} \quad y \;=\; x\, \log x$$

Once the shares of information gains for all considered attribute tests are computed, a small circuit and Yao's algorithm (Section 2.2) is used to select the maximizing test for the decision tree node.

Many data mining operations involving two tables can be expressed as scalar products of two vectors, one vector for each table. For example, suppose there is a supermarket that has two sections, and each customer buys products in both sections at once. Purchase transactions of the customers are stored by each section in its private database, each section stores information about its products. Now, suppose both sections want to compute association rules (see Section 2.4) jointly over their tables, without disclosing their databases to each other. In particular, they need to be able to determine the support of an itemset $A \cup B$, where, for each transaction t, one party knows if $A \subseteq t$, and the other party knows if $B \subseteq t$. Let $\delta(X, t)$ be one if $X \subseteq t$ and zero otherwise; then we have

$$\mathrm{supp}^T(A \cup B) \;=\; \sum_{t \in T} \delta(A, t)\, \delta(B, t),$$

which is a scalar product of two vectors, one with coordinates $\delta(A, t)$ and the other with coordinates $\delta(B, t)$. When we want to prevent disclosure of the vectors and only make available the scalar product value, we face the problem of oblivious scalar product computation. This problem is addressed by Atallah and Du [19, 55] and by Vaidya and Clifton [163]. The solutions involve masking a vector with random numbers, and then removing randomness from the result using the linearity properties of the prodict.

2.4 Association Rule Mining

The problem of mining association rules in a dataset whose records are sets of items (itemsets) has been introduced by Agrawal, Imielinski and Swami in [4]. Since then, association rule mining became very popular as a tool for studying dependencies within massive databases having categorical attributes. In particular, it is used to process market basket data that consists of the purchase transactions made by the clients of a certain store. Here I shall remind some basic definitions related to association rules.

Suppose we have a set $\mathcal{I}$ of n items: $\mathcal{I} = \{a_1, a_2, \ldots, a_n\}$. Our dataset is a vector of subsets of $\mathcal{I}$ that are often called "transactions" by analogy to market basket transactions:

$$T = (t_1, t_2, \ldots, t_N); \quad \forall i \;\; t_i \subseteq \mathcal{I}$$

Given some itemset $A \subseteq \mathcal{I}$, we can define its support in dataset T as follows:

Definition 1. Given an itemset $A \subseteq \mathcal{I}$, its *support* $\operatorname{supp}^T(A)$ is defined as

$$\operatorname{supp}^T(A) := \frac{\#\{t \in T \mid A \subseteq t\}}{N}. \tag{2.4}$$

In other words, support of A is the same as the frequency with which A occurs in the transactions. An itemset $A \subseteq \mathcal{I}$ is called *frequent* in T if $\operatorname{supp}^T(A) \geqslant s_{\text{min}}$, where s_{min} is a user-defined parameter.

It is assumed, in association rule mining, that the only interesting subsets are those that occur reasonably frequently within the dataset. Given two disjoint itemsets A and B, both subsets of $\mathcal{I}$, we can define an *association rule* $A \Rightarrow B$ and measure its confidence as follows:

Definition 2. Given an association rule $A \Rightarrow B$, its *confidence* $\operatorname{conf}^T(A \Rightarrow B)$ is defined as

$$\operatorname{conf}^T(A \Rightarrow B) := \frac{\operatorname{supp}^T(A \cup B)}{\operatorname{supp}^T(A)}. \tag{2.5}$$

This means, confidence of $A \Rightarrow B$ is the same as the frequency with which B occurs in the transactions that contain A.

Association rule $A \Rightarrow B$ *holds* for dataset T if itemset $A \cup B$ is frequent, and if $\mathrm{conf}^T(A \Rightarrow B)$ is at least as large as another threshold c_{min}. We can see that the key to mining association rules is in finding frequent itemsets and computing their supports. A very efficient algorithm for finding frequent itemsets, called *Apriori*, has been invented by Agrawal and Srikant [10]. It is based on the property of *lattice anti-monotonicity*, which says that any subset of a frequent set is also frequent. The algorithm in its simplest form starts with the computation of supports for all 1-item sets by making a pass through the entire dataset. Once it is done, the algorithm discards all 1-item sets that are not frequent. All the frequent itemsets are stored for output. Now it makes all 2-item sets whose all 1-item subsets were frequent, and makes another pass over the dataset to compute these 2-itemsets' supports. Again, only the frequent 2-itemsets are retained, and now we are looking for 3-itemsets whose all 2-subsets are frequent. The process goes on as long as there are still itemsets left. The description below summarizes the algorithm:

1. Let $k = 1$, let "candidate sets" be all single items. Repeat the following until no candidate sets are left:

   (a) Read the data file and compute the supports of all candidate sets;

   (b) Discard all candidate sets whose support is below s_{min};

   (c) Save the remaining candidate sets for output;

   (d) Form all possible $(k+1)$-itemsets such that all their k-subsets are among the remaining candidates. Let these itemsets be the new candidate sets.

   (e) Let $k = k + 1$.

2. Output all the saved itemsets.

Chapter 3

Association Rule Mining in Randomized Data

3.1 Introduction

The material in this Chapter has been previously published in [69, 70]. This work is done in collaboration with Ramakrishnan Srikant and Rakesh Agrawal (IBM Almaden Research Center), and with Johannes Gehrke (Cornell University).

Here and in Chapter 4 we continue the investigation of the use of randomization in developing privacy-preserving data mining techniques. As described in Section 2.3, there has already been some research in this direction, particularly for the case of numerical data [12]. We extend this line of inquiry along two dimensions:

- categorical data instead of numerical data, and
- association rule mining [4] instead of classification.

We will focus on the task of finding frequent itemsets in association rule mining, which we briefly reviewed in Section 2.4. Our setting is going to be as follows. Suppose we have a server and many clients. Each client has a set of items (e.g., books or web pages or TV programs). The clients want the server to gather statistical information about associations among items, perhaps in order to provide recommendations to the clients. However, the clients do not want the server to know with certainty who has got which items. When a client sends its set of items to the server, it modifies the set according to some specific randomization policy. The server then gathers statistical information from the modified sets of items (transactions) and recovers from it the actual associations.

The following are the important results contained in this chapter:

- In Section 3.2, we define the notion of "privacy breaches" and show that a straightforward uniform randomization leads to such breaches.
- We present a class of randomization operators in Section 3.3 that can be tuned for different tradeoffs between discoverability and privacy breaches.
- We derive formulae for the effect of randomization on support, and in Section 3.4 show how to recover the original support of an association from the randomized data.
- An estimator for the confidence of association rules is given as well, and its precision evaluated (see Section 3.4.3).
- We present experimental results on two real datasets in Section 3.5, as well as graphs showing the relationship between discoverability, privacy, and data characteristics.

This chapter gives most attention to the algorithms for randomization of transactions and for recovery of frequent associations from randomized data. The choice of randomization parameters that limits privacy breaches is based on a powerful heuristic (Section 3.5.1), but does not yet provide a provable guarantee. Please see Chapter 4 for a substantially more sound and general treatment of privacy breach limitation.

3.1.1 Randomization

Let us recall some basic notions related to our topic. *Randomization* is a technique for preserving privacy that replaces each given private record with a random record by means of a certain *randomization operator*. The random record is considered safe for release. Mathematically, we can write

$$R(x, \omega) \;=\; y,$$

where x is the private record, y is the randomized record, R is the randomization operator, and ω is the outcome of all random "coin tosses" within R. We can also say that $R(x)$ is a random variable whose distribution depends on x, and the randomized record y is a realization of $R(x)$. The same randomized record may be obtained from many different original records, so knowing the randomized record does not necessarily disclose private information. However, the probability of getting a given y is larger for some original records and smaller for others; therefore, y carries approximate or statistical information about x. For example, in the "randomized response" method by Warner [164] (Section 2.1.3) both x and y can take values of 0 or 1, and

$$\begin{aligned}\mathbf{P}[R(0)=1] \;&=\; \mathbf{P}[R(1)=0] \;=\; p\\ \mathbf{P}[R(0)=0] \;&=\; \mathbf{P}[R(1)=1] \;=\; 1-p\end{aligned}$$

It is important to note that, even if, say, $\mathbf{P}[R(1)=1]=90\%$, upon seeing $y=1$ we cannot conclude that likely $x=1$ as well; in a similar way, if 90% of patients with lung cancer have cough, we cannot conclude that a given patient with cough is likely to have lung cancer. Only if we are given some *prior* probabilities for values of the private record x, we can update them upon seeing $y=1$ and compute *posterior* probabilities using Bayes formula:

$$\mathbf{P}[x=1 \mid R(x)=1] \;=\; \frac{\mathbf{P}[x=1]\cdot\mathbf{P}[R(1)=1]}{\mathbf{P}[x=1]\cdot\mathbf{P}[R(1)=1] \,+\, \mathbf{P}[x=0]\cdot\mathbf{P}[R(0)=1]}$$

Thus, when randomization is used, usually there is an underlying assumption about a probability distribution over possible values of private data.

3.2 Privacy Breaches

Let us now turn to the problem of randomizing transactions (sets of items) so as to allow mining of association rules while preserving individual privacy.

3.2.1 An Example: Uniform Randomization

A straightforward approach for randomizing transactions would be to generalize Warner's "randomized response" method (Section 2.1.3). Before sending a transaction to the server, the client takes each item and with probability p replaces it by a new item not originally present in this transaction. Let us call this process *uniform* randomization.

Estimating true (nonrandomized) support of an itemset is nontrivial even for uniform randomization. Randomized support of, say, a 3-itemset depends not only on its true support, but also on the supports of its subsets. Indeed, it is much more likely that only one or two of the items are inserted by chance than all three. So, almost all "false" occurrences of the itemset are due to (and depend on) high subset supports. This requires estimating the supports of all subsets simultaneously. (The algorithm is similar to the algorithm presented later in Section 3.3 for select-a-size randomization, and the formulae from Statements 1, 3 and 4 apply here as well.) For large values of p, most of the items in most randomized transactions will be "false", so we seem to have obtained a reasonable privacy protection. Also, if there are enough clients and transactions, then frequent itemsets will still be "visible", though less frequent than originally. For instance, after uniform randomization with $p = 80\%$, an itemset of 3 items that originally occurred in 1% transactions will occur in about $1\% \cdot (0.2)^3 = 0.008\%$ transactions, which is about 80 transactions per each million. The opposite effect of "false" itemsets becoming more frequent is comparatively negligible if there are many possible items: for 10,000 items, the probability that, say, 10 randomly inserted items contain a given 3-itemset is less than $10^{-7}\%$.

Since most of the items get replaced, we may suppose that uniform randomization with $p = 80\%$ preserves privacy well. However, it is not so, at least not all the time. If we know that our 3-itemset escapes randomization in 80 per million transactions, and that it is unlikely to occur even once *because of* randomization, then every time we see it in a randomized transaction we know with near certainty of its presence in the nonrandomized transaction. With even more certainty we will know that at least one item from this itemset is "true": as we have mentioned, a chance insertion of only one or two of the items is much more likely than of all three. In this case we can say that a *privacy breach* has occurred.

To be somewhat more precise, let $R(t)$ be our randomization, and let $A = \{a, b, c\}$ be a 3-item set; assume that overall there are 10,000 items and 10 million transactions, all of size 10. Suppose 100,000 transactions (1%) contain A, and 500,000 more transactions (5%) contain all but one items of A. How many of these transactions t will contain A after they are randomized? The following is a rough average estimate ($t' = R(t)$):

- $A \subset t$ and $A \subset t'$: $100{,}000 \cdot 0.2^3 = 800$ transactions;
- $|A \cap t| = 2$ and $A \subset t'$: $500{,}000 \cdot 0.2^2 \cdot \frac{8 \cdot 0.8}{10{,}000} = 12.8$ transactions;
- $|A \cap t| \leqslant 1$ and $A \subset t'$: $< 10^7 \cdot 0.2 \cdot \left(\frac{9 \cdot 0.8}{10{,}000}\right)^2 \approx 1.04$ transactions.

So, there will be about 814 randomized transactions containing A, out of which about 800, or 98%, contained A before randomization as well. Now, suppose that the server receives from a client a randomized transaction $t' = R(t)$ that contains A. The server now knows that the actual, nonrandomized transaction t contains A with probability about 98%. On the other hand, the prior probability of $A \subset t$ is just 1%. The disclosure of $A \subset t'$ has caused a probability jump from 1% to 98%, bringing about a privacy breach.

Although privacy is preserved on average, personal information leaks through uniform randomization for some fraction of transactions, despite the high value of p.

Chapters 3 and 4 are devoted to defining a framework for studying privacy breaches and developing techniques for finding frequent itemsets while avoiding breaches. In Chapter 3 we concentrate mainly on algorithms for randomizing transactions and recovery of frequent associations, while in Chapter 4 we give a more general treatment to privacy breaches and their limitation.

3.2.2 Privacy Breaches in a Transaction Dataset

Definition 3. Let $(\Omega, \mathcal{F}, \mathbf{P})$ be a probability space of elementary events over some set Ω and σ-algebra $\mathcal{F}$. A *randomization operator* is a measurable function

$$R : \Omega \times \{\text{all possible } T\} \to \{\text{all possible } T\}$$

that randomly transforms a sequence of N transactions into a (usually) different sequence of N transactions. Given a sequence of N transactions T, we shall write $T' = R(T)$, where T is constant and $R(T)$ is a random variable.

Definition 4. Suppose that a nonrandomized sequence T is drawn from some known distribution, and $t_i \in T$ is the i-th transaction in T. A *general privacy breach of level* ρ with respect to a property $P(t_i)$ occurs if

$$\exists T' : \ \mathbf{P}\left[P(t_i) \mid R(T) = T'\right] \geqslant \rho.$$

We say that a property $Q(T')$ *causes a privacy breach* of level ρ with respect to $P(t_i)$ if

$$\mathbf{P}\left[P(t_i) \mid Q(R(T))\right] \geqslant \rho.$$

When we define privacy breaches, we think of the prior distribution of transactions as known, so that it makes sense to speak about a posterior probability of a property $P(t_i)$ versus prior. In practice, however, we do not know the prior distribution. In fact, there is no prior distribution; the transactions are not randomly generated. However, modeling transactions as being randomly generated from a prior distribution allows us to cleanly define privacy breaches.

Consider a situation when, for some transaction $t_i \in T$, an itemset $A \subseteq \mathcal{I}$ and an item $a \in A$, the property "$A \subseteq t'_i \in T'$" causes a privacy breach w. r. t. the property "$a \in t_i$." In other words, the presence of A in a randomized transaction makes it likely that item a is present in the corresponding nonrandomized transaction.

Definition 5. We say that itemset A *causes a privacy breach of level* ρ if for some item $a \in A$ and some $i \in 1 \ldots N$ we have $\mathbf{P}\left[a \in t_i \mid A \subseteq t'_i\right] \geqslant \rho$.

In this chapter, we will focus on controlling the class of privacy breaches given by Definition 5. Thus we ignore the effect of other information the server obtains from a randomized transaction, such as which items the randomized transaction does not contain, or the randomized transaction size; see Chapter 4 for a slightly different and more general approach. We also do not attempt to control breaches that occur because the server knows some other information about items and clients besides the transactions. For example, the server may know some geographical or demographic data about the clients. Finally, in Definition 5, we only considered positive breaches, i.e., we know with high probability that an item was present in the original transaction. In some scenarios, being confident that an item was *not* present in the original transaction may also be considered a privacy breach.

3.3 Randomization and Its Properties

> "Where does a wise man hide a leaf? In the forest. But what does he do if there is no forest?" ... "He grows a forest to hide it in." – G.K. Chesterton, "The Sign of the Broken Sword"

The intuition behind the proposed randomization operators is quite simple: in addition to replacing some of the items, we shall insert so many "false" items into a transaction that one is as likely to see a "false" itemset as a "true" one.

3.3.1 Randomization Operators

Definition 6. We call randomization R a *per-transaction* randomization if, for $T = (t_1, t_2, \ldots, t_N)$, we can represent $R(T)$ as

$$R(t_1, t_2, \ldots, t_N) = (R(1, t_1), R(2, t_2), \ldots, R(N, t_N)),$$

where $R(i, t)$ are independent random variables whose distributions depend only on t (and not on i). We shall write $t'_i = R(i, t_i) = R(t_i)$.

Definition 7. A randomization operator R is called *item-invariant* if, for every transaction sequence T and for every permutation $\pi : \mathcal{I} \to \mathcal{I}$ of items, the distribution of $\pi^{-1}R(\pi T)$ is the same as of $R(T)$. Here πT means the application of π to all items in all transactions of T at once.

Definition 8. A *select-a-size* randomization operator has the following parameters, for each possible input transaction size m:

- Default probability of an item (also called *randomization level*) $\rho_m \in (0, 1)$;
- Transaction subset size selection probabilities $p_m[0], p_m[1], \ldots, p_m[m]$, such that every $p_m[j] \geqslant 0$ and

$$p_m[0] + p_m[1] + \ldots + p_m[m] = 1.$$

Given a sequence of transactions $T = (t_1, t_2, \ldots, t_N)$, the operator takes each transaction t_i independently and proceeds as follows to obtain transaction t'_i ($m = |t_i|$).

1. The operator selects an integer j at random from the set $\{0, 1, \ldots, m\}$ so that $\mathbf{P}[j \text{ is selected}] = p_m[j]$.

2. It selects j items from t_i, uniformly at random (without replacement). These items, and no other items of t_i, are placed into t'_i.

3. It considers each item $a \notin t_i$ in turn and tosses a coin with probability ρ_m of "heads" and $1 - \rho_m$ of "tails". All those items for which the coin faces "heads" are added to t'_i.

Remark 1. Both uniform (Section 3.2.1) and select-a-size operators are per-transaction because they apply the same randomization algorithm to each transaction independently. They are also item-invariant since they do not use any item-specific information (if we rename or reorder the items, the outcome probabilities will not be affected).

Definition 9. A *cut-and-paste* randomization operator is a special case of a select-a-size operator (and which we shall actually test on datasets). For each possible input transaction size m, it has two parameters: $\rho_m \in (0, 1)$ (randomization level) and an integer $K_m > 0$ (the *cutoff*). The operator takes each input transaction t_i independently and proceeds as follows to obtain transaction t'_i (here $m = |t_i|$) :

1. It chooses an integer j uniformly at random between 0 and K_m; if $j > m$, it sets $j = m$.

2. The operator selects j items out of t_i uniformly at random (without replacement). These items are placed into t'_i.

3. Each other item (including the rest of t_i) is placed into t'_i with probability ρ_m, independently.

Remark 2. For any m, a cut-and-paste operator has only two parameters, ρ_m and K_m, to play with; moreover, K_m is an integer. Because it is easy to find optimal values for these parameters (Section 3.5.1), we chose to test this operator, leaving the problem of optimizing the m parameters of the "unabridged" select-a-size to Chapter 4. To see that cut-and-paste is a case of select-a-size, let us write down the formulae for the $p_m[j]$'s:

$$p_m[j] = \sum_{i=0}^{\min\{K,j\}} \binom{m-i}{j-i} \rho^{j-i}(1-\rho)^{m-j} \cdot \begin{cases} 1 - m/(K+1) & \text{if } i = m \text{ and } i < K \\ 1/(K+1) & \text{otherwise} \end{cases}$$

Now let us give one example of a randomization operator that is not a per-transaction randomization, because it uses the knowledge of several transactions per each randomized transaction.

Example 1. The *mixing* randomization operator has one integer parameter $K \geqslant 2$ and one real-valued parameter $p \in (0, 1)$. Given a sequence of transactions $T = (t_1, t_2, \ldots, t_N)$, the operator takes each transaction t_i independently and proceeds as follows to obtain transaction t'_i:

1. Other than t_i, pick $K-1$ more transactions (with replacement) from T and union the K transactions as sets of items. Let t''_i be this union.

2. Consider each item $a \in t''_i$ in turn and toss a coin with probability p of "heads" and $1-p$ of "tails".

3. All those items for which the coin faces "tails" are removed from the transaction. The remaining items constitute the randomized transaction.

For the purpose of privacy-preserving data mining, it is natural to focus mostly on per-transaction randomizations, since they are the easiest and safest to implement. Indeed, a per-transaction randomization does not require the users (who submit randomized transactions to the server) to communicate with each other in any way, nor to exchange random bits. On the contrary, implementing mixing randomization, for example, requires to organize an exchange of nonrandomized transactions between users, which opens an opportunity for cheating or eavesdropping.

3.3.2 Effect of Randomization on Support

Let T be a sequence of transactions of length N, and let A be some subset of items (that is, $A \subseteq \mathcal{I}$). Suppose we randomize T and get $T' = R(T)$. The support $s' = \mathrm{supp}^{T'}(A)$ of A for T' is a random variable that depends on the outcome of randomization. Here we are going to determine the distribution of s', under the assumption of having a per-transaction and item-invariant randomization.

Definition 10. The fraction of the transactions in T that have intersection with A of size l among all transactions in T is called *partial support* of A for intersection size l:

$$\mathrm{supp}_l^T(A) := \frac{\#\{t \in T \mid \#(A \cap t) = l\}}{N}. \tag{3.1}$$

It is easy to see that $\mathrm{supp}^T(A) = \mathrm{supp}_k^T(A)$ for $k = |A|$, and that

$$\sum_{l=0}^{k} \mathrm{supp}_l^T(A) = 1$$

since those transactions in T that do not intersect A at all are covered in $\mathrm{supp}_0^T(A)$.

Definition 11. Suppose that our randomization operator is both per-transaction and item-invariant. Consider a transaction t of size m and an itemset $A \subset \mathcal{I}$ of size k. After randomization, transaction t becomes t'. We define

$$p_k^m[l \to l'] \;=\; p[l \to l'] \;:=\; \mathbf{P}[\#(t' \cap A) = l' \mid \#(t \cap A) = l]. \tag{3.2}$$

Here both l and l' must be integers in $\{0, 1, \ldots, k\}$.

Remark 3. The value of $p_k^m\,[l \to l']$ is well-defined (does not depend on any other information about t and A, or other transactions in T and T' besides t and t'). Indeed, because we have a per-transaction randomization, the distribution of t' depends neither on other transactions in T besides t, nor on their randomized outcomes. If there were other t_1 and B with the same (m,k,l), but a different probability (3.2) for the same l', we could consider a permutation π of $\mathcal{I}$ such that $\pi t = t_1$ and $\pi A = B$; the application of π or of π^{-1} would preserve intersection sizes l and l'. By item-invariance we have

$$\mathbf{P}\,[\#(t' \cap A) = l'] = \mathbf{P}\,[\#(\pi^{-1}R(\pi t)\,\cap\,A) = l'],$$

but by the choice of π we also have

$$\begin{aligned}\mathbf{P}\,[\#(\pi^{-1}R(\pi t)\,\cap\,A) = l'] &= \mathbf{P}\,[\#(\pi^{-1}R(t_1)\,\cap\,\pi^{-1}B) = l'] \\ &= \mathbf{P}\,[\#(t_1' \cap B) = l'] \;\neq\; \mathbf{P}\,[\#(t' \cap A) = l'],\end{aligned}$$

a contradiction.

Statement 1. *Suppose that our randomization operator is both per-transaction and item-invariant. Suppose also that all the N transactions in T have the same size m. Then, for a given subset $A \subseteq \mathcal{I}$, $|A| = k$, the random vector*

$$N \cdot (s_0', s_1', \ldots, s_k'), \quad \textit{where } s_l' := \mathrm{supp}_l^{T'}(A) \tag{3.3}$$

is a sum of $k+1$ independent random vectors, each having a multinomial distribution. Its expected value is given by

$$\mathbf{E}\,(s_0', s_1', \ldots, s_k')^T = P \cdot (s_0, s_1, \ldots, s_k)^T \tag{3.4}$$

where P is the $(k{+}1)\times(k{+}1)$ matrix with elements $P_{l'\,l} = p\,[l \to l']$, and the covariance matrix is given by

$$\mathbf{Cov}\,(s_0', s_1', \ldots, s_k')^T = \frac{1}{N} \cdot \sum_{l=0}^{k} s_l\, D[l] \tag{3.5}$$

where each $D[l]$ is a $(k+1) \times (k+1)$ matrix with elements

$$D[l]_{i\,j} = p\,[l \to i] \cdot \delta_{i=j} - p\,[l \to i] \cdot p\,[l \to j]\,. \tag{3.6}$$

Here s_l denotes $\mathrm{supp}_l^T(A)$, and the T over vectors denotes the transpose operation; $\delta_{i=j}$ is one if $i = j$ and zero otherwise.

Proof. Each coordinate $N \cdot s'_{l'}$ of the vector in (3.3) is, by definition of partial supports, just the number of transactions in the randomized sequence T' that have intersections with A of size l'. Each randomized transaction t' contributes to one and only one coordinate $N \cdot s'_{l'}$, namely to the one with $l' = \#(t' \cap A)$. Since we are dealing with a per-transaction randomization, different randomized transactions contribute independently to one of the coordinates. Moreover, by item-invariance assumption, the probability that a given randomized transaction contributes to the coordinate number l' depends only on the size of the original transaction t (which equals m) and the size l of intersection $t \cap A$. This probability equals $p\,[l \to l']$.

So, for all transactions in T that have intersections with A of the same size l (and there are $N \cdot s_l$ such transactions) the probabilities of contributing to various coordinates $N \cdot s'_{l'}$ are the same. We can split all N transactions into $k+1$ groups according to their intersection size with A. Each group contributes to the vector in (3.3) as a multinomial distribution with probabilities

$$(p\,[l \to 0]\,, p\,[l \to 1]\,, \ldots, p\,[l \to k]),$$

independently from the other groups. Therefore the vector in (3.3) is a sum of $k+1$ independent multinomials. Now it is easy to compute both expectation and covariance.

For a multinomial distribution $(X_0, X_1, \ldots, X_k)$ with probabilities $(p_0, p_1, \ldots, p_k)$, where $X_0 + X_1 + \ldots + X_k = n$, we have $\mathbf{E}\; X_i = n \cdot p_i$ and

$$\mathbf{Cov}\,(X_i, X_j) = \mathbf{E}\,(X_i - p_i)(X_j - p_j) = n \cdot (p_i \delta_{i=j} - p_i p_j).$$

In our case, $X_i = l$'s part of $N \cdot s'_i$, $n = N \cdot s_l$, and $p_i = p\,[l \to i]$. For a sum of independent multinomial distributions, their expectations and covariances add together:

$$\begin{aligned}
\mathbf{E}\left(N \cdot s'_{l'}\right) &= \sum_{l=0}^{k} N \cdot s_l \cdot p\,[l \to l']\,, \\
\mathbf{Cov}\left(N \cdot s'_i, N \cdot s'_j\right) &= \sum_{l=0}^{k} N \cdot s_l \cdot \left(p\,[l \to i] \cdot \delta_{i=j} - p\,[l \to i] \cdot p\,[l \to j]\right)
\end{aligned}$$

Thus, after dividing by an appropriate power of N, the formulae in the statement are proven. □

Remark 4. In Statement 1 we have assumed that all transactions in T have the same size. If this is not so, we have to consider each transaction size separately and then use per-transaction independence.

Statement 2. *For a select-a-size randomization with randomization level ρ and size selection probabilities $\{p_m[j]\}$, we have:*

$$p_k^m\,[l \to l'] \;=\; \sum_{j=0}^{m} \; p_m[j] \cdot \sum_{q\,=\,\max\{0,\,j+l-m,\,l+l'-k\}}^{\min\{j,l,l'\}} \frac{\binom{l}{q}\binom{m-l}{j-q}}{\binom{m}{j}} \cdot \binom{k-l}{l'-q}\,\rho^{l'-q}(1-\rho)^{k-l-l'+q}. \tag{3.7}$$

Proof. We are given a transaction $t \in T$ and an itemset $A \subseteq \mathcal{I}$, such that $|t| = m$, $|A| = k$, and $\#(t \cap A) = l$. In the beginning of randomization, a number j is selected with distribution $\{p_m[j]\}$, and this is what the first summation takes care of. Now assume that we retain exactly j items of t, and discard $m - j$ items.

Suppose there are q items from $t \cap A$ among the retained items. How likely is this? Well, there are $\binom{m}{j}$ possible ways to choose j items from transaction t; and there are $\binom{l}{q}\binom{m-l}{j-q}$ possible ways to choose q items from $t \cap A$ and $j - q$ items from $t \setminus A$. Since all choices are equiprobable, we get $\binom{l}{q}\binom{m-l}{j-q}\big/\binom{m}{j}$ as the probability that exactly q A-items are retained.

To make t' contain exactly l' items from A, we have to get additional $l' - q$ items from $A \setminus t$. We know that $\#(A \setminus t) = k - l$, and that any such item has probability ρ to get into t'. The last terms in (3.7) immediately follow. Summation bounds restrict q to its actually possible (= nonzero probability) values. $\square$

3.4 Recovery of Frequent Associations

3.4.1 Support Recovery

Let us assume that all transactions in T have the same size m, and let us denote

$$\vec{s} := (s_0, s_1, \ldots, s_k)^T, \quad \vec{s}' := (s_0, s_1, \ldots, s_k)^T;$$

then, according to (3.4), we have

$$\mathbf{E}\ \vec{s}' = P \cdot \vec{s}. \tag{3.8}$$

Denote $Q = P^{-1}$ (assume that it exists) and multiply both sides of (3.8) by Q:

$$\vec{s} = Q \cdot \mathbf{E}\ \vec{s}' = \mathbf{E}\ Q \cdot \vec{s}'.$$

We have thus obtained an unbiased estimator for the original partial supports given randomized partial supports:

$$\vec{s}_{\text{est}} := Q \cdot \vec{s}' \tag{3.9}$$

Using (3.5), we can compute the covariance matrix of $\vec{s}_{\text{est}}$:

$$\mathbf{Cov}\ \vec{s}_{\text{est}} \;=\; \mathbf{Cov}\,(Q \cdot \vec{s}') \;=\; Q\,(\mathbf{Cov}\ \vec{s}')\,Q^T \;=\; \frac{1}{N} \cdot \sum_{l=0}^{k} s_l\, Q\, D[l]\, Q^T. \tag{3.10}$$

If we want to estimate this covariance matrix by looking only at randomized data, we may use $\vec{s}_{\text{est}}$ instead of $\vec{s}$ in (3.10):

$$(\mathbf{Cov}\ \vec{s}_{\text{est}})_{\text{est}} = \frac{1}{N} \cdot \sum_{l=0}^{k} (\vec{s}_{\text{est}})_l\, Q\, D[l]\, Q^T.$$

This estimator is also unbiased:

$$\mathbf{E}\left(\mathbf{Cov}\ \vec{s}_{\text{est}}\right)_{\text{est}} = \frac{1}{N} \cdot \sum_{l=0}^{k} (\mathbf{E}\ \vec{s}_{\text{est}})_l\, Q\, D[l]\, Q^T = \mathbf{Cov}\ \vec{s}_{\text{est}}.$$

In practice, we want only the k-th coordinate of $\vec{s}$, that is, the support $s = \text{supp}^T(A)$ of our itemset A in T. We denote by $\tilde{s}$ the k-th coordinate of $\vec{s}_{\text{est}}$, and use $\tilde{s}$ to estimate s. Let us compute simple formulae for $\tilde{s}$, its variance and the unbiased estimator of its variance. Denote

$$q\,[l \leftarrow l'] := Q_{l\,l'}.$$

Statement 3.

$$\tilde{s} = \sum_{l'=0}^{k} s'_{l'} \cdot q\,[k \leftarrow l']\,;$$

$$\mathbf{Var}\ \tilde{s} = \frac{1}{N} \sum_{l=0}^{k} s_l \left(\sum_{l'=0}^{k} p\,[l \rightarrow l']\ q\,[k \leftarrow l']^2 - \delta_{l=k} \right);$$

$$(\mathbf{Var}\ \tilde{s})_{\mathit{est}} = \frac{1}{N} \sum_{l'=0}^{k} s'_{l'} \left(q\,[k \leftarrow l']^2 - q\,[k \leftarrow l'] \right).$$

Proof. Let us denote

$$\vec{p_l} := (p\,[l \rightarrow 0]\,, p\,[l \rightarrow 1]\,, \ldots, p\,[l \rightarrow k])^T,$$

$$\vec{q_l} := (q\,[l \leftarrow 0]\,, q\,[l \leftarrow 1]\,, \ldots, q\,[l \leftarrow k])^T.$$

Since $PQ = QP = I$ (where I is the identity matrix), we have

$$\sum_{l=0}^{k} p\,[l \rightarrow i]\ q\,[l \leftarrow j] = \sum_{l'=0}^{k} p\,[i \rightarrow l']\ q\,[j \leftarrow l'] = \delta_{i=j}.$$

Notice also, from (3.6), that matrix $D[l]$ can be written as

$$D[l] = \text{diag}(\vec{p_l}) - \vec{p_l}\,\vec{p_l}^{\,T},$$

where $\mathrm{diag}(\vec{p}_l)$ denotes the diagonal matrix with $\vec{p}_l$-coordinates as its diagonal elements. Now it is easy to see that

$$\tilde{s} = \vec{q}_k^{\;T}\,\vec{s}\,' = \sum_{l'=0}^{k} q\,[k \leftarrow l'] \cdot s'_{l'};$$

$$\begin{aligned}
\textbf{Var}\ \tilde{s} &= \frac{1}{N}\sum_{l=0}^{k} s_l\,\vec{q}_k^{\;T}\,D[l]\,\vec{q}_k = \\
&= \frac{1}{N}\sum_{l=0}^{k} s_l\,\vec{q}_k^{\;T}\,(\mathrm{diag}(\vec{p}_l) - \vec{p}_l\,\vec{p}_l^{\;T})\,\vec{q}_k = \\
&= \frac{1}{N}\sum_{l=0}^{k} s_l\,(\vec{q}_k^{\;T}\,\mathrm{diag}(\vec{p}_l)\,\vec{q}_k - (\vec{p}_l^{\;T}\,\vec{q}_k)^2) = \\
&= \frac{1}{N}\sum_{l=0}^{k} s_l\Big(\sum_{l'=0}^{k} p\,[l \to l']\;q\,[k \leftarrow l']^2 - \delta_{l=k}\Big);
\end{aligned}$$

$$\begin{aligned}
(\textbf{Var}\ \tilde{s})_{\text{est}} \;&=\; \frac{1}{N}\sum_{l=0}^{k}(\vec{q}_l^{\;T}\,\vec{s}\,')\Big(\sum_{l'=0}^{k} p\,[l \to l']\;q\,[k \leftarrow l']^2 - \delta_{l=k}\Big) = \\
&=\; \frac{1}{N}\sum_{j=0}^{k} s'_j\,\Big(\sum_{l,l'=0}^{k} q\,[l \leftarrow j]\;p\,[l \to l']\;q\,[k \leftarrow l']^2 - \\
&\quad -\sum_{l=0}^{k}\delta_{l=k}\,q\,[l \leftarrow j]\Big) \;=\; \frac{1}{N}\sum_{j=0}^{k} s'_j\Big(\sum_{l'=0}^{k}\delta_{l'=j}\,q\,[k \leftarrow l']^2 - \\
&\quad -q\,[k \leftarrow j]\Big) \;=\; \frac{1}{N}\sum_{j=0}^{k} s'_j\big(q\,[k \leftarrow j]^2 - q\,[k \leftarrow j]\big).
\end{aligned}$$

$\square$

We conclude this section by giving a linear coordinate transformation in which the matrix P from Statement 1 becomes triangular. (We use this transformation for privacy breach analysis in Section 3.5.1.) The coordinates after the transformation have a combinatorial meaning, as given in the following definition.

Definition 12. Suppose we have a transaction sequence T and an itemset $A \subseteq \mathcal{I}$. Given an integer l between 0 and $k = |A|$, consider all subsets $C \subseteq A$ of size l. The sum of supports of all these subsets is called the *cumulative support* for A of order l and is denoted as follows:

$$\Sigma_l \;=\; \Sigma_l(A,T) \;:=\; \sum_{C\subseteq A,\;|C|=l} \mathrm{supp}^T(C)\,, \;\; \vec{\Sigma} \;:=\; (\Sigma_0, \Sigma_1, \ldots, \Sigma_k)^T \tag{3.11}$$

Statement 4. *The vector $\vec{\Sigma}$ of cumulative supports is a linear transformation of the vector $\vec{s}$ of partial supports, namely,*

$$\Sigma_l \;=\; \sum_{j=l}^{k} \binom{j}{l} s_j \quad \textit{and} \quad s_l \;=\; \sum_{j=l}^{k} (-1)^{j-l} \binom{j}{l} \Sigma_j; \tag{3.12}$$

in the $\vec{\Sigma}$ and $\vec{\Sigma}'$ space (instead of $\vec{s}$ and $\vec{s}'$) matrix P is lower triangular.

Proof. We prove the left formula in (3.12) first, and then show that the right one follows from the left one. Consider $N \cdot \Sigma_l$; it equals

$$\begin{aligned} N \cdot \Sigma_l \;&=\; N \cdot \sum_{C\subseteq A,\;|C|=l} \mathrm{supp}^T(C) = \sum_{C\subseteq A,\;|C|=l} \#\,\{t_i \in T \mid C \subseteq t_i\} = \\ &=\; \sum_{i=1}^{N} \#\,\{C \subseteq A \mid\; |C| \,=\, l, C \subseteq t_i\}. \end{aligned}$$

In other words, each transaction t_i should be counted as many times as many different l-sized subsets $C \subseteq A$ it contains. From simple combinatorics we know that if $j = \#(A \cap t_i)$ and $j \geqslant l$, then t_i contains $\binom{j}{l}$ different l-sized subsets of A. Therefore,

$$N{\cdot}\Sigma_l = \sum_{i=1}^{N} \binom{\#(A \cap t_i)}{l} = \sum_{j=l}^{k} \binom{j}{l} {\cdot} \#\{t_i \in T \mid \#(A{\cap}t_i) = j\} = \sum_{j=l}^{k} \binom{j}{l} N{\cdot}s_j,$$

and the left formula is proven. Now we can check the right formula just by replacing the Σ_j's according to the left formula. We have:

$$\begin{aligned}
\sum_{j=l}^{k} \quad (-1)^{j-l}\binom{j}{l}\Sigma_j &= \sum_{j=l}^{k}(-1)^{j-l}\binom{j}{l}\sum_{q=j}^{k}\binom{q}{j}s_q = \\
&= \sum_{l\leqslant j\leqslant q\leqslant k}(-1)^{j-l}\binom{j}{l}\binom{q}{j}s_q = \sum_{q=l}^{k} s_q \sum_{j=l}^{q}(-1)^{j-l}\binom{j}{l}\binom{q}{j} \\
&= \sum_{q=l}^{k} s_q \sum_{j'=0}^{q-l}(-1)^{j'}\frac{(j'+l)!}{l!\,j'!}\,\frac{q!}{(j'+l)!\,(q-j'-l)!} = \\
&= \sum_{q=l}^{k} s_q\cdot\frac{q!}{l!\,(q-l)!}\sum_{j'=0}^{q-l}(-1)^{j'}\frac{(q-l)!}{j'!\,(q-l-j')!} = \\
&= \sum_{q=l}^{k} s_q\binom{q}{l}\sum_{j'=0}^{q-l}(-1)^{j'}\binom{q-l}{j'} = s_l,
\end{aligned}$$

since the sum $\sum_{j'=0}^{q-l}(-1)^{j'}\binom{q-l}{j'}$ is zero whenever $q-l>0$.

To prove that matrix P becomes lower triangular after the transformation from $\vec{s}$ and $\vec{s}\,'$ to $\vec{\Sigma}$ and $\vec{\Sigma}\,'$, let us find how $\mathbf{E}\ \vec{\Sigma}\,'$ depends on $\vec{\Sigma}$ using the definition (3.11).

$$\begin{aligned}
\mathbf{E}\ \Sigma'_{l'} &= \sum_{C\subseteq A,\ |C|=l'} \mathbf{E}\ \mathrm{supp}^{T'}(C) = \\
&= \sum_{C\subseteq A,\ |C|=l'}\ \sum_{l=0}^{l'} p^m_{l'}\,[l\to l']\cdot\mathrm{supp}^T_l(C) \\
&= \sum_{C\subseteq A,\ |C|=l'}\ \sum_{l=0}^{l'} p^m_{l'}\,[l\to l']\sum_{j=l}^{l'}(-1)^{j-l}\binom{j}{l}\Sigma_j(C,T) = \\
&= \sum_{j=0}^{l'}\underbrace{\sum_{l=0}^{j}(-1)^{j-l}\binom{j}{l}p^m_{l'}\,[l\to l']}_{c_{l'\,j}}\sum_{C\subseteq A,\ |C|=l'}\Sigma_j(C,T) = \\
&= \sum_{j=0}^{l'} c_{l'\,j}\sum_{C\subseteq A,\ |C|=l'}\ \sum_{B\subseteq C,\ |B|=j}\mathrm{supp}^T(B) = \\
&= \sum_{j=0}^{l'} c_{l'\,j}\sum_{B\subseteq A,\ |B|=j}\#\{C\mid B\subseteq C\subseteq A, |C|=l'\}\cdot\mathrm{supp}^T(B) = \\
&= \sum_{j=0}^{l'} c_{l'\,j}\sum_{B\subseteq A,\ |B|=j}\binom{k-j}{l'-j}\mathrm{supp}^T(B) = \sum_{j=0}^{l'} c_{l'\,j}\binom{k-j}{l'-j}\cdot\Sigma_j.
\end{aligned}$$

Now it is clear that only the lower triangle of the matrix can have non-zeros. □

3.4.2 Discovering Associations

We show how to discover itemsets with high true support given a set of randomized transactions. Although we use the Apriori algorithm [9] to make the ideas concrete, the modifications directly apply to any algorithm that uses *Apriori* candidate generation, i.e., to most current association discovery algorithms.[1] The key *lattice property* of supports used by *Apriori* is that, for any two itemsets $A \subseteq B$, the true support of A is equal to or larger than the true support of B. A simplified version of *Apriori*, given a (nonrandomized) transactions file and a minimum support s_{min}, works as follows:

1. Let $k = 1$, let "candidate sets" be all single items. Repeat the following until no candidate sets are left:

    (a) Read the data file and compute the supports of all candidate sets;

    (b) Discard all candidate sets whose support is below s_{min};

    (c) Save the remaining candidate sets for output;

    (d) Form all possible $(k+1)$-itemsets such that all their k-subsets are among the remaining candidates. Let these itemsets be the new candidate sets.

    (e) Let $k = k + 1$.

2. Output all the saved itemsets.

It is (conceptually) straightforward to modify this algorithm so that now it reads the randomized dataset, computes partial supports of all candidate sets (for all nonran-

[1]The main class of algorithms where this would not apply are those that find only maximal frequent itemsets, e.g., [21]. However, randomization precludes finding very long itemsets, so this is a moot point.

domized transaction sizes) and recovers their predicted supports and sigmas using the formulae from Statement 3. However, for the predicted supports the lattice property is no longer true. It is quite likely that for an itemset that is slightly above minimum support and whose predicted support is also above minimum support, that one of its subsets will have predicted support below minimum support. So if we discard all candidates below minimum support for the purpose of candidate generation, we will miss many (perhaps even the majority) of the longer frequent itemsets. Hence, for candidate generation, we discard only those candidates whose predicted support is "significantly" smaller than s_{min}, where significance is measured by means of predicted sigmas. Here is the modified version of *Apriori*:

1. Let $k = 1$, let "candidate sets" be all single-item sets. Repeat the following until k is too large for support recovery (or until no candidate sets are left):

   (a) Read the randomized data file and compute the partial supports of all candidate sets, separately for each nonrandomized transaction size[2];

   (b) Recover the predicted supports and sigmas for the candidate sets;

   (c) Discard every candidate set whose support is below its *candidate limit*;

   (d) Save for output only those candidate sets whose predicted support is at least s_{min};

   (e) Form all possible $(k+1)$-itemsets such that all their k-subsets are among the remaining candidates. Let these itemsets be the new candidate sets.

   (f) Let $k = k + 1$.

2. Output all the saved itemsets.

[2] In our experiments, the nonrandomized transaction size is always known and included as a field into every randomized transaction

We tried $s_{\min}-\sigma$ and $s_{\min}-2\sigma$ as the candidate limit, and found that the former does a little better than the latter. It prunes more itemsets and therefore makes the algorithm work faster, and, when it discards a subset of an itemset with high predicted support, it usually turns out that the true support of this itemset is not as high.

3.4.3 Estimating Confidence of Association Rules

Now we would like to see what happens if the support estimators from a randomized dataset are used for the computation of confidence for association rules.

Definition 13. Consider two disjoint itemsets A and B; the *confidence* of the association rule "$A \Rightarrow B$" is defined as

$$\mathrm{conf}^T(A \Rightarrow B) \;:=\; \frac{\mathrm{supp}^T(A \cup B)}{\mathrm{supp}^T(A)}. \tag{3.13}$$

Perhaps the simplest way to estimate $\mathrm{conf}^T(A \Rightarrow B)$ is by replacing the actual supports in 3.13 with the predicted supports estimated from a randomized dataset as described in Section 3.4.1. Denote by X the unbiased estimator for the support of $A \cup B$, and by Y the unbiased estimator for the support of A; then the actual confidence $\mathrm{conf}^T(A \Rightarrow B)$ equals $\mathbf{E}\,X/\,\mathbf{E}\,Y$, and the confidence estimator $(\mathrm{conf}^T(A \Rightarrow B))_{\mathrm{est}}$ equals X/Y. Mathematical expectation of this estimator does not behave well, because Y may sometimes fall too close to zero (though it happens rarely if A is a frequent itemset). However, one can compute the probability $p = f(\delta)$ that the actual confidence is within a given distance $\delta > 0$ from the estimator. Below we shall give the formulae needed for computing $f(\delta)$.

Let us denote

$$\alpha \;:=\; \frac{\mathbf{E}\;X}{\mathbf{E}\;Y} - \delta; \quad \beta \;:=\; \frac{\mathbf{E}\;X}{\mathbf{E}\;Y} + \delta.$$

Then the probability $f(\delta)$ that our estimator X/Y gets within δ from $(\mathbf{E}\,X/\,\mathbf{E}\,Y)$ is

$$\begin{aligned} f(\delta) \quad &:= \quad \mathbf{P}\left[\frac{X}{Y} - \delta \;\leqslant\; \frac{\mathbf{E}\;X}{\mathbf{E}\;Y} \;\leqslant\; \frac{X}{Y} + \delta\right] \;= \\ &= \quad \mathbf{P}\left[\alpha \;\leqslant\; X/Y \;\leqslant\; \beta\right] \;= \\ &= \quad \mathbf{P}\left[\alpha Y \;\leqslant\; X \;\leqslant\; \beta Y,\; Y > 0\right] \;+ \mathbf{P}\left[\beta Y \;\leqslant\; X \;\leqslant\; \alpha Y,\; Y < 0\right] \end{aligned}$$

For simplicity, we assume that the distribution of (X, Y) is a 2-dimensional Gaussian. The assumption seems justified because both X and Y are linear combinations of multinomials (see Statement 1). In this case,

$$f(\delta) = \int\limits_{-\infty}^{+\infty} \left| \int\limits_{\alpha y}^{\beta y} \frac{1}{2\pi\,|\Sigma|^{1/2}}\, e^{-\frac{1}{2}\binom{x-\mathbf{E}\,X}{y-\mathbf{E}\,Y}^{T}\Sigma^{-1}\binom{x-\mathbf{E}\,X}{y-\mathbf{E}\,Y}}\,dx \right| dy,$$

where Σ is the 2×2 matrix

$$\Sigma \;:=\; \begin{pmatrix} \mathbf{Var}\;X & \mathbf{Cov}\,(X,Y) \\ \mathbf{Cov}\,(X,Y) & \mathbf{Var}\;Y \end{pmatrix}.$$

The formulae for **Var** X and **Var** Y are given in Statement 3. The computation of **Cov** (X, Y) is a little more involved; intuitively, it requires to do Sections 3.3.2 and 3.4.1 for two itemsets A and B simultaneously. Let us start with a notion of "two-set partial supports."

Definition 14. The *two-set partial supports* for disjoint itemsets A and B are defined as follows:

$$s_{l_A,l_B} \quad = \quad \mathrm{supp}^{T}_{l_A,l_B}(A;B) \;:=\; \frac{\#\left\{t \in T \;:\; \begin{array}{c} |t \cap A| = l_A, \\ |t \cap B| = l_B \end{array}\right\}}{N}.$$

Here $l_A = 0, 1, \ldots, |A|$ and $l_B = 0, 1, \ldots, |B|$.

The vector of two-set partial supports is denoted by $\vec{s}_{II}$ and has $(|A| + 1)(|B| + 1)$ coordinates that sum up to 1. The vector of two-set partial supports for the randomized dataset is denoted by $\vec{s}\,'_{II}$. Analogously to (3.2), there is a two-set version of $p\,[l \to l']$:

$$p^m_{|A|,\,|B|}\begin{bmatrix} l_A \to l'_A \\ l_B \to l'_B \end{bmatrix} := \mathbf{P}\begin{bmatrix} |t' \cap A| = l'_A & \vdots & |t \cap A| = l_A \\ |t' \cap B| = l'_B & & |t \cap B| = l_B \end{bmatrix}$$

For a per-transaction, item-invariant randomization operator, this probability depends only on m, $|A|$, $|B|$, l_A, l_B, l'_A, and l'_B by the same argument as for $p\,[l \to l']$ (see Remark 3). If our randomization operator is a select-a-size with parameters ρ and $p_m[i]$ (see Definition 8), then

$$\begin{aligned} p^m_{|A|,\,|B|}\begin{bmatrix} l_A \to l'_A \\ l_B \to l'_B \end{bmatrix} = \sum_{i=0}^{m-l_A-l_B} \sum_{\substack{j_A = \max\{0,\, l_A + l'_A - |A|\} \\ j_B = \max\{0,\, l_B + l'_B - |B|\}}}^{\substack{j_A \leqslant \min\{l_A,\, l'_A\} \\ j_B \leqslant \min\{l_B,\, l'_B\}}} & p_m[i + j_A + j_B] \\ \cdot \binom{m}{i + j_A + j_B}^{-1} \binom{l_A}{j_A}\binom{l_B}{j_B}\binom{m - l_A - l_B}{i}\binom{|A| - l_A}{l'_A - j_A}\binom{|B| - l_B}{l'_B - j_B} & \cdot \\ \cdot\ \rho^{l'_A + l'_B - j_A - j_B}(1-\rho)^{|A|+|B|-l_A-l_B-l'_A-l'_B+j_A+j_B} & \end{aligned}$$

We denote the $(|A|+1)(|B|+1) \times (|A|+1)(|B|+1)$ matrix of these two-set probabilities by P_{II}, and its inverse P_{II}^{-1} by Q_{II}. Then, analogously to Statement 1, we have

$$\mathbf{E}\ \vec{s}\,'_{II} \ = \ P_{II}\vec{s}_{II}; \quad \mathbf{Cov}\ \vec{s}\,'_{II} \ = \ \frac{1}{N} \cdot \sum_{\substack{0 \leqslant l_A \leqslant |A| \\ 0 \leqslant l_B \leqslant |B|}} s_{l_A, l_B}\, D_{II}[l_A, l_B],$$

where

$$D_{II}[l_A, l_B]^{i_A, i_B}_{j_A, j_B} \quad = \quad p\begin{bmatrix} l_A \to i_A \\ l_B \to i_B \end{bmatrix} \cdot \delta_{\substack{i_A = j_A \\ i_B = j_B}} \ - \ p\begin{bmatrix} l_A \to i_A \\ l_B \to i_B \end{bmatrix} \cdot p\begin{bmatrix} l_A \to j_A \\ l_B \to j_B \end{bmatrix}.$$

Now we define the two-set partial support estimator and compute its covariance matrix:

$$\begin{aligned} (\vec{s}_{II})_{\text{est}} \ &= \ Q_{II}\vec{s}\,'_{II} \\ \mathbf{Cov}\,(\vec{s}_{II})_{\text{est}} \ &= \ \frac{1}{N} \cdot \sum_{\substack{0 \leqslant l_A \leqslant |A| \\ 0 \leqslant l_B \leqslant |B|}} s_{l_A, l_B}\, Q_{II}\, D_{II}[l_A, l_B]\, Q_{II}^T. \end{aligned} \tag{3.14}$$

The "one-set" partial support estimators for sets $A \cup B$ and A can be obtained from $(\vec{s}_{II})_{\text{est}}$ by linear combination of its coordinates. In particular, X and Y can be computed as follows:

$$X \;=\; (\vec{s}_{II})_{\text{est}}\,[\,|A|, |B|\,], \quad Y \;=\; \sum_{l_B=0}^{|B|} (\vec{s}_{II})_{\text{est}}\,[\,|A|, l_B\,]. \tag{3.15}$$

Remark 5. These estimators are exactly the same as those computed in Section 3.4.1. Indeed, consider for example set A; let an $(|A|+1) \times (|A|+1)(|B|+1)$-matrix M be such that $M\,\vec{s}_{II}$ equals the "one-set" vector $\vec{s}$ of partial supports for A. Since we have

$$\mathbf{E}\;\vec{s}\,' = P\,\vec{s}, \quad \mathbf{E}\;\vec{s}\,'_{II} \;=\; P_{II}\,\vec{s}_{II}, \quad \vec{s} = M\,\vec{s}_{II}, \quad \vec{s}\,' = M\,\vec{s}\,'_{II}$$

for all possible partial support vectors, hence we also have

$$P\,M = M\,P_{II} \quad \Longrightarrow \quad M\,Q_{II} = Q\,M,$$

i. e. both estimators do the same thing.

By combining equations (3.14) and (3.15), the formula for **Cov** (X, Y) is given by

$$\mathbf{Cov}\,(X, Y) \;=\; \frac{1}{N} \cdot \sum_{\substack{0 \leqslant l_A \leqslant |A| \\ 0 \leqslant l_B \leqslant |B|}} s_{l_A, l_B}\, \vec{q}_X^{\,T}\, D_{II}[l_A, l_B]\, \vec{q}_Y,$$

where (the $q\,[\ldots]$'s are the coordinates of Q_{II})

$$\vec{q}_X \;:=\; \left(q \begin{bmatrix} |A| \leftarrow l'_A \\ |B| \leftarrow l'_B \end{bmatrix} : \begin{array}{l} l'_A = 0, 1, \ldots, |A| \\ l'_B = 0, 1, \ldots, |B| \end{array} \right)^T$$

$$\vec{q}_Y \;:=\; \sum_{l_B=0}^{|B|} \left(q \begin{bmatrix} |A| \leftarrow l'_A \\ l_B \leftarrow l'_B \end{bmatrix} : \begin{array}{l} l'_A = 0, 1, \ldots, |A| \\ l'_B = 0, 1, \ldots, |B| \end{array} \right)^T$$

3.5 Experimental Results

Here we study the tradeoff between privacy and discoverability of association rules, by looking both at simulations and at real-life data experiments. Before we come to the experiments with datasets, we first show in Section 3.5.2 how our ability to recover supports depends on the permitted breach level, as well as on other data characteristics. The same is shown for the confidence of association rules in Section 3.5.3. We then describe the real-life datasets in Section 3.5.4, and present results for support estimation on these datasets in Section 3.5.5.

3.5.1 Privacy Evaluation

Let us determine how privacy depends on randomization. We shall use Definition 5 and assume a per-transaction and item-invariant randomization.

Consider some itemset $A \subseteq \mathcal{I}$ and some item $a \in A$; fix a transaction size m. We shall assume that m is known to the server, so that we do not have to combine probabilities for different nonrandomized sizes. Assume also that a partial support $s_l = \mathrm{supp}_l^T(A)$ approximates the corresponding prior probability $\mathbf{P}\left[\#(t \cap A) = l\right]$. Suppose we know the following prior probabilities:

$$\begin{aligned} s_l^+ &:= \mathbf{P}\left[\#(t \cap A) = l,\ a \in t\right], \\ s_l^- &:= \mathbf{P}\left[\#(t \cap A) = l,\ a \notin t\right]. \end{aligned}$$

Notice that $s_l = s_l^+ + s_l^-$ simply because

$$\#(t \cap A) = l \;\Leftrightarrow\; \left[\begin{array}{l} a \in t \;\;\&\;\; \#(t \cap A) = l, \text{ or} \\ a \notin t \;\;\&\;\; \#(t \cap A) = l. \end{array}\right.$$

Let us use these priors and compute the posterior probability of $a \in t$ given $A \subseteq t'$:

$$\begin{aligned}
\mathbf{P}[a \in t \mid A \subseteq t'] \;&=\; \frac{\mathbf{P}[a \in t,\; A \subseteq t']}{\mathbf{P}[A \subseteq t']} \;=\\
&= \sum_{l=1}^{k} \mathbf{P}[\#(t \cap A) = l,\; a \in t,\; A \subseteq t'] \Big/ \sum_{l=0}^{k} s_l \cdot p[l \to k]\\
&= \sum_{l=1}^{k} \mathbf{P}[\#(t \cap A) = l,\; a \in t] \cdot p[l \to k] \Big/ \sum_{l=0}^{k} s_l \cdot p[l \to k]\\
&= \sum_{l=1}^{k} s_l^+ \cdot p[l \to k] \Big/ \sum_{l=0}^{k} s_l \cdot p[l \to k].
\end{aligned}$$

Thus, in order to prevent privacy breaches of level 50% as defined in Definition 5, we need to ensure that always

$$\sum_{l=1}^{k} s_l^+ \cdot p[l \to k] \;<\; 0.5 \cdot \sum_{l=0}^{k} s_l \cdot p[l \to k]\,. \tag{3.16}$$

The problem is that we have to randomize the data *before* we know any supports. Also, we may not have the luxury of setting "oversafe" randomization parameters because then we may not have enough data to perform a reasonably accurate support recovery. One way to achieve a compromise is to:

1. Estimate maximum possible support $s_{max}(k, m)$ of a k-itemset in the transactions of given size m, for different k and m;

2. Given the maximum supports, find values for s_l and s_l^+ that are most likely to cause a privacy breach;

3. Make randomization just strong enough to prevent such a privacy breach.

Since $s_0^+ = 0$, the most privacy-challenging situations occur when s_0 is small, that is, when our itemset A and its subsets are frequent.

In our experiments we consider a privacy-challenging k-itemset A such that, for every $l > 0$, all its subsets of size l have the maximum possible support $s_{max}(l, m)$. The partial supports for such a test-itemset are computed from the cumulative supports Σ_l using Statement 4. By it and by (3.11), we have ($l > 0$)

$$s_l = \sum_{j=l}^{k} (-1)^{j-l} \binom{j}{l} \Sigma_j, \quad \Sigma_j = \binom{k}{j} s_{max}(j, m) \tag{3.17}$$

since there are $\binom{k}{j}$ j-subsets in A. The values of s_l^+ follow if we note that all l-subsets of A, with a and without, appear equally frequently as $t \cap A$:

$$\begin{aligned} s_l^+ \quad &:= \quad \mathbf{P}\left[\#(t \cap A) = l,\ a \in t\right] \ = \\ &= \quad \mathbf{P}\left[a \in t \mid \#(t \cap A) = l\right] \cdot s_l \ = \ l/k \ \cdot \ s_l. \end{aligned}$$

While one can construct cases that are even more privacy-challenging (for example, if $a \in A$ occurs in a transaction every time any nonempty subset of A does), we found the above model (3.17) and (3.18) to be sufficiently pessimistic on our datasets.

We can now use these formulae to obtain cut-and-paste randomization parameters ρ_m and K_m as follows. Given m, consider all cutoffs from $K_m = 3$ to some K_{max} (usually this K_{max} equals the maximum transaction size) and determine the smallest randomization levels $\rho_m(K_m)$ that satisfy (3.16). Then select (K_m, ρ_m) that gives the best discoverability (by computing the lowest discoverable supports, see Section 3.5.2).

3.5.2 Privacy, Discoverability and Dataset Characteristics

We define the *lowest discoverable support* as the support at which the predicted support of an itemset is four sigmas away from zero, i.e, we can clearly distinguish the support of this itemset from zero. In practice, we may achieve reasonably good results even if the minimum support level is slightly lower than four sigma (as was the case for 3-itemsets in the randomized `soccer`, see below). However, the lowest discoverable support is a nice way to illustrate the interaction between discoverability, privacy breach levels, and data characteristics.

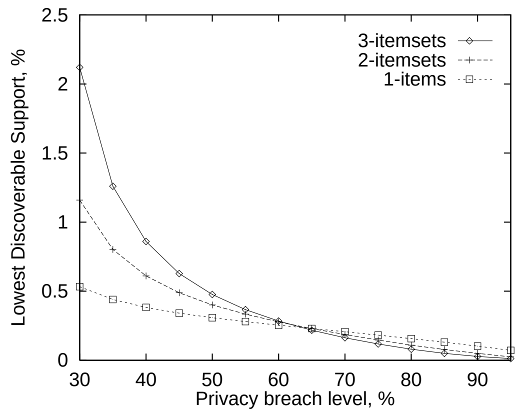

Figure 3.1: Lowest discoverable support for different breach levels. Transaction size is 5, five million transactions.

In order to compute the cut-and-paste randomization parameters for the graphs, we have to fix some values for maximum supports $s_{max}(k, m)$. We do it as follows:

Itemset size:	1	2	3	4	5	6	7	8	9	10
Support(%):	20.0	10.0	5.0	2.0	1.0	0.5	0.2	0.1	0.05	0.02

We set the cutoff value to 5 (except in Fig. 3.3, where it is 7). To set the partial supports of the itemset being mined, we use *independence assumption*: all items within the itemset occur independently of each other and with the same frequency.

Figure 3.1 shows how the lowest discoverable support changes with the privacy breach level. For higher privacy breach levels such as 95% (which could be considered a "plausible denial" breach level), we can discover 3-itemsets at very low supports. For more conservative privacy breach levels such as 50%, the lowest discoverable support is significantly higher. It is interesting to note that at higher breach levels (i.e. weaker randomization) it gets harder to discover 1-itemset supports than 3-itemset supports. This happens because the variance of a 3-itemset predictor depends highly nonlinearly on the amount of false items added while randomizing. When we add fewer false items at higher breach levels, we generate so much fewer false 3-itemset positives than false

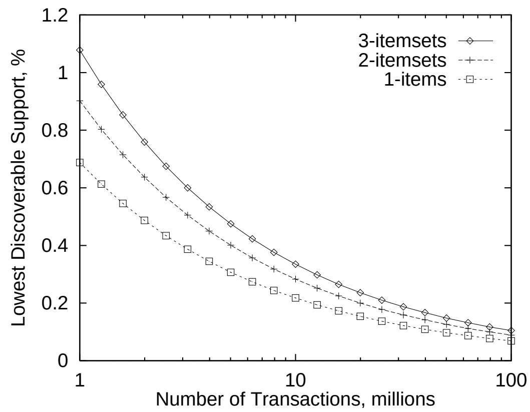

Figure 3.2: Lowest discoverable support versus number of transactions. Transaction size is 5, breach level is 50%.

1-itemset positives that 3-itemsets get an advantage over single items.

Figure 3.2 shows that the lowest discoverable support is roughly inversely proportional to the square root of the number of transactions. Indeed, the lowest discoverable support is defined to be proportional to the standard deviation (square root of the variance) of this support's prediction. If all the partial supports are fixed, the prediction's variance is inversely proportional to the number N of transactions according to Statement 3. In our case, the partial supports depend on N (because the lowest discoverable support does), i.e. they are not fixed; however, this does not appear to affect the variance very significantly (but justifies the word "roughly").

Finally, Figure 3.3 shows that transaction size has a significant influence on support discoverability. In fact, for transactions of size 10 and longer, it is typically not possible to make them both breach-safe and simultaneously get useful information for mining transactions. Intuitively, a long transaction contains too much personal information to hide, because it may contain long frequent itemsets whose appearance in the randomized transaction could result in a privacy breach. We have to insert a lot of false items and cut off many true ones to ensure that such a long itemset in the randomized transaction is

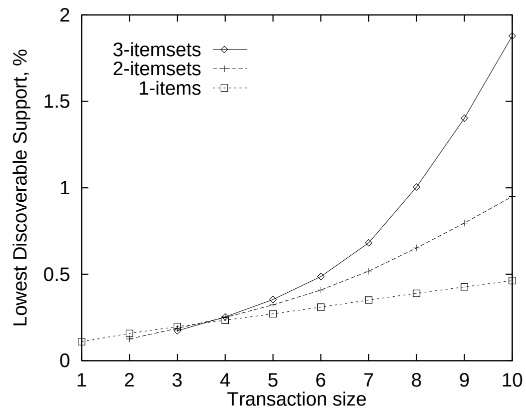

Figure 3.3: Lowest discoverable support for different transaction sizes. Five million transactions, breach level is 50%.

about as likely to be a false positive as to be a true positive. Such a strong randomization causes an exceedingly high variance in the support predictor for 2- and especially 3-itemsets, since it drives down their probability to "tunnel" through while raising high the probability of a false positive. In both our datasets we discard long transactions. The question of how to safely randomize and mine long transactions is left open.

3.5.3 Discoverability of Confidence

The formulae from Section 3.4.3 can be used for computing the radius δ of the interval

$$[\,\mathrm{conf}^T(A \Rightarrow B) - \delta,\ \mathrm{conf}^T(A \Rightarrow B) + \delta\,]$$

around the actual confidence of the rule "$A \Rightarrow B$" such that our estimator $(\mathrm{conf}^T(A \Rightarrow B))_{\mathrm{est}}$ is $p\%$-likely to fall into this interval, where p is any given probability. To find the δ, we should solve the equation $f(\delta) = p$ by means of any suitable numerical algorithm. The δ depends on p, on transaction size m, on sizes of itemsets $|A|$ and $|B|$, on the number N of transactions, and on the parameters of the randomization operator. It also depends on all two-set partial supports for A and B (see Definition 14).

For our confidence graphs we use the interval probability value of 95%. Like we did in the case of support prediction, we define the *lowest discoverable confidence* as the smallest confidence $\mathrm{conf}^T(A \Rightarrow B)$ that is twice as large as the radius of the 95%-interval for its predictor.

In order to set the partial supports for our graphs, we shall use *independence assumption*: all items in $A \cup B$ occur in transactions independently from each other, all items in A have the same frequency, and all items in B have the same frequency. The two frequencies are chosen according to the given $\mathrm{supp}^T(A)$ and $\mathrm{conf}^T(A \Rightarrow B)$. We have also tried *maximum dependence assumption*:

- All items in A and in B are independent from other items in their set and have the same frequency;
- Always if $B \subset t$ then $A \subset t$. Thus,

$$\mathrm{supp}^T(B) \;=\; \mathrm{supp}^T(A) \;\cdot\; \mathrm{conf}^T(A \Rightarrow B)\,;$$

- Transactions t are "filled" with items from B in the order of decreasing $|t \cap A|$, observing the above constraints.

Graph on Fig. 3.5 is computed under maximum dependence assumption, the other graphs are computed under independence assumption. As it turns out, the two assumptions produce similar results, the maximum dependence assumption usually giving a little higher values of lowest discoverable confidence (by several percent or less).

The default parameter settings for the graphs are the same as in Section 3.5.2, including 5 million transactions, each of size 5 items, and privacy breach level of 50%. The default value for the support of A is set to 2%. We plot the results for four sizes of A and B, as given in the graphs. The cutoff for cut-and-paste is set to 5, except in

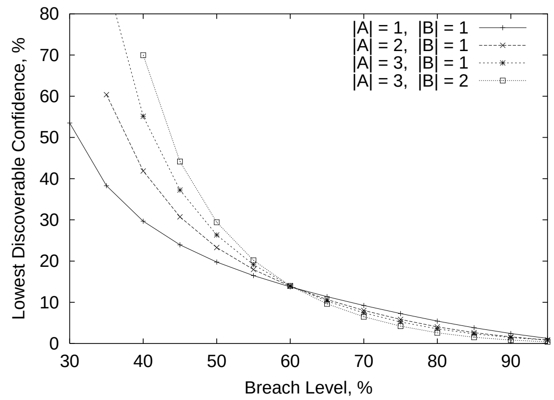

Figure 3.4: Lowest discoverable confidence for different breach levels. Five million transactions, transaction size is 5, $\text{supp}^T(A) = 2\%$.

graphs on Fig. 3.7 and 3.8, where the cutoff is set to 7 (for Fig. 3.7) or to the transaction size (for Fig. 3.8).

Graphs on Fig. 3.4 and 3.5 show how the lowest discoverable confidence depends on the privacy breach level; graph on Fig. 3.6 shows its dependence on the support of the set A; graphs on Fig. 3.7 and 3.8 demonstrate its dependence on transaction size; finally, graph on Fig. 3.9 shows the influence of the number of transactions. The tendencies here are similar to the ones for the lowest discoverable support, with some additional sensitivity to the support of A since its predictor appears in the denominator of the confidence estimator. The graphs demonstrate that our estimator can be used, as long as the confidence being predicted is not low and the support of A is predicted well above zero.

3.5.4 The Datasets

We experimented with two "real-life" datasets. The `soccer` dataset is generated from the clickstream log of the 1998 World Cup web site, which is publicly available at

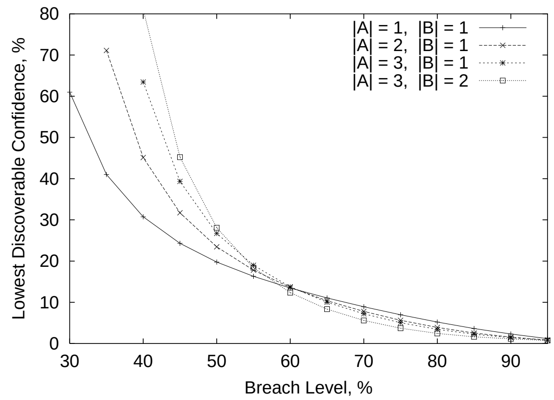

Figure 3.5: Lowest discoverable confidence for different breach levels, under "maximum dependence" assumption. Five million transactions, transaction size is 5, $\mathrm{supp}^T(A) = 2\%$.

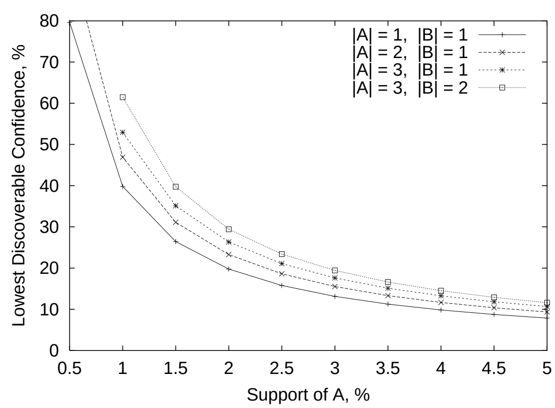

Figure 3.6: Lowest discoverable confidence for different values of $\mathrm{supp}^T(A)$. Five million transactions, transaction size is 5, breach level is 50%.

`ftp://researchsmp2.cc.vt.edu/pub/worldcup/`[3]. We scanned the log and produced a transaction file, where each transaction is a session of access to the site by a client. Each item in the transaction is a web request. Not all web requests were turned into items; to become an item, the request must satisfy the following:

[3]M. Arlitt and T. Jin, "1998 World Cup Web Site Access Logs", August 1998. Available at `http://www.acm.org/sigcomm/ITA/`

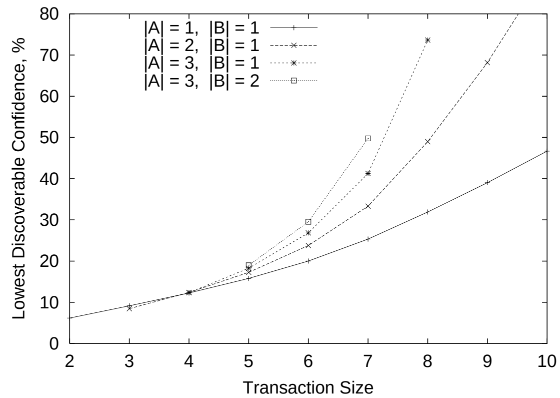

Figure 3.7: Lowest discoverable confidence for different transaction sizes. Five million transactions, breach level is 50%, $\mathrm{supp}^T(A) = 2\%$, cutoff is 7.

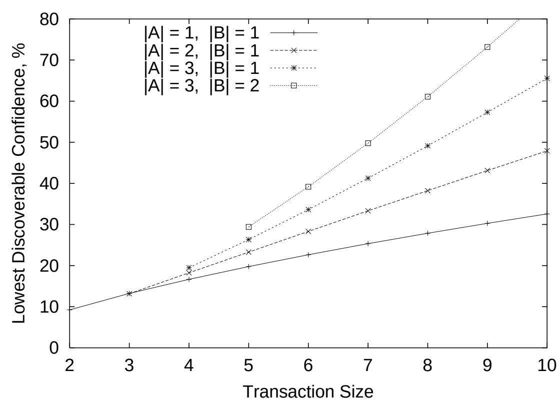

Figure 3.8: Lowest discoverable confidence for different transaction sizes. Five million transactions, breach level is 50%, $\mathrm{supp}^T(A) = 2\%$, cutoff equals transaction size.

1. Client's request method is `GET`;
2. Request status is `OK`;
3. File type is `HTML`.

A session starts with a request that satisfies the above properties, and ends when the last click from this ClientID timeouts. The timeout is set as 30 minutes. All requests in a session have the same ClientID. The `soccer` transaction file was then processed further: we deleted from all transactions the items corresponding to the French and English

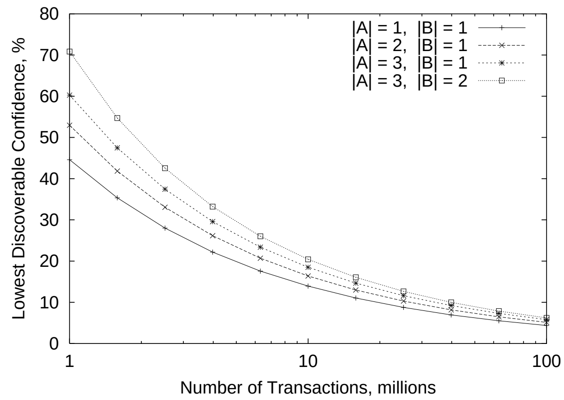

Figure 3.9: Lowest discoverable confidence for different numbers of transactions. Transaction size is 5, breach level is 50%, $\mathrm{supp}^T(A) = 2\%$.

front page frames, and then we deleted all empty transactions and all transactions of size above 10. The resulting `soccer` dataset consists of $6,525,879$ transactions, distributed as shown in Fig. 3.10.

The `mailorder` dataset is the same as that used in [11]. The original dataset consisted of around 2.9 million transactions, 15,836 items, and around 2.62 items per transaction. Each transaction was the set of items purchased in a single mail order. However, very few itemsets had reasonably high supports. For instance, there were only two 2-itemsets with support $\geqslant 0.2\%$, only five 3-itemsets with support $\geqslant 0.05\%$. Hence we decided to substitute all items by their parents in the taxonomy, which had reduced the number of items from 15836 to 96. It seems that, in general, moving items up the taxonomy is a natural thing to do for preserving privacy without losing aggregate information. We also discarded all transactions of size $\geqslant 8$ (which was less than 1% of all transactions) and finally obtained a dataset containing $2,859,314$ transactions (Fig. 3.10).

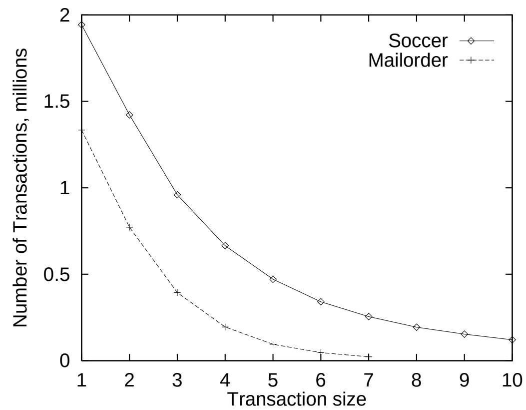

Figure 3.10: Number of transactions for each transaction size in the `soccer` and `mailorder` datasets.

3.5.5 The Results

We report the results for both datasets at a minimum support that is close to the lowest discoverable support, in order to show the resilience of our algorithm even at these very low support levels. We targeted a conservative breach level of 50%, so that, given a randomized transaction, for any item in the transaction it is at least as likely that someone did not buy that item (or access a web page) as that they did buy that item.

We used cut-and-paste randomization (see Definition 9) that has only two parameters, randomization level and cutoff, per each transaction size. We chose a cutoff of 7 for our experiments as a good compromise between privacy and discoverability (for transactions with more than 7 items, the cutoff is 10). Given the values of maximum supports, we then used the methodology from Section 3.5.1 to find the lowest randomization level such that the breach probability (for each itemset size) is still below the desired breach level. The actual parameters (K_m is the cutoff, ρ_m is the randomization level for transaction size m) for `soccer` were:

m	1	2	3	4	5	6	7	8	9	10
K_m	7	7	7	7	7	7	7	10	10	10
$\rho_m\%$	4.7	16.8	21.4	32.2	35.3	42.9	46.1	52.5	57.8	59.7

and for `mailorder` were:

m	1	2	3	4	5	6	7
K_m	7	7	7	7	7	7	7
$\rho_m\%$	8.9	20.4	25.0	33.4	43.5	50.5	59.2

Table 3.1 shows what happens if we mine itemsets from both randomized and non-randomized files and then compare the results. We can see that, even for a low minimum support of 0.2%, most of the itemsets are mined correctly from the randomized file. There are comparatively few false positives (itemsets wrongly included into the output) and even fewer false drops (itemsets wrongly omitted). The predicted sigma for 3-itemsets ranges in $0.061{-}0.062\%$ for `soccer` and in $0.047{-}0.048\%$ for `mailorder`; for 2- and 1-itemsets sigmas are even less.

One might be concerned about the true supports of the false positives. Since we know that there are *many* more low-supported itemsets than there are highly supported, we might wonder whether most of the false positives are outliers, that is, have true support near zero. We have indeed seen outliers; however, it turns out that most of the false positives are not so far off. The tables 3.2 and 3.3 show that usually the true supports of false positives, as well as the predicted supports of false drops, are closer to 0.2% than to zero. This good news demonstrates the promise of randomization as a practical privacy-preserving approach.

We evaluate privacy breaches, i.e., the conditional probabilities from Definition 5, as follows. We count the occurrences of an itemset in a randomized transaction and its sub-items in the corresponding nonrandomized transaction. For example, assume an

Table 3.1: Results on Real Datasets

(a) mailorder, 0.2% minimum support

Itemset Size	True Itemsets	True Positives	False Drops	False Positives
1	65	65	0	0
2	228	212	16	28
3	22	18	4	5

(b) soccer, 0.2% minimum support

Itemset Size	True Itemsets	True Positives	False Drops	False Positives
1	266	250	16	33
2	217	197	20	38
3	48	39	9	27

itemset $\{a, b, c\}$ occurs 100 times in the randomized data among transactions of length 5. Out of these 100 occurrences, 60 of the corresponding original transactions had the item b. We then say that this itemset caused a 60% privacy breach for transactions of length 5, since for these 100 randomized transactions, we estimate with 60% confidence that the item b was present in the original transaction.

Out of all sub-items of an itemset, we choose the item that causes the worst privacy breach. Then, for each combination of transaction size and itemset size, we compute over all frequent[4] itemsets the worst and the average value of this breach level. Finally, we pick the itemset size that gave the worst value for each of these two values.

[4]If there are no frequent itemsets of certain size, we pick the itemsets with the highest support.

Table 3.2: Analysis of false drops

(a) mailorder, $\geqslant 0.2\%$ true support

		predicted support			
size	Itemsets	< 0.1	$0.1–0.15$	$0.15–0.2$	$\geqslant 0.2$
1	65	0	0	0	65
2	228	0	1	15	212
3	22	0	1	3	18

(b) soccer, $\geqslant 0.2\%$ true support

		predicted support			
size	Itemsets	< 0.1	$0.1–0.15$	$0.15–0.2$	$\geqslant 0.2$
1	266	0	1	15	250
2	217	0	4	16	197
3	48	0	0	9	39

Table 3.4 shows the results of the above analysis. To the left of the semicolon is the itemset size that was the worst. For instance, for all transactions of length 5 for soccer, the worst average breach was with 4-itemsets (36.0% breach), and the worst breach was with a 5-itemset (50.4% breach). We can see that, apart from fluctuations, the 50% level is observed where tested.

To prevent biased randomization, for the `soccer` dataset experiment we used a high-quality pseudorandom number generator "Mersenne Twister".[5]

Summary Despite choosing a conservative privacy breach level of 50%, and further choosing a minimum support around the lowest discoverable support, we were able to successfully find most of the frequent itemsets, with relatively small numbers of false

[5] `http://www.math.keio.ac.jp/matumoto/emt.html`

Table 3.3: Analysis of false positives

(a) mailorder, $\geqslant 0.2\%$ predicted support

		true support			
size	Itemsets	< 0.1	$0.1 - 0.15$	$0.15 - 0.2$	$\geqslant 0.2$
1	65	0	0	0	65
2	240	0	0	28	212
3	23	1	2	2	18

(b) soccer, $\geqslant 0.2\%$ predicted support

		true support			
size	Itemsets	< 0.1	$0.1 - 0.15$	$0.15 - 0.2$	$\geqslant 0.2$
1	283	0	7	26	250
2	235	4	7	27	197
3	66	6	11	10	39

drops and false positives.

Table 3.4: Actual Privacy Breaches

`soccer`

Transaction size:	1	2	3	4	5
Worst Average:	1: 4.4%	2: 20.2%	3: 39.0%	4: 35.8%	4: 36.0%
Worst of the Worst:	1: 45.6%	2: 45.6%	3: 52.5%	4: 49.9%	5: 50.4%
Transaction size:	6	7	8	9	10
Worst Average:	4: 31.3%	3: 30.6%	3: 31.6%	3: 30.1%	3: 30.5%
Worst of the Worst:	5: 44.0%	7: 44.3%	7: 45.4%	9: 40.4%	10: 42.3%

`mailorder`

Transaction size:	1	2	3	4	5
Worst Average:	1: 12.0%	2: 27.5%	3: 48.4%	4: 51.5%	5: 51.7%
Worst of the Worst:	1: 47.6%	2: 51.9%	3: 53.6%	4: 53.1%	5: 53.6%

Transaction size:	6	7
Worst Average:	5: 51.9%	6: 49.8%
Worst of the Worst:	6: 55.4%	7: 51.9%

Chapter 4

Limiting Privacy Breaches by Randomization

4.1 Introduction

The material in this Chapter has been previously published in [68]. This work is done in collaboration with Johannes Gehrke (Cornell University) and Ramakrishnan Srikant (IBM Almaden Research Center).

This chapter builds upon the work described in Chapter 3. Here, we present a new formulation of privacy breaches, together with a methodology, "amplification", for limiting them. Unlike earlier approaches, amplification makes it possible to guarantee limits on privacy breaches without any knowledge of the distribution of the original data. We instantiate this methodology for the problem of mining association rules, and augment the randomization algorithm from Chapter 3 to limit privacy breaches without knowledge of the data distribution, replacing the heuristic privacy measure from Section 3.5.1.

The method from Section 3.5.1 was used in Section 3.5 to test performance of randomization and recovery algorithms, but for many practical applications it needs to be improved, for the following three reasons. First, it is designed to limit only one kind of privacy breaches, namely when the occurrence of an itemset in the randomized transaction makes it likely that some item is present in the original transaction (Section 3.2.2, Definition 5). Other possible properties of the original transaction, such as the absence of an item or any logical expression over items, may still be revealed to the server. Second, Section 3.5.1 does not prove that privacy breaches cannot happen; rather, it provides a heuristic that usually ensures privacy. Third, the algorithm requires some information about the distribution of original data (namely, the maximal supports for itemsets of a given size) in order to compute the parameters of randomization. In practice, any information about the original data is learned *after* randomization has been performed, so no properties of prior distribution should be used until then. In the current chapter, we address all these shortcomings through amplification, a new approach for limiting privacy breaches.

Next, in Section 4.4 we address the problem that the amount of randomization required to avoid privacy breaches (when mining association rules) results in very long transactions. By using pseudorandom generators and carefully choosing seeds such that the desired items from the original transaction are present in the randomized transaction, we can send just the seed instead of the transaction, resulting in a dramatic drop in communication and storage cost. In Section 4.5, we define new information measures that take privacy breaches into account when quantifying the amount of privacy preserved by randomization.

Let us recall the problem setup (Section 2.3). We have one server (the company) and many clients (the users); each client has a piece of personal data. The server wants to build an aggregate model of the clients' data, a model that would allow the server to learn those properties that are frequent among the users (and properties that are not frequent). For example, the personal data may be numerical attributes like a client's age and salary, or categorical attributes like a client's preferences for the company's products; the aggregate model may be distribution of the numerical attributes, or the collection of frequent itemset for the products. This model says very little about each client individually, so it is natural to expect that the server does not need to collect the precise personal data to build the model. Instead, the clients randomize their personal records before they are sent to the server, and the server employs statistical methods to recover aggregate information from the randomized data.

It is not enough to simply concentrate on randomization and on recovery of the aggregate model. We must also ensure that the randomization is sufficient for preserving privacy, since we randomized in the first place to achieve privacy. For example, suppose we randomize age x_i by adding a random number r_i drawn uniformly from a segment $[-50, 50]$. Assuming that the server receives age 120 from a user, privacy is somewhat compromised, as the server can conclude that the real age of the user cannot be less than 70 (otherwise $x_i + r_i < 70 + 50 = 120$). Thus the server has learned a potentially valuable piece of information about the client — information that is correct with 100% probability. Analogously, suppose we randomize a small set of items (a transaction) by replacing each item by a random item with probability 80%. If the transaction contains a subset A of 3 items that has a support of 1%, it has $(0.2)^3 = 0.008 = 0.8\%$ chance to retain the same set of three items after the randomization. Thus whenever the server sees A in the randomized transaction, it learns with high probability of the presence of A in the original transaction as well. Indeed, there are $1\% \cdot 0.008 = 0.008\%$ randomized transactions that have A both before and after randomization, while the probability that A occurs in 10 randomly inserted items (out of, say, 10,000 possible items) is less than $10^{-7}\%$ (Section 3.2)

Here we look at two approaches for quantifying how privacy-preserving a randomization method is. One approach relies on information theory [2], the other approach is based on our notion of privacy breaches. The former approach measures the average amount of information disclosed in a randomized attribute by computing the mutual information between the original and the randomized distribution. The latter approach is a worst-case notion, it gives a criterion that should be satisfied by any privacy-preserving algorithm. Intuitively, a *privacy breach* occurs if a property of the original data record gets revealed if we see a certain value of the randomized record. In our previous example, the randomized age of 120 is an example of a privacy breach as it reveals that the actual age is at least 70. As another example, a privacy breach occurs if a subset within a randomized transaction makes it likely that some item occurs in the original transaction.

As we show in Section 4.5, these two approaches are different: privacy breaches can occur even though mutual information is small. We therefore propose other information-theoretical measures, called "worst-case information," that do bound privacy breaches.

Chapter Outline We introduce some basic notation in Section 4.2, followed by an overview of the contributions. We define the augmented notion of privacy breaches in Section 4.2.3, and show how the amplification methodology can limit privacy breaches in Section 4.3. In Section 4.4, we use pseudorandom generators to dramatically reduce communication and storage cost of randomized transactions. We present new information measures that take privacy breaches into account in Section 4.5.

4.2 Generalized Privacy Breaches

4.2.1 Basic Notions

The Model Suppose there are N clients $C_1, \ldots, C_N$ connected to one server; each client C_i has some private information x_i. The server needs to learn certain aggregate (statistical) properties of the clients' data. The clients are comfortable with this, but they are reluctant to disclose their personal information x_i. To ensure privacy, each client C_i sends to the server a modified version y_i of x_i. The server collects the modified information from all clients and uses it to recover the statistical properties it needs.

Assumptions We assume that each client's piece x_i of private information belongs to the same fixed *finite* set V_X. Furthermore, we assume that each x_i is chosen *independently at random* according to the same fixed probability distribution. This distribution, denoted p_X, is not private, the clients allow the server to learn it. The assumption of independence implies that, once p_X is known, the private information x_j of all clients C_j besides client C_i tells nothing new about C_i's own private information x_i.

Randomization Before sending it to the server, each client C_i hides its personal data x_i by applying a *randomization operator* $R(x)$. The output of $R(x_i)$ is random, whose distribution depends on x_i and on nothing else. Only one instance y_i of $R(x_i)$ is sent to the server by client C_i. The set of all possible outputs of $R(x)$ is denoted by V_Y and is assumed to be finite. For all $x \in V_X$ and $y \in V_Y$, the probability that $R(x)$ outputs y is denoted by

$$p\,[x \to y] \;:=\; \mathbf{P}\,[R(x) = y].$$

By receiving y_i from C_i, the server learns something about x_i. Note that, by independence assumption above, all y_j for $j \neq i$ disclose nothing about x_i and can be ignored in privacy analysis; they certainly help the server to learn distribution p_X, but for privacy analysis we assume that the server knows p_X. The problem is to measure how much can be disclosed by y_i about x_i, and to find randomization operators that keep the disclosure limited.

4.2.2 Contributions of This Chapter

Refined Definition of Privacy Breaches A privacy breach is a situation when, for some client C_i, the disclosure of its randomized private information y_i to the server reveals that a certain property of C_i's private information holds with high probability. Privacy breaches were defined in Section 3.2.2 (Definition 4) for transaction datasets; here we refine that definition by explicitly setting the limit to *prior probability* of a property. Prior probability is the likelihood of the property in the absence of any knowledge about C_i's private information; posterior probability is the likelihood of the property given the randomized value y_i. Without a bound on prior probability, there always are properties whose posterior probability is very high even if no information is disclosed, e. g. the property $Q(x) \equiv$ "$x = x$", In Section 4.2.3, we give the new definition (Definition 15) of privacy breaches and then further classify them into upward and downward privacy breaches. We give an example for both kinds of breaches.

Amplification Section 4.3.1 develops a new approach that allows to ensure limitations on privacy breaches for a randomization operator, without any knowledge about the prior distribution p_X and applicable to any property of client's private information. Our privacy preserving restriction involves only the operator's transition probabilities $p\,[x \to y]$:

$$\forall\, x_1, x_2 \in V_X : \; \frac{p\,[x_1 \to y]}{p\,[x_2 \to y]} \;\leqslant\; \gamma \tag{4.1}$$

(see Definition 17 for details). In Statement 5 we prove that if a randomization operator satisfies this condition for some randomized value y, then the disclosure of y to the server has a limited effect at breaching privacy, depending on the value of γ.

Itemset Randomization In Section 4.3.2 we apply amplification (Statement 5) to randomizing itemsets (in the framework of mining association rules). We give a heuristic, based on the solution of an optimization problem, that allows us to choose randomization parameters so that

- the randomization operator satisfies condition (4.1);
- the supports of the original itemsets can be recovered from randomized transactions.

We illustrate the practical utility of our method through some tradeoff charts.

Compression of Randomized Transactions Both in the earlier approaches (Chapter 3) and in the amplification approach for itemset randomization, the randomized transactions may be very long and memory-consuming. Each randomized transaction often contains many thousands of items (order of magnitude more than original transactions); this is needed in order to hide the true items, for preserving privacy. Fortunately, there is a way to "compress" randomized transactions without compromising privacy or support recovery. The idea is to use a *pseudorandom generator* for computing which items belong to each randomized transaction. The seed for the pseudorandom generator, one seed per transaction, is chosen so that the randomized transaction contains or does not contain certain pre-selected items from the original transaction. This seed is sufficient to compute the whole randomized transaction or any portion of it, so it serves as a "compressed" randomized transaction.

Section 4.4 explains how one can construct a suitable pseudorandom generator using error-correcting codes. The method can reduce the size of randomized transactions by several orders of magnitude, without any effect on either privacy or support recovery. The use of the pseudorandom generator results in dropping the full probabilistic independence of "false" items inserted into the randomized transaction, but instead having only q-wise independence for a sufficiently large integer q. Privacy preserving capability of the new randomization operator can be evaluated using amplification.

Worst-Case Information In Section 4.5, we elaborate upon the work in [2] on measures of privacy. We show that the value of the classical mutual information does not ensure safety from privacy breaches, and introduce new information-theoretic privacy measures whose values provably bound privacy breaches. It turns out that two different subclasses of privacy breaches called "upward" and "downward" privacy breaches (Definition 16) are bounded by different measures, though measures are defined in a very similar way. Worst-case information is obtained from mutual information by writing it in terms of the Kullback-Leibler distance and replacing the expectation with the maximum.

4.2.3 Definitions and Examples

Let C_i be any client, let x_i be its private information. For the server, prior to randomization, each possible value x of C_i's private information has probability $p_X(x)$. Let us define a random variable X such that

$$\mathbf{P}\left[X = x\right] \;:=\; p_X(x).$$

Random variable X is the best description of the server's prior knowledge about x_i. Now, suppose that the client randomizes x_i by computing $y_i = R(x_i)$, then sends y_i to the server. From the server's point of view, the randomized value y_i is an instance of a random variable Y such that

$$\mathbf{P}\left[Y = y\right] \;:=\; \sum_{x \in V_X} \mathbf{P}\left[X = x\right] \cdot p\left[x \to y\right].$$

Random variables X and Y are dependent; their joint distribution is given by:

$$\mathbf{P}\left[X = x,\; Y = y\right] \;=\; p_X(x) \cdot p\left[x \to y\right].$$

Given y_i, the server can better evaluate the probabilities of possible values for C_i's private information. It uses Bayes formula and computes posterior probabilities:

$$\mathbf{P}\left[X = x \mid Y = y_i\right] \;:=\; \frac{\mathbf{P}\left[X = x\right] \cdot p\left[x \to y_i\right]}{\mathbf{P}\left[Y = y_i\right]}.$$

We can also find the posterior probability of any *property* $Q(x)$, where $Q : V_X \to \{\text{true},\ \text{false}\}$:

$$\mathbf{P}\left[Q(X) \mid Y = y_i\right] \;=\; \sum_{Q(x),\, x \in V_X} \mathbf{P}\left[X = x \mid Y = y_i\right].$$

Informally, a privacy breach is a situation when, for some property $Q(x)$, the disclosure of y_i to the server significantly increases the probability of this property. If it is important to the client that property $Q(x_i)$ of its private information is not disclosed, then a significant increase in probability may be a violation of privacy. Here is the formal definition of a privacy breach:

Definition 15. We say that there is a ρ_1-to-ρ_2 *privacy breach* with respect to property $Q(x)$ if for some $y \in V_Y$

$$\mathbf{P}[Q(X)] \leqslant \rho_1 \quad \text{and} \quad \mathbf{P}[Q(X) \mid Y = y] \geqslant \rho_2.$$

Here $0 < \rho_1 < \rho_2 < 1$ and $\mathbf{P}[Y = y] > 0$.

Let us consider the following example on privacy breaches.

Example 2. Suppose that private information x is a number between 0 and 1000. This number is chosen as a random variable X such that 0 is 1%-likely whereas any non-zero is only about 0.1%-likely:

$$\begin{aligned} \mathbf{P}[X = 0] &= 0.01 \\ \mathbf{P}[X = k] &= 0.00099, \quad k = 1 \ldots 1000 \end{aligned}$$

Suppose we want to randomize such a number by replacing it with a new random number $y = R(x)$ that retains some information about the original number x. Here are three possible ways to do it:

1. Given x, let $R_1(x)$ be x with 20% probability, and some other number (chosen uniformly at random) with 80% probability.
2. Given x, let $R_2(x)$ be $x + \xi \pmod{1001}$, where ξ is chosen uniformly at random in $\{-100, \ldots, 100\}$.

Table 4.1: Prior and posterior (given $R(X) = 0$) probabilities for properties in Example 2

Given:	$X = 0$	$X \notin \{200, \ldots, 800\}$
nothing	1%	$\approx$ 40.5%
$R_1(X) = 0$	$\approx$ 71.6%	$\approx$ 83.0%
$R_2(X) = 0$	$\approx$ 4.8%	100%
$R_3(X) = 0$	$\approx$ 2.9%	$\approx$ 70.8%

3. Given x, let $R_3(x)$ be $R_2(x)$ with 50% probability, and a uniformly random number otherwise.

In Table 4.1 we compute prior and posterior probabilities of two properties of X: property $Q_1(X) \equiv$ "$X = 0$" and property $Q_2(X) \equiv$ "$X \notin \{200, \ldots, 800\}$." We can see that randomization operator R_1 reveals a lot of information about X when $R_1(X)$ happens to equal zero: the server learns with high probability that X originally was zero. Without knowing that $R_1(X) = 0$, the server considers $X = 0$ to be just 1%-likely; but when $R_1(X) = 0$ is revealed, $X = 0$ becomes about 70%-likely. This does not happen when $R_2(X) = 0$ is revealed, the probability of $X = 0$ becomes only 4.8%. However, a different kind of personal information breaks through: the server knows with 100% *certainty* that X does not lie between 200 and 800. The prior probability of this property is about 40%. Only R_3 seems to be a good privacy preserving randomization.

As Example 2 shows, some randomization operators may not be safe because, if they are used, learning a randomized value sometimes significantly affects posterior probabilities for certain properties of the original private value. To fix this, we either have to make sure that all involved properties are harmless when disclosed to the server, or that no property significantly changes its posterior probability. In this chapter we take the latter approach. According to Definition 15, for $R_1(x)$ we have a 1%-to-70% privacy breach with respect to property $Q_1(x)$, and for $R_2(x)$ we have a 40%-to-100% privacy breach with respect to property $Q_2(x)$.

What changes in probability should we classify as "significant"? In Example 2 there are two kinds of changes:

1. Some property $Q_1(x)$ has very low prior probability (i.e., is *unlikely*), but becomes *likely* once we learn that $R(X) = y$. In Example 2, the property $X = 0$ has a probability jump from 1% to above 70% when $R_1(X) = 0$ is revealed.
2. Some property $Q_2(x)$ has a probability far from 100% (i.e., is *uncertain*), but becomes almost 100%-probable (i.e., almost *certain*) once we learn that $R(X) = y$. Another way of looking at it is by taking a negation: property $\neg Q_2(X)$ is likely, but becomes very unlikely once $R(X) = y$ is revealed. In Example 2, the property "$200 \leqslant X \leqslant 800$" has a downward probability jump from almost 60% to 0% when $R_2(X) = 0$ is revealed, making it certain that either $X < 200$ or $X > 800$.

This observation suggests that there are two important subclasses of privacy breaches. Let us now give the formal definitions for both of these subclasses. Let ρ_1 and ρ_2 be two probabilities, such that ρ_1 corresponds to our intuitive notion of "very unlikely" (e.g., $\rho_1 = 0.01$) whereas ρ_2 corresponds to "likely" (e.g., $\rho_2 = 0.5$); let $Q_1(x)$ and $Q_2(x)$ be two properties.

Definition 16. We say that there is a *straight* or *upward* ρ_1-to-ρ_2 privacy breach with respect to Q_1 if for some $y \in V_Y$

$$\mathbf{P}\left[Q_1(X)\right] \leqslant \rho_1, \quad \mathbf{P}\left[Q_1(X) \mid R(X) = y\right] \geqslant \rho_2.$$

We say that there is an *inverse* or *downward* ρ_2-to-ρ_1 privacy breach with respect to Q_2

if for some $y \in V_Y$

$$\mathbf{P}\left[Q_2(X)\right] \geqslant \rho_2, \quad \mathbf{P}\left[Q_2(X) \mid R(X) = y\right] \leqslant \rho_1.$$

Using property $Q_2' = \neg Q_2$, we could write this as

$$\mathbf{P}\left[Q_2'(X)\right] \leqslant 1 - \rho_2, \quad \mathbf{P}\left[Q_2'(X) \mid R(X) = y\right] \geqslant 1 - \rho_1.$$

We also say that the breach is *caused* by the value $y \in V_Y$ from the inequalities; we assume that $\mathbf{P}\left[R(X) = y\right] > 0$.

So, the probability of $1 - \rho_2$ corresponds to the intuitive notion of "uncertain," and the probability of $1 - \rho_1$ means "almost certain." Our task in the next section is to define sufficient conditions for randomization operators that guarantee no breaches of either kind for any property (for given ρ_1 and ρ_2), regardless of the prior distribution p_X. Then we shall look at the problem of constructing the operators that satisfy these conditions and still allow aggregate data mining.

4.3 Amplification

If we attempt to use Definition 15 directly to check whether a given randomization operator R causes privacy breaches, we immediately encounter two difficulties:

1. There are $2^{|V_X|}$ possible properties, far too many to check them all;
2. We cannot use Definition 15 if we do not know the prior distribution p_X of X. In practice, however, a randomization operator must be chosen before p_X is learned.

It turns out that there exists a sufficient test that has neither of these shortcomings, and there are practically useful randomization operators that satisfy this test. The test is based on comparing the operator's transitional probabilities $p\left[x \to y\right]$ for the same $y \in$

V_Y but different $x \in V_X$. Intuitively, if all of the x-values are reasonably likely to be randomized into a given y, then revealing "$R(x) = y$" does not tell too much about x. We call this approach *amplification* because it limits how much some $p[x \to y]$'s can be amplified with respect to others.

4.3.1 General Approach

Let us define our privacy preserving restriction on randomization operators, and then prove a statement on bounding privacy breaches:

Definition 17. A randomization operator $R(x)$ is *at most γ-amplifying* for $y \in V_Y$ if

$$\forall\, x_1, x_2 \in V_X : \ \frac{p[x_1 \to y]}{p[x_2 \to y]} \ \leqslant \ \gamma; \tag{4.2}$$

here $\gamma \geqslant 1$ and $\exists x : p[x \to y] > 0$. Operator $R(x)$ is *at most γ-amplifying* if it is at most γ-amplifying for all suitable $y \in V_Y$.

Statement 5. *Let R be a randomization operator, let $y \in V_Y$ be a randomized value such that $\exists x : p[x \to y] > 0$, and let $0 < \rho_1 < \rho_2 < 1$ be two probabilities from Definition 16. Suppose that R is at most γ-amplifying for y. Revealing "$R(X) = y$" will cause neither upward ρ_1-to-ρ_2 privacy breach nor downward ρ_2-to-ρ_1 privacy breach with respect to any property if the following condition is satisfied:*

$$\frac{\rho_2}{\rho_1} \cdot \frac{1 - \rho_1}{1 - \rho_2} \ > \ \gamma. \tag{4.3}$$

Proof. Note that $\forall x \in V_X$ we have $p[x \to y] > 0$ because otherwise γ is infinite. Let us denote $Y \equiv R(X)$ as a random variable. Consider any distribution p_X; since it is nonzero on at least one $x \in V_X$, we have

$$\mathbf{P}[Y = y] \ \geqslant \ \mathbf{P}[X = x] \cdot p[x \to y] \ > \ 0.$$

By way of contradiction, let us assume that for property $Q(x)$ we have a ρ_1-to-ρ_2 privacy breach. $Q(x)$ cannot be true for all $x \in V_X$ because $\mathbf{P}[Q(X)] \leqslant \rho_1 < 1$ by the definition of privacy breach. Analogously, $Q(x)$ cannot be false for all $x \in V_X$ because $\mathbf{P}[Q(X) \mid Y = y] \geqslant \rho_2 > 0$. So, the following definitions make sense:

$$x_1 \;\in\; \{x \in V_X \mid Q(x) \text{ and } p\,[x \to y] \;=\; \max_{Q(x')} p\,[x' \to y]\,\}$$

$$x_2 \;\in\; \{x \in V_X \mid \neg Q(x) \text{ and } p\,[x \to y] \;=\; \min_{\neg Q(x')} p\,[x' \to y]\,\}$$

In words, x_1 is a private value that has property $Q(x)$ and is most likely to get randomized into y, and x_2 is another value that does not satisfy $Q(x)$ and is least likely to get randomized into y. By the definition of conditional probability,

$$\begin{aligned}
\mathbf{P}[Q(X) \mid Y = y] \;=\; \sum_{Q(x)} \mathbf{P}[X = x \mid Y = y] \;=& \\
=\; \sum_{Q(x)} \frac{\mathbf{P}[X = x] \cdot p\,[x \to y]}{\mathbf{P}[Y = y]} \;\leqslant& \\
\leqslant\; \frac{p\,[x_1 \to y]}{\mathbf{P}[Y = y]} \cdot \sum_{Q(x)} \mathbf{P}[X = x] \;=\; p\,[x_1 \to y] \cdot \frac{\mathbf{P}[Q(X)]}{\mathbf{P}[Y = y]}&
\end{aligned}$$

and, in the same way,

$$\begin{aligned}
\mathbf{P}[\neg Q(X) \mid Y = y] \;=\; \sum_{\neg Q(x)} \mathbf{P}[X = x \mid Y = y] \;=& \\
=\; \sum_{\neg Q(x)} \frac{\mathbf{P}[X = x] \cdot p\,[x \to y]}{\mathbf{P}[Y = y]} \;\geqslant& \\
\geqslant\; \frac{p\,[x_2 \to y]}{\mathbf{P}[Y = y]} \cdot \sum_{\neg Q(x)} \mathbf{P}[X = x] \;=\; p\,[x_2 \to y] \cdot \frac{\mathbf{P}[\neg Q(X)]}{\mathbf{P}[Y = y]}&
\end{aligned}$$

We know that $\mathbf{P}[Q(X) \mid Y = y] \geqslant \rho_2 > 0$, and it follows from the above that $\mathbf{P}[Q(X)] > 0$. Therefore, we can divide the lower inequality by the upper one:

$$\frac{\mathbf{P}[\neg Q(X) \mid Y = y]}{\mathbf{P}[Q(X) \mid Y = y]} \;\geqslant\; \frac{p\,[x_2 \to y]}{p\,[x_1 \to y]} \cdot \frac{\mathbf{P}[\neg Q(X)]}{\mathbf{P}[Q(X)]}$$

Let us remember that $R(x)$ is at most γ-amplifying for y:

$$\frac{1 - \mathbf{P}[Q(X) \mid Y = y]}{\mathbf{P}[Q(X) \mid Y = y]} \;\geqslant\; \frac{1}{\gamma} \cdot \frac{1 - \mathbf{P}[Q(X)]}{\mathbf{P}[Q(X)]}$$

It remains to notice that

$$\frac{1-\rho_2}{\rho_2} \geqslant \frac{1 - \mathbf{P}[Q(X) \mid Y = y]}{\mathbf{P}[Q(X) \mid Y = y]}; \;\; \frac{1 - \mathbf{P}[Q(X)]}{\mathbf{P}[Q(X)]} \geqslant \frac{1-\rho_1}{\rho_1}$$

and we arrive to contradiction with condition (4.3).

To prove the statement for downward ρ_2-to-ρ_1 breaches, we first represent them as upward ρ'_1-to-ρ'_2 breaches with $\rho'_1 = 1 - \rho_2$ and $\rho'_2 = 1 - \rho_1$, and then note that condition (4.3) stays satisfied:

$$\frac{\rho'_2}{\rho'_1} \cdot \frac{1-\rho'_1}{1-\rho'_2} \;=\; \frac{1-\rho_1}{1-\rho_2} \cdot \frac{\rho_2}{\rho_1} \;>\; \gamma.$$

□

We sometimes call inequality (4.2) *amplification condition* for a given $y \in V_Y$. We need to enforce this condition for all $y \in V_Y$ if we do not want privacy breaches regardless of the randomized private value $R(x)$.

In Example 2, randomization operator R_3 satisfies the amplification condition (4.2) with $\gamma < 6$. Indeed, for this operator, transitional probabilities are

$$p[x \to y] \;=\; \begin{cases} \frac{1}{2}\left(\frac{1}{201} + \frac{1}{1001}\right), & \text{if } y \in [x{-}100, x{+}100] \\ \frac{1}{2}\left(0 + \frac{1}{1001}\right), & \text{otherwise.} \end{cases}$$

Their fractional difference is $1 + 1001/201 < 6$. Using Statement 5, we can claim that there are no ρ_1-to-ρ_2 upward breaches from $\rho_1 = 1/7 \approx 14\%$ to $\rho_2 = 1/2 = 50\%$, nor the corresponding downward breaches. And we do not even need to know p_X to claim this.

Background Knowledge. Amplification condition (4.2) limits privacy breaches in the presence of certain kinds of background information about the clients. Suppose that client C_i has private information x_i, and the server knows the value of some function $f(x_i)$, or more generally, an instance of some random variable Z that depends on x_i. From the server's point of view, the probability distribution of the possible values for x_i (i. e. of random variable X), prior and posterior, becomes conditional:

- Prior: $\mathbf{P}[X = x] \;\to\; \mathbf{P}[X = x \mid Z = z]$
- Posterior: $\mathbf{P}[X = x \mid R(X) = y] \;\to\; \mathbf{P}[X = x \mid R(X) = y,\, Z = z]$

If the background information is independent from the randomization operator, all transitional probabilities $p[x \to y]$ remain the same, so amplification condition remains unaffected and Statement 5 still applies. However, Definition 15 of ρ_1-to-ρ_2 privacy breach in the presence of background knowledge is modified: the breach now occurs when

$$\mathbf{P}[Q(X) \mid Z = z] \leqslant \rho_1 \;\text{ and }\; \mathbf{P}[Q(X) \mid Y = y,\, Z = z] \geqslant \rho_2.$$

4.3.2 Itemset Randomization

Now we are going to show how to construct randomization operators that satisfy amplification condition (4.2) for a given γ and still allow for aggregate data mining by the server. This will be done for one important special case, previously discussed in Chapter 3: randomization of itemsets in association rule mining. Let us start with defining the problem.

Let $\mathcal{I}$ be a set of items, for example products in an on-line store. Suppose there are N clients, each having a *transaction* t_i, where t_i is a subset of $\mathcal{I}$. The items in t_i may represent purchases or preferences of client i. We assume that all transactions have the same size m and that each transaction is an independent instance of some distribution that is not hidden. In real life, transactions have different sizes, but the server can group together transactions according to their nonrandomized size if the size is not hidden.

The server wants to learn itemsets $A \subset \mathcal{I}$ that occur frequently within transactions. That is, it needs all itemsets whose support

$$\text{sup}(A) \;:=\; \frac{\#\{i \mid i = 1 \ldots N,\, A \subseteq t_i\}}{N}$$

is at least some minimal support $s_{\min}$. However, the clients are not willing to disclose their personal transactions, so they use randomization. Here we are going to consider the class of randomizations called “select-a-size,” defined in Section 3.3. The definition was as follows:

Definition 18. The *select-a-size* randomization operator has parameters $0 \leqslant \rho \leqslant 1$ and $\{p[j]\}_{j=0}^{m}$, the latter being a probability distribution over $\{0, 1, \ldots, m\}$. Given a transaction t, the operator generates another transaction $t' = R(t)$ in three steps:

1. The operator selects an integer j at random from the set $\{0, 1, \ldots, m\}$ so that $\mathbf{P}\,[j \text{ is chosen}] = p[j]$;
2. It selects j items from t, uniformly at random (without replacement). These items, and no other items of t, are placed into t';
3. It considers each item $a \notin t$ in turn and tosses a coin with probability ρ of “heads” and $1 - \rho$ of “tails”. All those items for which the coin faces “heads” are added to t'.

Let us constrain the select-a-size operator with our amplification condition, to ensure the desired limitation on privacy breaches. We shall use the non-strict form (4.2), because it will allow us to solve an optimization problem. Denote $t' = R(t)$, $m' = |t'|$, $j = |t \cap t'|$, and $n = |\mathcal{I}|$. Then the transitional probabilities of the select-a-size can be written as

$$p\,[t \to t'] \;=\; \frac{p[j]}{\binom{m}{j}} \cdot \rho^{m'-j}(1-\rho)^{n-m-m'+j}.$$

If there are two transactions t_1 and t_2 with $j_1 = |t_1 \cap t'|$ and $j_2 = |t_2 \cap t'|$, we have

$$\frac{p\,[t_1 \to t']}{p\,[t_2 \to t']} \;=\; \frac{\left(\dfrac{p[j_1]}{\binom{m}{j_1} \cdot \rho^{j_1}(1-\rho)^{m-j_1}}\right)}{\left(\dfrac{p[j_2]}{\binom{m}{j_2} \cdot \rho^{j_2}(1-\rho)^{m-j_2}}\right)}.$$

We can call

$$p^{\mathrm{def}}[j] \;:=\; \binom{m}{j} \cdot \rho^j (1-\rho)^{m-j}$$

the "default" probability of selecting j items from t, and "balance" the $p[j]$'s by dividing them by the "default" probabilities:

$$b[j] \;:=\; p[j] \;/\; p^{\mathrm{def}}[j] \tag{4.4}$$

Then condition (4.2) becomes

$$\forall j_1, j_2: \quad b[j_1] \;/\; b[j_2] \;\leqslant\; \gamma. \tag{4.5}$$

While satisfying this condition, we want to transmit as much aggregate information as possible. Randomized transactions are used by the server in order to determine frequent itemsets. So, we would like to ensure that frequent itemsets in randomized transactions have supports as different as possible from infrequent itemsets, with respect to the standard deviation of the supports. Among the parameters of select-a-size, ρ determines the amount of new items added, and $\{p[j]\}_{j=0}^m$ determines the amount of original items deleted. Given ρ, a reasonable heuristic is to set the $p[j]$'s so that, on average, as many original items as possible make it to the randomized transaction. Thus, we are maximizing the following expectation:

$$\nu(p) \;:=\; \mathop{\mathbf{E}}_{t' \sim R(t)} |t \cap t'| \;=\; \sum_{j=0}^{m} j \cdot p[j]$$

Statement 6. *For any nonconstant function $f(j)$ increasing on $j = 0 \ldots m$, the quantity*

$$\nu_f(p) \;:=\; \sum_{j=0}^{m} f(j) \cdot p[j] \;=\; \sum_{j=0}^{m} b[j] \cdot f(j)\, p^{\mathrm{def}}[j]$$

reaches maximum (conditioned by (4.5) and by $\{p[j]\}_{j=0}^{m}$ being a probability distribution) when, for some $j_* \in \{0, 1, \ldots, m-1\}$, we have

$$\gamma \cdot b[0] = \gamma \cdot b[1] = \ldots = \gamma \cdot b[j_*] = b[j_* + 1] = \ldots = b[m]. \tag{4.6}$$

The proof of this statement is in Section 4.6.1.

If, instead of trying to have more *items* of t in t', we are trying to have more k-*itemsets* of t in t', then we are maximizing

$$\mathop{\mathbf{E}}_{t' \sim R(t)} \#\{A \subseteq t \mid A \subseteq t', |A| = k\} \;=\; \sum_{j=0}^{m} \binom{j}{k} \cdot p[j],$$

which is also subject to Statement 6 since function $f(j) = \binom{j}{k}$ is increasing. So, the solution again has the form (4.6), possibly with a different j_*.

Our heuristic thus reduces the problem of selecting parameters ρ and $\{p[j]\}_{j=0}^{m}$ to the problem of selecting ρ and j_*, where j_* is discrete. How to set these two parameters depends on the expected properties of the data, such as how many items are in the itemsets we are mining and what supports these itemsets and their subsets are likely to have. We can use methods from Section 3.4 to evaluate the variance in the support estimators, with extra caution when inverting the transition matrix for partial supports since it may be singular for some ρ and j_*.

We computed how much is recoverable after a select-a-size randomization whose parameters are restricted by the amplification condition. The graphs presented here are similar to those in Section 3.5. Again, we use the notion of the *lowest discoverable* *support* (LDS), which is the lowest possible support that, when recovered after randomization, has a statistically significant separation from zero. By "statistically significant" we mean a separation from zero by four standard deviations. We have computed LDS, in percent, for 1-item, 2-item, and 3-itemsets while varying three numbers:

1. The privacy breach level ρ_1 (in percent), which we define as the least prior probability for an allowed ρ_1-to-ρ_2 privacy breach with $\rho_2 = 50\%$;
2. The transaction size;
3. The number of transactions used for support recovery.

The amplification parameter γ is computed according to formula (4.3) of Statement 5:

$$\gamma \;=\; \frac{\rho_2}{\rho_1} \cdot \frac{1-\rho_1}{1-\rho_2} \;=\; \frac{0.5}{\rho_1} \cdot \frac{1-\rho_1}{0.5} \;=\; \frac{1}{\rho_1} - 1.$$

Parameters j_* and ρ are chosen to minimize the maximum of 1-item, 2-item, and 3-item LDS.

Figure 4.1 shows how LDS depends on the privacy requirement. We require that there are no breaches with the prior below ρ_1 and posterior at 50%, where $\rho_1 = 1\% \ldots 10\%$. As we see, we can recover supports of about 0.5% when the worst breaches (to 50%) allowed are from 5% to 50%.

The graph on Fig. 4.2 has its ρ_1 fixed at 5%, but varies transaction size from 3 to 10. Of course, the longer the transaction, the harder it is to recover supports, since there is more private data to be randomized. Finally, the graph on Fig. 4.3 shows how the number of transactions affects the recovery (in other graphs the default is 5 million transactions). LDS is roughly inversely proportional to the square root of the number of transactions.

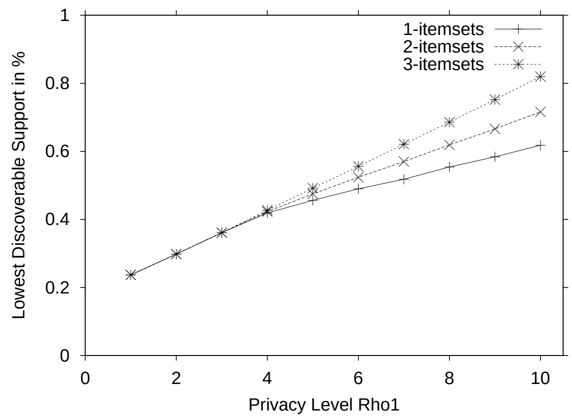

Figure 4.1: Lowest discoverable support versus breach level ρ_1. 5 million transactions, transaction size is 5.

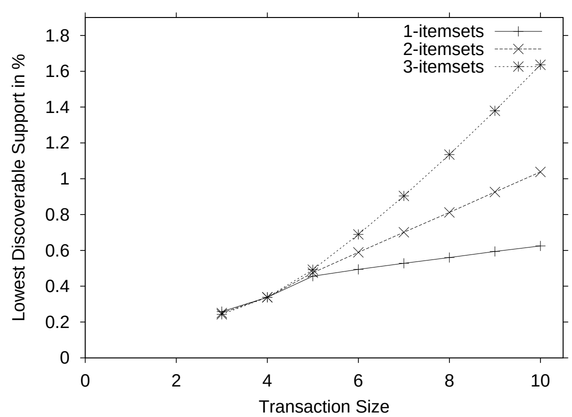

Figure 4.2: Lowest discoverable support versus transaction size. 5 million transactions, breach level is $\rho_1 = 5\%$.

4.4 Compressing Randomized Transactions

When applying select-a-size randomization operator (Definition 18) to transactions, we generate randomized transactions with lots of false items. In fact, the size of each randomized transaction is comparable to the overall number of considered items, which may be in many thousands. Sending these randomized transactions may take significant network resources, and such a database will require a lot of memory. Fortunately, there

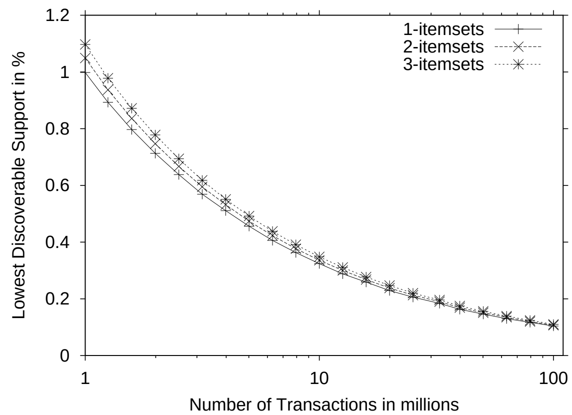

Figure 4.3: Lowest discoverable support versus number of transactions. Transaction size is 5, breach level is $\rho_1 = 5\%$.

is a way to compress randomized transactions without causing privacy breaches. The idea is to use a pseudorandom number generator and drop the requirement of treating each item independently.

Definition 19. A $(\mathrm{Seed}, n, q, \rho)$-*pseudorandom generator* is a function

$$G: \ \mathrm{Seed} \times \{1, \ldots, n\} \to \{0, 1\}$$

that has the following properties:

1. $\forall i = 1 \ldots n: \ \ \mathbf{P}\left[G(\xi, i) = 1 \mid \xi \in_r \mathrm{Seed}\right] \ = \ \rho;$
2. For all integers $1 \leqslant i_1 < i_2 < \ldots < i_q \leqslant n$ and when $\xi \in_r \mathrm{Seed}$, the random variables $G(\xi, i_1), G(\xi, i_2), \ldots, G(\xi, i_q)$ are statistically independent.

Here Seed is a finite set, n and q are positive integers, $0 < \rho < 1$, and the sign "$\in_r$" means "is chosen uniformly at random from."

Let n be the overall number of possible items. Instead of representing a randomized transaction by the list of items it contains, we are going to represent it by a seed $\xi \in \mathrm{Seed}$. Then, for every item number i, we can check whether or not this item belongs to the randomized transaction by computing $G(\xi, i)$: item i belongs to the transaction if and only if $G(\xi, i) = 1$. In other words, there is a mapping τ from seeds to transactions:

$$\tau(\xi) \;=\; \{\text{item } i \mid G(\xi, i) = 1\}. \tag{4.7}$$

The set Seed in many cases can be the set of Boolean strings $\{0, 1\}^k$, where $k \ll n$.

Suppose we want to randomize transaction t that has m items. We shall define a randomization operator (called *pseudorandom select-a-size*) that uses a pseudorandom generator. The operator is similar to select-a-size operator from Definition 18 and has the same parameters: $0 < \rho < 1$ and $\{p[j]\}_{j=0}^{m}$, the latter being a probability distribution over $\{0, 1, \ldots, m\}$. Given a transaction t and a $(\mathrm{Seed}, n, q, \rho)$-pseudorandom generator with $q \geqslant m$, the operator generates the seed $\xi = R'(t)$ in three steps:

1. The operator selects an integer j at random from the set $\{0, 1, \ldots, m\}$ so that $\mathbf{P}\,[j \text{ is chosen}] = p[j]$;
2. It selects j items from t, uniformly at random (without replacement). Without loss of generality, assume that items $t[1], t[2], \ldots, t[j]$ are selected.
3. It selects a random seed $\xi \in \mathrm{Seed}$ such that

$$G(\xi, t[1]) = \ldots = G(\xi, t[j]) = 1 \;\text{ and }\; G(\xi, t[j+1]) = \ldots = G(\xi, t[m]) = 0.$$

   Any seed that satisfies this condition must have equal chances to be selected. This seed is returned as (the seed for) the randomized transaction.

Pseudorandom select-a-size operator will always find some seed at Step 3 because, if we take $\xi \in_r \mathrm{Seed}$, then by Definition 19 the variables $G(\xi, t[1])$, $G(\xi, t[2])$, ..., $G(\xi, t[m])$ are statistically independent and therefore can take any combination of values. Moreover, the following statement shows that a transaction randomized by pseudorandom select-a-size operator R' has the same distribution with respect to any *small* subset of items as when it is randomized by the "usual" select-a-size R from Definition 18:

Statement 7. *Let t be a transaction, $|t| = m$, and let $G(\xi, i)$ be a $(\mathrm{Seed}, n, q, \rho)$-pseudorandom generator with $q > m$. Let $R(t)$ be the "usual" select-a-size operator (Definition 18) with parameters ρ and $\{p[j]\}_{j=0}^{m}$, and let $R'(t)$ be pseudorandom select-a-size operator with the same parameters, with G as its pseudorandom generator. For any itemset A of size at most $q - m$ items and for any $B \subseteq A$ we have (see (4.7) for the definition of τ):*

$$\mathbf{P}\left[R(t) \cap A = B\right] \quad = \quad \mathbf{P}\left[\tau(R'(t)) \cap A = B\right]. \tag{4.8}$$

Proof. Let us pay attention only to the items within set $A \cup t$. There are at most q such items. By the definition of pseudorandom generator (Definition 19), as long as seed ξ is chosen uniformly at random, the values of $G(\xi, i)$ for $i \in A \cup t$ are independent and equal 1 with probability ρ. The first two steps of both randomization operators are the same: we select which subset $t_0 \subseteq t$ is going to belong to the randomized transaction. During the last step,

- In the "usual" operator, each item from $A \setminus t$, independently, has probability ρ to get into $R(t)$;
- In the pseudorandom operator, we select a seed $\xi \in_r \mathrm{Seed}$ such that $\tau(R'(t)) \cap t = t_0$. Since $\xi \in_r \mathrm{Seed}$, the distribution of items from $A \setminus t$ is not affected by the choice of t_0; each item, independently, has probability ρ to get into $\tau(R'(t))$.

So, both operators induce identical distributions on items within $A \setminus t$, and in particular, satisfy (4.8). □

It follows from Statement 7 that all the mathematical apparatus for support and variance estimation from Sections 3.3 and 3.4 is applicable to pseudorandom select-a-size operators as well, as long as we are working with itemsets of size at most $q-m$. Indeed, pseudorandom operator $R(t)$ is a per-transaction operator (it randomizes each transaction independently and its distribution is defined by t). Generally speaking, it is not item-invariant; however, for an itemset A of size at most $q-m$ we have

$$\mathbf{P}\left[\,|\tau(R'(t)) \cap A| = l'\,\right] \;=\; \sum_{B \subseteq A,\, |B| = l'} \mathbf{P}\left[\tau(R'(t)) \cap A = B\right] \;=\; \\ =\; \sum_{B \subseteq A,\, |B| = l'} \mathbf{P}\left[R(t) \cap A = B\right] \;=\; \mathbf{P}\left[\,|R(t) \cap A| = l'\,\right] \;=\; p\left[l \to l'\right]$$

where $p\left[l \to l'\right]$ is defined (see Definition 11) as

$$p\left[l \to l'\right] \;=\; \mathbf{P}\left[\,|R(t) \cap A| = l' \;\mid\; |t \cap A| = l\,\right]$$

and is shown to depend only on l, l', m, $|A|$, and on the parameters of select-a-size randomization. Therefore, we can “bypass” item-invariance.

Now let us find out when pseudorandom select-a-size operator protects from privacy breaches. Here we can no longer restrict ourselves to a few items only, since all items at once are involved in a privacy breach. Instead, we can use the amplification condition (4.2) and Statement 5 in the same way as we used them for the “usual” select-a-size operator in Section 4.3.2. The following statement shows that the amplification condition in pseudorandom case translates into exactly the same condition (4.5) on the randomization parameters:

Statement 8. *Let $R(t)$ and $R'(t)$ be the “usual” and the pseudorandom select-a-size operators respectively, with the same randomization parameters; suppose that R' uses a $(\mathrm{Seed}, n, q, \rho)$-pseudorandom generator with $q \geqslant m = |t|$. Then*

$$\forall t_1, t_2, \xi: \;\; \frac{\mathbf{P}\left[R'(t_1) = \xi\right]}{\mathbf{P}\left[R'(t_2) = \xi\right]} \;=\; \frac{\mathbf{P}\left[R(t_1) = \tau(\xi)\right]}{\mathbf{P}\left[R(t_2) = \tau(\xi)\right]}.$$

Proof. Consider any seed $\xi \in \text{Seed}$ and any transactions t_1 and t_2 of size m. Suppose $\tau(\xi) \cap t_1 = t_1^0$ and $\tau(\xi) \cap t_2 = t_2^0$, and let $j_i = |t_i^0|$ for $j = 1, 2$. Then

$$\begin{aligned}
&\mathbf{P}\left[R'(t_i) = \xi\right] \; = \\
&= \; \mathbf{P}\left[R'(t_i) = \xi \;\mid\; \tau(R'(t_i)) \cap t_i = t_i^0\right] \cdot \mathbf{P}\left[t_i^0 \text{ chosen at Step 2}\right] \\
&= \; \mathbf{P}\left[\xi_r = \xi \;\mid\; \xi_r \in_r \text{Seed},\; \tau(\xi_r) \cap t_i = t_i^0\right] \,\cdot\, p[j_i] \binom{m}{j_i}^{-1} \\
&= \; \frac{\mathbf{P}\left[\xi_r = \xi \;\mid\; \xi_r \in_r \text{Seed}\right]}{\mathbf{P}\left[\tau(\xi_r) \cap t_i = t_i^0 \;\mid\; \xi_r \in_r \text{Seed}\right]} \,\cdot\, p[j_i] \binom{m}{j_i}^{-1} \\
&= \; \frac{|\text{Seed}|^{-1}}{\rho^{j_i}(1-\rho)^{m-j_i}} \,\cdot\, p[j_i] \binom{m}{j_i}^{-1}
\end{aligned}$$

For the "usual" operator $R(t)$, this probability is (for $t' = \tau(\xi)$, $|t'| = m'$):

$$\begin{aligned}
&\mathbf{P}\left[R(t_i) = t'\right] \; = \; p[j_i] \binom{m}{j_i}^{-1} \cdot \rho^{m'-j_i}(1-\rho)^{n-m-m'+j_i} \\
&= \frac{\rho^{m'}(1-\rho)^{n-m'}}{\rho^{j_i}(1-\rho)^{m-j_i}} \,\cdot\, p[j_i] \binom{m}{j_i}^{-1}
\end{aligned}$$

If we divide two probabilities like this, the constant multiplier will cancel out:

$$\frac{\mathbf{P}\left[R'(t_1) = \xi\right]}{\mathbf{P}\left[R'(t_2) = \xi\right]} \;=\; \frac{\mathbf{P}\left[R(t_1) = \tau(\xi)\right]}{\mathbf{P}\left[R(t_2) = \tau(\xi)\right]} \;=\; \frac{b[j_1]}{b[j_2]}$$

where $b[j_i]$ were defined in (4.4). $\square$

As a consequence of Statement 8, all methodology described in Section 4.3.2 for select-a-size randomization operators can be applied for pseudorandom select-a-size operators.

It remains to construct an example of a $(\text{Seed}, n, q, \rho)$-pseudorandom generator. For $\rho = 1/2$, these generators, also known as *orthogonal arrays* [91], can be constructed using linear error-correcting codes [115, 132]. A binary linear *error-correcting code* of size n and distance d is the kernel $\{x \in \mathbf{Z}_2^n \mid Mx = \vec{0}\}$ of a $(k \times n)$-matrix M (called the *parity check matrix*) over the field $\mathbf{Z}_2$ of residues modulo 2 such that any nonzero n-dimensional vector x from the kernel has at least d nonzero coordinates.

The following statement is well-known:

Statement 9. *In a parity check $(k \times n)$-matrix for an error-correcting code of distance d any collection of $d-1$ columns is linearly independent over $\mathbf{Z}_2$. If a vector ξ is chosen uniformly at random from $\mathbf{Z}_2^k$, then in $M^T\xi$ any collection of $q = d-1$ bits is distributed as q independent random bits, each with probability $1/2$ of being zero.*

The proof of this statement is in Section 4.6.2.

Let n be the number of all possible items, let m be the original transaction size (considered fixed), and let ρ be the default probability of an item in the select-a-size operator, represented in the form $\rho = a/2^b$ where a and b are integers. Suppose the server is interested in supports of itemsets of size up to s, but no more; then we need a $(\mathrm{Seed}, n, q, \rho)$-pseudorandom generator with $q = m + s$. Consider an error-correcting code with size bn and distance $d = bq + 1$; let M be its parity check matrix with k rows, and let $\mathrm{Seed} = \{0, 1\}^k$.

Given $\xi \in \{0,1\}^k$ and $i \in \{1, \ldots, n\}$, our pseudorandom generator computes a bit as follows:

1. Compute vector $x = M^T\xi$ over $\mathbf{Z}_2$;
2. Take the following subvector of size b bits:

$$x_i \;=\; \langle x[b(i-1)],\, x[b(i-1)+1],\, \ldots,\, x[b\,i-1]\rangle$$

3. Output 1 if and only if

$$\sum_{j=0}^{b-1} x\,[b(i-1)+j] \cdot 2^j \;<\; a.$$

This pseudorandom generator satisfies Definition 19. Indeed, by Statement 9, if $\xi \in_r \{0,1\}^k$ then any combination of bq bits of $x = M^T\xi$ is independently distributed, each bit being 1 with probability $1/2$. As a consequence, any combination of q disjoint

b-bit subvectors is independently distributed, and each b-bit subvector is "showing" a binary representation of a number below a with probability $a/2^b = \rho$.

How well can we compress randomized transactions using error-correcting codes? Consider, for example, the Bose-Chaudhuri-Hocquenghem (BCH) codes [115, 132]; there, for any positive integers r and $l \leqslant 2^{r-1} - 1$, we have a parity check matrix of size $rl \times (2^r - 1)$ with distance $2l + 1$. If we are dealing with transactions of size $m = 10$ and are interested in itemsets of size up to $s = 5$ and if $\rho = 1/2$ making $b = 1$, for example, then we need distance $b(m + s) + 1 = 16$, which makes $l = 8$. If there are $100,000$ items overall, we need $r = 17$, and hence the size of the compressed transaction is $rl = 136$ bits, much less than the ordinary way which needs $100,000$ bits. For $\rho = 1/16$ we have $b = 4$ and $b(m+s)+1 = 61$ making $l = 30$, $r = 19$, and compressed transaction becomes 570 bits, while ordinary way needs at least $H(1/16) \cdot 10^5 > 33,729$ bits.

4.5 Worst-Case Information

Amplification approach from Section 4.3 is designed to be independent on the prior distribution, to depend only on the randomization operator itself. There can be other ways to restrict disclosure, other privacy measures that depend both on the prior distribution of private data and on the operator. In this section we consider a class of privacy measures inspired by Shannon's information theory [149], adjusted so that they bound privacy breaches.

In the paper [2] the authors introduce a measure of privacy which is a function of mutual information between two distributions, the original data distribution and the randomized data distribution. Suppose that X is a random variable such that each data record is its independent instance. Let $Y = R(X)$ be another random variable (R is randomization) such that each randomized data record is an instance of Y. Then mutual information $I(X;Y)$ is

$$I(X;Y) := KL\left(p_{X,Y} \parallel p_X p_Y\right) = \\ = \mathop{\mathbf{E}}_{y \sim Y} KL\left(p_{X|Y=y} \parallel p_X\right)$$

where $KL\left(p_1 \parallel p_2\right)$ is Kullback-Leibler distance between the distributions $p_1(x)$ and $p_2(x)$ of two random variables:

$$KL\left(p_1 \parallel p_2\right) := \mathop{\mathbf{E}}_{x \sim p_1} \log \frac{p_1(x)}{p_2(x)},$$

$$p_{X,Y}(x,y) := \mathbf{P}\left[X = x, Y = y\right],$$

$$p_{X|Y=y}(x) := \mathbf{P}\left[X = x \mid Y = y\right].$$

It is assumed that the larger $I(X;Y)$ is, the less privacy is preserved. Unfortunately, there are situations where privacy is obviously not preserved, but mutual information does not show any sign of trouble. Here is an example:

Example 3. Let our private data be just one bit: $V_X = \{0, 1\}$. Assume that both 0 and 1 are equally likely: $\mathbf{P}\left[X = 0\right] = \mathbf{P}\left[X = 1\right] = 1/2$. Now consider two randomizations, $Y_1 = R_1(X)$ and $Y_2 = R_2(X)$. The first randomization, given $x \in V_X$, outputs x with probability 60% and outputs $1 - x$ with probability 40%:

$$\mathbf{P}\left[Y_1 = x \mid X = x\right] = 0.6,$$

$$\mathbf{P}\left[Y_1 = 1 - x \mid X = x\right] = 0.4$$

The second randomization R_2 can output 0, 1, or “empty record” e. Whatever its input x is, it outputs e with probability 99.99%, otherwise it outputs x with probability

0.0099% and $1-x$ with probability 0.0001%:

$$\begin{aligned}
\mathbf{P}[Y_2 = e \mid X = x] &= 0.9999, \\
\mathbf{P}[Y_2 = x \mid X = x] &= 0.000099 = 99 \cdot 10^{-6}, \\
\mathbf{P}[Y_2 = 1-x \mid X = x] &= 0.000001 = 1 \cdot 10^{-6}.
\end{aligned}$$

Intuitively, R_2 is a very poor randomizer since if we see, say, $Y_2 = 1$, then we know with very high probability that $X = 1$:

$$\mathbf{P}[X = 1 \mid Y_2 = 1] \;=\; \mathbf{P}[X = 0 \mid Y_2 = 0] \;=\; \frac{99 \cdot 10^{-6} \cdot 0.5}{99 \cdot 10^{-6} \cdot 0.5 + 1 \cdot 10^{-6} \cdot 0.5} \;=\; 0.99$$

For Y_1, this probability is only 0.6, which is much more reasonable. What does mutual information indicate, however? Let us compute $KL\left(p_{X|Y_i=y} \parallel p_X\right)$ for $i = 1, 2$ and $y = 0, 1, e$:

$$\begin{aligned}
& y = 0,1: \; \log \frac{\mathbf{P}[X = y \mid Y_1 = y]}{\mathbf{P}[X = y]} = \log \frac{0.6}{0.5} \approx 0.2630, \\
& \log \frac{\mathbf{P}[X = 1-y \mid Y_1 = y]}{\mathbf{P}[X = 1-y]} = \log \frac{0.4}{0.5} \approx -0.3219, \\
& KL\left(p_{X|Y_1=y} \parallel p_X\right) \approx \\
& \approx 0.6 \cdot 0.2630 - 0.4 \cdot 0.3219 \approx 0.02905 \\
& y = 0,1: \; \log \frac{\mathbf{P}[X = y \mid Y_2 = y]}{\mathbf{P}[X = y]} = \log \frac{0.99}{0.5} \approx 0.9855, \\
& \log \frac{\mathbf{P}[X = 1-y \mid Y_2 = y]}{\mathbf{P}[X = 1-y]} = \log \frac{0.01}{0.5} \approx -5.6439, \\
& KL\left(p_{X|Y_2=y} \parallel p_X\right) \approx \\
& \approx 0.99 \cdot 0.9855 - 0.01 \cdot 5.6439 \approx 0.91921; \\
& y = e,\; x = 0,1: \; \log \frac{\mathbf{P}[X = x \mid Y_2 = e]}{\mathbf{P}[X = x]} = \log \frac{0.5}{0.5} = 0, \\
& KL\left(p_{X|Y_2=e} \parallel p_X\right) = 0
\end{aligned}$$

Now we can compute and compare mutual informations. For Y_1, both of $KL\,(p_{X|Y_1=y} \parallel p_X)$ for $y = 0, 1$ are the same, so the average is

$$I(X; Y_1) \approx 0.02905;$$

For Y_2, the Kullback-Leibler distances are very different, and since **P** $[Y_2 = e] = 0.9999$, the average is

$$I(X; Y_2) \approx 0.9999 \cdot 0 + 0.0001 \cdot 0.91921 \ll I(X; Y_1).$$

Thus, counter to intuition, mutual information says that R_2 is more privacy-preserving than R_1.

Of course, mutual information fails to detect privacy breaches in Example 3 because they are very infrequent: they occur only in 0.01% randomizations. But once a breach occurs, it is detectable, and noone wants to be the unfortunate client who has the breach. Mutual information averages all Kullback-Leibler distances; however, by looking at these distances without taking the average, some breaches become visible. Indeed, in Example 3, distances $KL\,(p_{X|Y_1=y} \parallel p_X)$ for R_1 are both small ($\approx$ 0.02905), whereas for R_2 some distances are big, e.g.

$$KL\,(p_{X|Y_2=1} \parallel p_X) \;\approx\; 0.91921$$

This indicates that revealing "$Y_2 = 1$" may lead to a privacy breach. The measure that shows the worst possible Kullback-Leibler distance rather than averages them will do better at measuring privacy. We come to the following definition:

Definition 20. Let X and Y be discrete random variables. We define *worst-case information* as follows:

$$I_w(X; Y) \;:=\; \max_y \; KL\,(p_{X|Y=y} \parallel p_X).$$

Instead of the logarithm, we can use a different numerical function $f(t)$ as long as $t\,f(t)$ is a convex function on the interval $t > 0$:

Definition 21. Let X and Y be discrete random variables, and let $f(t)$ be a numerical function such that $t\,f(t)$ is convex on $t > 0$. We define worst-case information *with respect to* f as follows:

$$\begin{aligned} I_w^f(X;Y) \;&:=\; \max_y \, KL^f\,(p_{X|Y=y} \,\|\, p_X), \quad \text{where} \\ KL^f\,(p_1 \,\|\, p_2) \;&:=\; \mathop{\mathbf{E}}_{x \sim p_1} f\big(p_1(x)/p_2(x)\big). \end{aligned}$$

Now we are going to show that knowing worst-case information gives a bound on upward privacy breaches.

Statement 10. *Suppose that revealing* $R(X) = y$ *for some* y *causes an upward* ρ_1*-to-*ρ_2 *privacy breach with respect to property* $Q(X)$*. Then*

$$\rho_2 \cdot f\left(\frac{\rho_2}{\rho_1}\right) \,+\, (1-\rho_2)\cdot f\left(\frac{1-\rho_2}{1-\rho_1}\right) \,\leqslant\, I_w^f(X; R(X)).$$

The proof of this statement is in Section 4.6.3.

As claimed in Statement 10, worst-case information allows us to bound upward privacy breaches. But what to do with downward privacy breaches? It turns out that they are bounded by a measure similar to worst-case information, but in a way "inside-out," or inverse worst-case information. Here is the definition:

Definition 22. Let X and Y be discrete random variables, and let $f(t)$ be a numerical function such that $t\,f(t)$ is convex on $t > 0$. We define *inverse worst-case information* with respect to f as follows:

$$J_w^f(X;Y) \;:=\; \max_y \, KL^f\,(p_X \,\|\, p_{X|Y=y}).$$

Even though Kullback-Leibler distance is called "distance," it is not symmetrical, so usually $J_w^f(X;Y) \neq I_w^f(X;Y)$. The main difference between the two is that the value of $I_w^f(X;Y)$ depends on the behavior of properties likely after Y has been revealed, whereas the value of $J_w^f(X;Y)$ depends on the behavior of properties likely *a priori*. Indeed, in $I_w^f(X;Y)$, the average is taken with respect to distribution $p_{X|Y=y}$, while in $J_w^f(X;Y)$ the average is with respect to p_X. The inverse worst-case information is related to downward breaches in the same way as the straight worst-case information to upward breaches. Let us formulate it in the following statement.

Statement 11. *Suppose that revealing $R(X) = y$ for some y causes a downward ρ_2-to-ρ_1 privacy breach with respect to property $Q(X)$. Then*

$$\rho_2 \cdot f\left(\frac{\rho_2}{\rho_1}\right) + (1-\rho_2) \cdot f\left(\frac{1-\rho_2}{1-\rho_1}\right) \leqslant J_w^f(X; R(X)).$$

The proof of this statement is in Section 4.6.4.

Table 4.2 gives average-case and worst-case information measures (with $f = \log$) for the three randomization operators from Example 2 (see Section 4.2.3). The table shows that R_1 is more sensitive to upward privacy breaches, R_2 is more sensitive to downward privacy breaches, and R_3 has little sensitivity to both of them. The same trend was shown in Table 4.1.

4.6 Proofs

4.6.1 Proof of Statement 6

Proof. Let us show that, for any $\{p[j]\}_{j=0}^m$ that satisfies (4.5), we can construct a distribution of type (4.6) for which the value of $\nu_f(p)$ is at least as large. The idea is to raise and lower some $p[j]$'s while keeping the other $p[i]$'s, $i \neq j$, in constant relation to each other, and so that $\nu_f(p)$ does not decrease in the process.

Table 4.2: The values of average-case and worst-case information measures in Example 2.

Measure	R_1	R_2	R_3
$I(X; R(X))$	1.27	2.32	0.55
$I_w(X; R(X))$	3.90	2.33	0.55
$J_w(X; R(X))$	1.72	∞	0.49

Given $j = 0, 1, \ldots, m$, suppose we increase $p[j]$ by the factor of y and decrease all other $p[i]$'s, for $i \neq j$, by a factor of x (thereby producing distribution $\{\tilde{p}[j]\}_{j=0}^{m}$). Then

$$\sum_{i=0}^{m} \tilde{p}[i] = y \cdot p[j] + x \cdot \sum_{i \neq j} p[i] = 1.$$

We can see that $y = y(x)$ is a linear function of x; therefore, the function

$$\nu_f(\tilde{p}) = \sum_{i=0}^{m} f(i) \cdot \tilde{p}[i] = f(j) \cdot y\, p[j] + \sum_{i \neq j} f(i) \cdot x\, p[i]$$

is also linear with respect to x. We have $\nu_f(\tilde{p}) = \nu_f(p)$ when $x = 1$ and $\nu_f(\tilde{p}) = f(j)$ when $x = 0$ (because then $\tilde{p}[j] = 1$ and $\tilde{p}[i] = 0$ for $i \neq j$). So,

$$\nu_f(\tilde{p}) = f(j) + x \cdot (\nu(p) - f(j)).$$

We say that we *raise* probability $p[j]$ if we change the distribution by decreasing x (starting from $x = 1$) and increasing y until $\tilde{b}[j] = \gamma \cdot \tilde{b}[i]$ for some number $i \neq j$. We can raise only those $p[j]$'s for which $f(j) > \nu(p)$, so that $\nu_f(\tilde{p})$ can only increase. This raising always stops before $\tilde{p}[j]$ reaches 1 or (and) any other probability reaches 0. Analogously, we say that we *lower* $p[j]$ if we increase x and decrease y until $\tilde{b}[j] = \tilde{b}[i]/\gamma$ for some $i \neq j$. We can lower only if $f(j) \leqslant \nu_f(p)$ to prevent $\nu_f(\tilde{p})$ from decreasing. Note that raising and lowering does not affect relations $\tilde{b}[i_1]/\tilde{b}[i_2]$ for $i_1 \neq j \neq i_2$.

We modify the distribution in two steps. First, we lower $p[0]$ and raise $p[m]$. We have $b[m] = \gamma \cdot b[0]$: indeed, when we lowered $p[0]$, $b[0]$ became the smallest of all $b[j]$'s, so it will set the limit to raising $p[m]$. Then, we repeat the following process: choose $p[j]$ that can be lowered or raised, and lower it or raise. Clearly, neither $p[0]$ nor $p[m]$ will ever be chosen since they limit each other and since always $f(0) < \nu(p) < f(m)$. A lowered $b[j]$ becomes equal to $b[0]$, a raised one becomes equal to $b[m]$. Any probability can be lowered only once and will never be raised again because once $f(j) \leqslant \nu(p)$, it stays this way. Any probability can be raised only once and then possibly lowered. After no more than $2(m-1)$ raisings and lowerings, there is nothing to change. For $j_* = \max\{j \mid f(j) \leqslant \nu(p)\}$, the form (4.6) is attained. $\square$

4.6.2 Proof of Statement 9

Proof. Suppose, w.l.o.g., that columns M_1, M_2, ..., M_{d-1} are not linearly independent. Then there is a nonzero linear combination that equals a zero vector:

$$x_1 M_1 + x_2 M_2 + \ldots + x_{d-1} M_{d-1} = \vec{\mathbf{0}}.$$

Extend x to an n-dimensional vector by setting coordinates x_d, ..., x_n to zeros. We get a nonzero vector from the matrix's kernel, which has less than d nonzero coordinates — a contradiction.

Consider any $d-1$ coordinates in $M^T \xi$; w.l.o.g., assume they are the first $d-1$ coordinates. Since the first $d-1$ rows of M^T are linearly independent, they form a $(d-1) \times k$-submatrix M' of rank $d-1$. For any $(d-1)$-dimensional vector v, equation $M'\xi = v$ has the same number 2^{k-d+1} of solutions. When every vector ξ is equally likely, every vector $v = M'\xi$ is therefore also equally likely, that is, its coordinates behave independently. $\square$

4.6.3 Proof of Statement 10

We first prove two simple lemmas and a corollary.

Lemma 1. *If function* $t\,f(t)$ *is convex (or strictly convex) on the interval* $t > 0$*, then so is function* $f(1/t)$.

Proof. Let $0 < t_1 < t_2$ be two different numbers, and let $0 < \lambda < 1$. For strict convexity, we need

$$\lambda\, f\left(\frac{1}{t_1}\right) + (1-\lambda)\, f\left(\frac{1}{t_2}\right) \;>\; f\left(\frac{1}{\lambda t_1 + (1-\lambda) t_2}\right); \tag{4.9}$$

for non-strict convexity, just replace "$>$" with "$\geqslant$." Denote

$$t \;=\; \lambda t_1 + (1-\lambda) t_2, \quad \alpha \;=\; \frac{\lambda t_1}{t};$$

it is clear that $0 < \alpha < 1$. Then, by strict convexity of $t\,f(t)$ we have

$$\alpha\,\frac{1}{t_1} f\left(\frac{1}{t_1}\right) + (1-\alpha)\,\frac{1}{t_2} f\left(\frac{1}{t_2}\right) \;>\; \left(\alpha\frac{1}{t_1} + (1-\alpha)\frac{1}{t_2}\right) f\left(\alpha\frac{1}{t_1} + (1-\alpha)\frac{1}{t_2}\right).$$

Substitution of the definition of α gives

$$\frac{\lambda}{t}\, f\left(\frac{1}{t_1}\right) + (1-\lambda)\,\frac{1}{t}\, f\left(\frac{1}{t_2}\right) \;>\; \left(\lambda\frac{1}{t} + (1-\lambda)\frac{1}{t}\right) f\left(\lambda\frac{1}{t} + (1-\lambda)\frac{1}{t}\right),$$

which is equivalent to (4.9). $\square$

Lemma 2. *Let* X*,* Y*, and* Z *be discrete random variables such that* Z *is independent from* Y *given* X*, and let* $t\,f(t)$ *be convex on* $t > 0$*. Then, for all possible* y*,*

$$KL^f\left(p_{Z|Y=y} \parallel p_Z\right) \;\leqslant\; KL^f\left(p_{X|Y=y} \parallel p_X\right) \quad \text{and}$$
$$KL^f\left(p_Z \parallel p_{Z|Y=y}\right) \;\leqslant\; KL^f\left(p_X \parallel p_{X|Y=y}\right).$$

Proof. Let us prove the first and then the second inequality using the definition of KL^f. We shall use Jensen's inequality $\mathbf{E}\, g(\tau) \geqslant g(\mathbf{E}\, \tau)$ with respect to function $g(t) = f(1/t)$, which is convex on $t > 0$ by Lemma 1.

$$\begin{aligned}
&KL^f\left(p_{X|Y=y} \parallel p_X\right) = \mathop{\mathbf{E}}_{x\sim X|Y=y} f\left(\frac{\mathbf{P}\left[X=x \mid Y=y\right]}{\mathbf{P}\left[X=x\right]}\right)\\
&= \mathop{\mathbf{E}}_{z\sim Z|Y=y} \mathop{\mathbf{E}}_{x\sim X|\substack{Z=z\\Y=y}} f\left(1 \middle/ \frac{\mathbf{P}\left[X=x\right]}{\mathbf{P}\left[X=x \mid Y=y\right]}\right) \geqslant\\
&\geqslant \mathop{\mathbf{E}}_{z\sim Z|Y=y} f\left(1 \middle/ \left(\mathop{\mathbf{E}}_{x\sim X|\substack{Z=z\\Y=y}} \frac{\mathbf{P}\left[X=x\right]}{\mathbf{P}\left[X=x \mid Y=y\right]}\right)\right);
\end{aligned}$$

Using the independence of Z from Y given X, we transform the internal expectation to the desired fraction:

$$\begin{aligned}
&\mathop{\mathbf{E}}_{x\sim X|\substack{Z=z\\Y=y}} \frac{\mathbf{P}\left[X=x\right]}{\mathbf{P}\left[X=x \mid Y=y\right]} =\\
&= \mathop{\mathbf{E}}_{x\sim X} \frac{\mathbf{P}\left[X=x \mid Z=z, Y=y\right]}{\mathbf{P}\left[X=x \mid Y=y\right]}\\
&= \mathop{\mathbf{E}}_{x\sim X} \frac{\mathbf{P}\left[Z=z \mid X=x, Y=y\right]}{\mathbf{P}\left[Z=z \mid Y=y\right]}\\
&= \mathop{\mathbf{E}}_{x\sim X} \frac{\mathbf{P}\left[Z=z \mid X=x\right]}{\mathbf{P}\left[Z=z \mid Y=y\right]} = \frac{\mathbf{P}\left[Z=z\right]}{\mathbf{P}\left[Z=z \mid Y=y\right]}.
\end{aligned}$$

The first inequality is thus proven. The second inequality is very analogous:

$$\begin{aligned}
&KL^f\left(p_X \parallel p_{X|Y=y}\right) = \mathop{\mathbf{E}}_{x\sim X} f\left(\frac{\mathbf{P}\left[X=x\right]}{\mathbf{P}\left[X=x \mid Y=y\right]}\right)\\
&= \mathop{\mathbf{E}}_{z\sim Z} \mathop{\mathbf{E}}_{x\sim X|Z=z} f\left(1 \middle/ \frac{\mathbf{P}\left[X=x \mid Y=y\right]}{\mathbf{P}\left[X=x\right]}\right) \geqslant\\
&\geqslant \mathop{\mathbf{E}}_{z\sim Z} f\left(1 \middle/ \left(\mathop{\mathbf{E}}_{x\sim X|Z=z} \frac{\mathbf{P}\left[X=x \mid Y=y\right]}{\mathbf{P}\left[X=x\right]}\right)\right);
\end{aligned}$$

$$\begin{aligned}
&\mathop{\mathbf{E}}_{x\sim X|Z=z} \frac{\mathbf{P}\left[X=x \mid Y=y\right]}{\mathbf{P}\left[X=x\right]} =\\
&= \mathop{\mathbf{E}}_{x\sim X|Y=y} \frac{\mathbf{P}\left[X=x \,|\, Z=z\right]}{\mathbf{P}\left[X=x\right]} = \mathop{\mathbf{E}}_{x\sim X|Y=y} \frac{\mathbf{P}\left[Z=z \,|\, X=x\right]}{\mathbf{P}\left[Z=z\right]}\\
&= \mathop{\mathbf{E}}_{x\sim X|Y=y} \frac{\mathbf{P}\left[Z=z \,|\, X=x, Y=y\right]}{\mathbf{P}\left[Z=z\right]} = \frac{\mathbf{P}\left[Z=z \,|\, Y=y\right]}{\mathbf{P}\left[Z=z\right]}.
\end{aligned}$$

Both inequalities are now proven. □

Corollary 1. *Under the conditions in Lemma 2, we have*

$$I_w^f(Z;Y) \leqslant I_w^f(X;Y).$$

Proof. Follows immediately from the first inequality of Lemma 2: if for every number in one set there is at least as large number in the other set, then the maximal number of the first set $\leqslant$ the maximal number of the other. □

Proof of Statement 10. Now we have all the tools to prove the bound on upward privacy breaches.

Proof. Let us denote $Y = R(X)$ and

$$P_1 = \mathbf{P}[Q(X)], \quad P_2 = \mathbf{P}[Q(X) \mid Y = y].$$

By Definition 16, we have $P_1 \leqslant \rho_1 < \rho_2 \leqslant P_2$. Let us define α, q_1, and q_2 as follows:

$$\begin{aligned} q_1 &= \rho_2 + \alpha(1 - P_2), \quad q_2 = \rho_1 - \alpha P_1, \\ \alpha &= \frac{\rho_2 - \rho_1}{P_2 - P_1}. \end{aligned}$$

It is clear that $0 < \alpha \leqslant 1$, and therefore

$$\begin{aligned} 0 \leqslant \rho_2 \leqslant q_1 \leqslant 1 - (P_2 - \rho_2) \leqslant 1, \\ 0 \leqslant \rho_1 - P_1 \leqslant q_2 \leqslant \rho_1 \leqslant 1; \end{aligned}$$

so, q_1 and q_2 can serve as probabilities. Let us employ them, then. Define a Boolean random variable Z that depends on X as follows:

1. If $Q(X)$, then Z says "true" with probability q_1;

2. If $\neg Q(X)$, then Z says "true" with probability q_2.

Of course, Z is independent from Y given X, so Corollary 1 is applicable:

$$KL^f\left(p_{Z|Y=y} \parallel p_Z\right) \leqslant I_w^f(Z;Y) \leqslant I_w^f(X;Y).$$

It remains to check that this inequality is exactly what we are proving. Indeed, denote $I = I_w^f(X;Y)$ and "open up" the definition of KL^f:

$$\mathbf{P}\left[Z \mid Y = y\right] \cdot f\left(\frac{\mathbf{P}\left[Z \mid Y = y\right]}{\mathbf{P}\left[Z\right]}\right) + \mathbf{P}\left[\neg Z \mid Y = y\right] \cdot f\left(\frac{\mathbf{P}\left[\neg Z \mid Y = y\right]}{\mathbf{P}\left[\neg Z\right]}\right) \leqslant I.$$

Now compute the prior and posterior probabilities of Z:

$$\begin{aligned}
&\mathbf{P}\left[Z\right] = q_1 \cdot P_1 + q_2 \cdot (1 - P_1) = \\
&= P_1\left(\rho_2 + \alpha(1 - P_2)\right) + (1 - P_1)(\rho_1 - \alpha P_1) \\
&= \rho_1 + P_1\left(\rho_2 - \rho_1\right) - \alpha\, P_1\left(P_2 - P_1\right) \\
&= \rho_1 + P_1\left(\rho_2 - \rho_1\right) - P_1\left(\rho_2 - \rho_1\right) = \rho_1;
\end{aligned}$$

analogously,

$$\begin{aligned}
&\mathbf{P}\left[Z \mid Y = y\right] = q_1 \cdot P_2 + q_2 \cdot (1 - P_2) = \\
&= P_2\left(\rho_2 + \alpha(1 - P_2)\right) + (1 - P_2)(\rho_1 - \alpha P_1) \\
&= \rho_1 + P_2\left(\rho_2 - \rho_1\right) + \alpha\,(1 - P_2)(P_2 - P_1) \\
&= \rho_1 + P_2\left(\rho_2 - \rho_1\right) + (1 - P_2)(\rho_2 - \rho_1) = \rho_2.
\end{aligned}$$

The statement is proven. $\square$

4.6.4 Proof of Statement 11

Let us start with another corollary of Lemma 2.

Corollary 2. *Under the conditions in Lemma 2, we have*

$$J_w^f(Z;Y) \leqslant J_w^f(X;Y).$$

Proof. Follows from the second inequality of Lemma 2 in the same way as Corollary 1 from the first. □

Proof of Statement 11. We are now ready to prove Statement 11.

Proof. The proof is almost analogous to that of Statement 10. We only have to change places between prior and posterior distributions. Namely, we define

$$\begin{aligned}
P_2 &= \mathbf{P}[Q(X)], \quad P_1 = \mathbf{P}[Q(X) \mid Y = y],\\
q_1 &= \rho_2 + \alpha(1 - P_2), \quad q_2 = \rho_1 - \alpha P_1,\\
\alpha &= \frac{\rho_2 - \rho_1}{P_2 - P_1}.
\end{aligned}$$

Again, by Definition 16 we have $P_1 \leqslant \rho_1 < \rho_2 \leqslant P_2$; we define Z exactly like before, and in the end get

$$\mathbf{P}[Z] \cdot f\left(\frac{\mathbf{P}[Z]}{\mathbf{P}[Z \mid Y = y]}\right) + \mathbf{P}[\neg Z] \cdot f\left(\frac{\mathbf{P}[\neg Z]}{\mathbf{P}[\neg Z \mid Y = y]}\right) \leqslant J_w^f(X;Y),$$

where

$$\mathbf{P}[Z \mid Y = y] = \rho_1, \quad \mathbf{P}[Z] = \rho_2.$$

The statement is proven. □

Chapter 5

Information Sharing Across Private Databases

5.1 Introduction

The material in this Chapter has been previously published in [3]. This work is done in collaboration with Rakesh Agrawal and Ramakrishnan Srikant at IBM Almaden Research Center.

Literature on information integration across databases tacitly assumes that the data in each database can be revealed to the other databases. However, there is an increasing need for sharing information across autonomous entities in such a way that no information apart from the answer to the query is revealed (Section 1.1) In this chapter, we formalize the notion of *minimal information sharing* across private databases, and develop protocols for intersection, equijoin, intersection size, and equijoin size. We also show how new applications can be built using the described protocols.

Intuitively, given a database query spanning multiple private databases, we wish to compute the answer to the query without revealing any additional information apart from the query result. However, we will sometimes relax this constraint to allow some minimal additional information to be revealed. Let us give two prototypical applications to make the above paradigm concrete.

5.1.1 Motivating Applications

Application 1: Selective Document Sharing Enterprise R is shopping for technology and wishes to find out if enterprise S has some intellectual property it might want to license. However, R would not like to reveal its complete technology shopping list, nor would S like to reveal all its unpublished intellectual property. Rather, they would like to first find the specific technologies for which there is a match, and then reveal information only about those technologies. This problem can be abstracted as follows.

We have two databases D_R and D_S, where each database contains a set of documents. The documents have been preprocessed to only include the most significant words, using some measure such as term frequency times inverse document frequency [144]. We wish to find all pairs of similar documents $d_R \in D_R$ and $d_S \in D_S$, without revealing the other documents. In database terminology, we want to compute the join of D_R and D_S using the join predicate $f(|d_R \cap d_S|, |d_R|, |d_S|) > \tau$, for some similarity function f and threshold τ. The function f could be $|d_R \cap d_S|/(|d_R|+|d_S|)$, for instance.

Many applications map to this abstraction. For example, two government agencies may want to share documents, but only on a need-to-know basis. They would like to find similar documents contained in their repositories in order to initiate their exchange.

Application 2: Medical Research Imagine a future where many people have their DNA sequenced. A medical researcher wants to validate a hypothesis connecting a DNA sequence D with a reaction to drug G. People who have taken the drug are partitioned into four groups, based on whether or not they had an adverse reaction and whether or not their DNA contained the specific sequence; the researcher needs the number of people in each group. DNA sequences and medical histories are stored in databases in autonomous enterprises. Due to privacy concerns, the enterprises do not wish to provide any information about an individual's DNA sequence or medical history, but still wish to help with the research.

Assume that the table T_R(person_id, pattern) stores whether a person's DNA contains pattern D and T_S(person_id, drug, reaction) captures whether a person took drug G and whether the person had an adverse reaction. T_R and T_S belong to two different enterprises. The researcher wants to get the answer to the following query.

`select` pattern, reaction, count(*)

`from` T_R, T_S

`where` T_R.person_id = T_S.person_id `and` T_S.drug = "true"

`group by` T_R.pattern, T_S.reaction

We want the property that the researcher should get to know the counts and nothing else, and the enterprises should not learn any new information about any individual.

5.1.2 Current Techniques

Some existing techniques that one might use for building the above applications are as follows:

- **Trusted Third Party**: The main parties give the data to a "trusted" third party and have the third party do the computation [15, 98]. However, the third party has to be *completely* trusted, both with respect to intent and competence against security breaches. The level of trust required is too high for this solution to be acceptable.
- **Secure Multi-Party Computation**: Given two parties with inputs x and y respectively, the goal of secure multi-party computation is to compute a function $f(x, y)$ such that the two parties learn only $f(x, y)$, and nothing else [83]. Yao [171] showed that any multi-party computation can be solved by building a boolean circuit for solving the problem, and simulating that circuit. We show in Section 5.6 that our specialized algorithms are substantially faster than using a circuit, and in particular, the communication costs for circuits make them impractical for our problems.

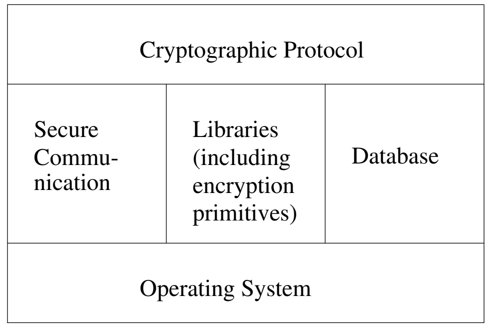

Figure 5.1: System Components

5.1.3 Minimal Information Sharing

Security Model

We develop our solutions in a setting in which there is no third party [83]. The main parties directly execute a protocol, which is designed to guarantee that they do not learn any more than they would have learnt had they given the data to a trusted third party and got back the answer.

We assume *honest-but-curious* behavior [83]. The parties follow the protocol properly with the exception that they may keep a record of all the intermediate computations and received messages, and analyze the messages to try to learn additional information. This behavior is also referred to as *semi-honest* or *passive* behavior.

Figure 5.1 shows the different components required for building a system for information integration with minimal sharing. Our focus will be on the cryptographic protocol. We assume the use of standard libraries or packages for secure communication and encryption primitives.

Problem Statement

We now formally state the problem we study in this chapter.

Problem Statement (Ideal) Let there be two parties R (receiver) and S (sender) with databases D_R and D_S respectively. Given a database query Q spanning the tables in D_R and D_S, compute the answer to Q and return it to R without revealing any additional information to either party. $\square$

Problem Statement (Minimal Sharing) Let there be two parties R and S with databases D_R and D_S respectively. Given a database query Q spanning the tables in D_R and D_S, and some categories of information I, compute the answer to Q and return it to R without revealing any additional information to either party except for information contained in I. $\square$

For example, if the query Q is a join $T_R \bowtie T_S$ over two tables T_R and T_S, the additional information I might be the number of records in each table: $|T_R|$ and $|T_S|$. Note that whatever R can infer from knowing the answer to the query Q and the additional information I is fair game. For instance, if the query Q is an intersection $V_S \cap V_R$ between two sets V_S and V_R, then for all $v \in (V_R - (V_S \cap V_R))$, R knows that these values were not in V_S.

We assume that the query Q is revealed to both parties. One can think of other applications where the format of Q is revealed, but not the parameters of Q (e.g., in private information retrieval, discussed in Section 2.2).

Operations

We are going to focus on four operations: intersection, equijoin, intersection size, and equijoin size.

Let S have a database table T_S, and R have a table T_R, with both tables having a specific attribute A in their schemas. The attribute A takes its values from a given set V. Let V_S be the set of values (without duplicates) that occur in $T_S.A$, and let V_R be the set of values occurring in $T_R.A$. For each $v \in V_S$, let $\text{ext}(v)$ be all records in T_S where $T_S.A = v$, i.e., $\text{ext}(v)$ is the *extra information* in T_S pertaining to v. We show how to compute three kinds of queries over T_S and T_R:

- *Intersection:* Party R learns the set $V_S \cap V_R$, the value $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 5.2).
- *Equijoin:* Party R learns $V_S \cap V_R$, $\text{ext}(v)$ for all $v \in V_S \cap V_R$, $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 5.3).
- *Intersection Size:* Party R learns the values of $|V_S \cap V_R|$, $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 5.4).

Thus in the terminology of our problem statement above, the query Q for the three problems corresponds to $V_S \cap V_R$, $T_S \bowtie T_R$ (with $\text{ext}(v)$ used to compute the join), and $|V_S \cap V_R|$ respectively. In all three cases, the additional information I consists of $|V_R|$ and $|V_S|$.

We also extend the intersection size protocol to obtain an *equijoin size* protocol that computes $|T_S \bowtie T_R|$ (Section 5.4.2). However, R learns $|V_S|$, the distribution of duplicates in $T_S.A$, and based on the distribution of duplicates, some subset of information in $V_S \cap V_R$. S learns $|V_R|$ and the distribution of duplicates in $T_R.A$.

5.1.4 Limitations

Multiple Queries While we provide guarantees on how much the parties learn from a single query, our techniques do not address the question of what the parties might learn by combining the results of multiple queries. The first line of defence against this problem is the scrutiny of the queries by the parties. In addition, query restriction techniques from the statistical database literature [1, 153] can also help. These techniques include restricting the size of query results [46, 73], controlling the overlap among successive queries [49], and keeping audit trails of all answered queries to detect possible compromises [33].

Schema Discovery and Heterogeneity We do not address the question of how to find which database contains which tables and what the attribute names are; we assume that the database schemas are known. We also do not address issues of schema heterogeneity. See [62, 94] and references therein for some approaches to these problems.

5.2 Intersection Protocol

5.2.1 A simple, but incorrect, protocol

A straightforward idea for computing the intersection $V_S \cap V_R$ would be to use one-way hash functions [131]. Here is a simple protocol that *appears* to work:

1. Both S and R apply hash function h to their sets, yielding $X_S = h(V_S) = \{h(v) \mid v \in V_S\}$ and
   $X_R = h(V_R) = \{h(v) \mid v \in V_R\}$.
2. S sends its hashed set X_S to R.
3. R sets aside all $v \in V_R$ for which $h(v) \in X_S$; these values form the set $V_S \cap V_R$.

Unfortunately, R can learn a lot more about V_S (with honest-but-curious behavior). For any arbitrary value $v \in V - (V_S \cap V_R)$, R can simply compute $h(v)$ and check whether $h(v) \in X_S$ to determine whether or not $v \in V_S$. In fact, if the domain V is small, R can exhaustively go over all possible values and completely learn V_S.

The intersection protocol we propose next fixes the deficiencies of this protocol.

5.2.2 Building Blocks

We first describe two building blocks used in the proposed protocols.

Commutative Encryption

Our definition of commutative encryption below is similar to the constructions used in [23, 47, 61, 148] and others. Informally, a commutative encryption is a pair of encryption functions f and g such that $f(g(v)) = g(f(v))$. Thus by using the combination $f(g(v))$ to encrypt v, we can ensure that R cannot compute the encryption of a value without the help of S. In addition, even though the encryption is a combination of two functions, each party can apply their function first and still get the same result.

Definition 23 (Indistinguishability). Let $\Omega_k \subseteq \{0,1\}^k$ be a finite domain of k-bit numbers. Let $\mathcal{D}_1 = \mathcal{D}_1(\Omega_k)$ and $\mathcal{D}_2 = \mathcal{D}_2(\Omega_k)$ be distributions over Ω_k. Let $\mathcal{A}_k(x)$ be an algorithm that, given $x \in \Omega_k$, returns either true or false. We define distribution $\mathcal{D}_1$ of random variable $x \in \Omega_k$ to be *computationally indistinguishable* from distribution $\mathcal{D}_2$ if for any family of polynomial-step (w.r.t. k) algorithms $\mathcal{A}_k(x)$, any polynomial $p(k)$, and all sufficiently large k

$$\Pr[\mathcal{A}_k(x) \mid x \sim \mathcal{D}_1] \,-\, \Pr[\mathcal{A}_k(x) \mid x \sim \mathcal{D}_2] \,<\, \frac{1}{p(k)}$$

where $x \sim \mathcal{D}$ denotes that x is distributed according to $\mathcal{D}$, and $\Pr[\mathcal{A}_k(x)]$ is the probability that $\mathcal{A}_k(x)$ returns true.

Throughout this chapter, we will use "indistinguishable" as shorthand for "computationally indistinguishable".

Definition 24 (Commutative Encryption). A *commutative encryption* $\mathcal{F}$ is a computable (in polynomial time) function $f : \operatorname{Key}\mathcal{F} \times \operatorname{Dom}\mathcal{F} \to \operatorname{Dom}\mathcal{F}$, defined on finite computable domains, that satisfies all properties listed below.[1] We denote $f_e(x) \equiv f(e, x)$, and use "$\in_r$" to mean "is chosen uniformly at random from".

1. Commutativity: For all $e, e' \in \operatorname{Key}\mathcal{F}$ we have
$$f_e \circ f_{e'} = f_{e'} \circ f_e \ .$$
2. Each $f_e : \operatorname{Dom}\mathcal{F} \to \operatorname{Dom}\mathcal{F}$ is a bijection.
3. The inverse f_e^{-1} is also computable in polynomial time given e.
4. The distribution of $\langle x, f_e(x), y, f_e(y)\rangle$ is indistinguishable from the distribution of $\langle x, f_e(x), y, z\rangle$, where $x, y, z \in_r \operatorname{Dom}\mathcal{F}$ and $e \in_r \operatorname{Key}\mathcal{F}$.

Informally, Property 1 says that when we compositely encrypt with two different keys, the result is the same irrespective of the order of encryption. Property 2 says that two different values will never have the same encrypted value. Property 3 says that given an encrypted value $f_e(x)$ and the encryption key e, we can find x in polynomial time.[2] Property 4 says that given a value x and its encryption $f_e(x)$ (but not the key e), for a new value y, we cannot distinguish between $f_e(y)$ and a random value z in polynomial time. Thus we can neither encrypt y nor decrypt $f_e(y)$ in polynomial time. Note that this property holds only if x is a random value from $\operatorname{Dom}\mathcal{F}$, i.e., the adversary does not control the choice of x.

Example 1 Let $\operatorname{Dom}\mathcal{F}$ be all quadratic residues modulo p, where p is a "safe" prime number, i.e., both p and $q = (p-1)/2$ are primes. Let $\operatorname{Key}\mathcal{F}$ be $\{1, 2, \ldots, q-1\}$. Then, assuming the Decisional Diffie-Hellman hypothesis (DDH) [24], the power function

[1] Formally, we have to talk about a *sequence* of domains rather than one domain $\operatorname{Dom}\mathcal{F}$, in order for polynomial computability to make sense.

[2] We only need this property for the join protocol, not for the intersection protocol.

$$f_e(x) \equiv x^e \bmod p$$

is a commutative encryption:

- The powers commute:

  $(x^d \bmod p)^e \bmod p = x^{de} \bmod p = (x^e \bmod p)^d \bmod p.$
- Each of the powers f_e is a bijection with its inverse being $f_e^{-1} = f_{e^{-1} \bmod q}$.
- DDH claims that for any generating ($\neq 1$) element $g \in \operatorname{Dom} \mathcal{F}$ the distribution of $\langle g^a, g^b, g^{ab} \rangle$ is indistinguishable from the distribution of $\langle g^a, g^b, g^c \rangle$, where $a, b, c \in_r \operatorname{Key} \mathcal{F}$. A 3-tuple $\langle g^a, g^b, z \rangle$ from the DDH can be reduced to our 4-tuple $\langle x, x^e, y, z \rangle$ by taking $d \in_r \operatorname{Key} \mathcal{F}$ and making tuple $\langle g^d, (g^a)^d, g^b, z \rangle$. Now a plays the role of e, g^d of x, and g^b of y; we test whether $z = (g^b)^a$ or is random. Thus, given DDH, $\langle x, x^e, y, y^e \rangle$ and $\langle x, x^e, y, z \rangle$ are also indistinguishable.

Hash Function

Besides a commutative encryption $\mathcal{F}$, we need a hash function to encode the values $v \in V$ into $x \in \operatorname{Dom} \mathcal{F}$. The hashes of values should not collide and should "look random," i.e., there should be no dependency between them that could help encrypt or decrypt one hashed value given the encryption of another. Since we apply commutative encryption to the hashed values $h(v)$ instead of v, the input for the encryption function will appear random, and we will be able to use Property 4 of commutative encryption to prove that our protocols are secure.

In the proofs of our security statements we shall rely on the standard *random oracle model* [22, 74, 158]. We assume that our hash function $h : V \to \operatorname{Dom} \mathcal{F}$ is *ideal*, which means that $h(v)$ can be considered computed by a random oracle: every time $h(v)$ is evaluated for a new $v \in V$, an independent random $x \in_r \operatorname{Dom} \mathcal{F}$ is chosen for $x = h(v)$.

We assume also that $|\operatorname{Dom}\mathcal{F}|$ is so large compared to $|V_S \cup V_R|$ that the probability of a collision is exponentially small. Let $N = |\operatorname{Dom}\mathcal{F}|$; in the random oracle model, the probability that n hash values have at least one collision equals [158]:

$$\Pr[\text{collision}] \;=\; 1 - \prod_{i=1}^{n-1} \frac{N-i}{N} \;\approx\; 1 - \exp\left(\frac{-n\,(n-1)}{2N}\right)$$

With 1024-bit hash values, half of which are quadratic residues, we have $N \approx 2^{1024}/2 \approx 10^{307}$, and for $n = 1$ million

$$\Pr[\text{collision}] \;\approx\; 1 - \exp\left(-\frac{10^{12}}{10^{307}}\right) \;\approx\; \frac{10^{12}}{10^{307}} \;=\; 10^{-295}.$$

For real-life hash functions, a collision within V_S or V_R can be detected by the server at the start of each protocol by sorting the hashes. If there is a collision between $v \in V_S$ and $v' \in V_R$, it will cause inclusion of v' into the join (or intersection) by R and the disclosure to R of S's records containing v.[3]

5.2.3 Intersection Protocol

Our proposed intersection protocol is as follows.

1. Both S and R apply hash function h to their sets:

   $X_S = h(V_S)$ and $X_R = h(V_R)$.

   Each party randomly chooses a secret key:

   $e_S \in_r \operatorname{Key}\mathcal{F}$ for S and $e_R \in_r \operatorname{Key}\mathcal{F}$ for R.

2. Both parties encrypt their hashed sets:

   $Y_S = f_{e_S}(X_S) = f_{e_S}(h(V_S))$ and

   $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.

[3]For the join protocol (Section 5.3), R can check whether there was a collision between $v \in V_S$ and $v' \in V_R$ by having S include the value v in $\operatorname{ext}(v)$.

3. R sends to S its encrypted set $Y_R = f_{e_R}(h(V_R))$, reordered lexicographically.[4]

4. (a) S ships to R its set $Y_S = f_{e_S}(h(V_S))$, reordered lexicographically.

   (b) S encrypts each $y \in Y_R$ with S's key e_S and sends back to R pairs $\langle y, f_{e_S}(y)\rangle = \langle f_{e_R}(h(v)), f_{e_S}(f_{e_R}(h(v)))\rangle$.

5. R encrypts each $y \in Y_S$ with e_R, obtaining

   $Z_S = f_{e_R}(f_{e_S}(h(V_S)))$.

   Also, from pairs $\langle f_{e_R}(h(v)), f_{e_S}(f_{e_R}(h(v)))\rangle$ obtained in Step 4(b) for $v \in V_R$, it creates pairs $\langle v, f_{e_S}(f_{e_R}(h(v)))\rangle$ by replacing $f_{e_R}(h(v))$ with the corresponding v.

6. R selects all $v \in V_R$ for which $(f_{e_S}(f_{e_R}(h(v))) \in Z_S$; these values form the set $V_S \cap V_R$.

5.2.4 Proofs of Correctness and Security

Statement 12. *Assuming there are no hash collisions, S learns the size $|V_R|$ and R learns the size $|V_S|$ and the set $V_S \cap V_R$.*

Proof. By definition, f_{e_S} and f_{e_R} commute and are bijective. Assuming that hash function h has no collisions on $V_S \cup V_R$,

$$v \in V_S \cap V_R \;\text{ iff }\; v \in V_R \text{ and } (f_{e_S} \circ f_{e_R})(h(v)) \in Z_S,$$

which means that R does recover the correct set $V_S \cap V_R$. Both parties also learn the sizes $|V_R|$ and $|V_S|$, since $|V_R| = |Y_R|$ and $|V_S| = |Y_S|$. □

Next we prove that, assuming the parties follow the protocol correctly, they learn

[4] If we did not reorder and instead sent the values in the same order as the values in V_R, significant additional information could be revealed.

nothing else about the other's sets. We first show that even given

$$\begin{pmatrix} x_1 & \ldots & x_m \\ f_e(x_1) & \ldots & f_e(x_m) \end{pmatrix} \quad \text{and} \quad x_{m+1},$$

there is no polynomial-time algorithm that can determine whether or not a value u is in fact $f_e(x_{m+1})$.

Lemma 3. *For polynomial m, the distribution of the $2 \times m$-tuple*

$$\begin{pmatrix} x_1 & \ldots & x_{m-1} & x_m \\ f_e(x_1) & \ldots & f_e(x_{m-1}) & f_e(x_m) \end{pmatrix}$$

is indistinguishable from the distribution of the tuple

$$\begin{pmatrix} x_1 & \ldots & x_{m-1} & x_m \\ f_e(x_1) & \ldots & f_e(x_{m-1}) & z_m \end{pmatrix},$$

where $\forall i:\ x_i \in_r \operatorname{Dom} \mathcal{F}$, $z_m \in_r \operatorname{Dom} \mathcal{F}$, and $e \in_r \operatorname{Key} \mathcal{F}$.

Proof. Let us denote the distribution of the upper tuple by $\mathcal{D}_m$, and the distribution of the lower tuple by $\mathcal{D}_{m-1}$. If $\mathcal{D}_m$ and $\mathcal{D}_{m-1}$ are distinguishable by some polynomial algorithm $\mathcal{A}$, then $\langle x, f_e(x), y, f_e(y)\rangle$ and $\langle x, f_e(x), y, z\rangle$ from Property 4 of commutative encryption are also distinguishable by the following algorithm that takes $\langle x, f_e(x), y, u\rangle$ as argument:

1. For $i = 1 \ldots m-1$, let $x_i = f_{e_i}(x)$ and $z_i = f_{e_i}(f_e(x))$, where $e_i \in_r \operatorname{Key} \mathcal{F}$;
2. Let $x_m = y$ and $z_m = u$;
3. Submit tuple
$$\begin{pmatrix} x_1 & \ldots & x_m \\ z_1 & \ldots & z_m \end{pmatrix}$$
to algorithm $\mathcal{A}$ and output whatever it outputs.

For $i = 1 \ldots m-1$, we have

$$z_i = f_{e_i}(f_e(x)) = f_e(f_{e_i}(x)) = f_e(x_i),$$

and all x_i are indistinguishable from uniformly random (from Property 4 of commutative encryption). Therefore the distribution of the tuple given to $\mathcal{A}$ is indistinguishable from $\mathcal{D}_m$ when $\langle x, f_e(x), y, u\rangle$ is distributed as $\langle x, f_e(x), y, f_e(y)\rangle$, and from $\mathcal{D}_{m-1}$ when $\langle x, f_e(x), y, u\rangle$ is distributed as $\langle x, f_e(x), y, z\rangle$. So the assumption that $\mathcal{D}_m$ and $\mathcal{D}_{m-1}$ are distinguishable leads to the contradiction that Property 4 does not hold. $\square$

Lemma 4. *For polynomial m and n, the distribution of the $2\times n$-tuple*

$$\begin{pmatrix} x_1 & \ldots & x_m & x_{m+1} & \ldots & x_n \\ f_e(x_1) & \ldots & f_e(x_m) & f_e(x_{m+1}) & \ldots & f_e(x_n) \end{pmatrix}$$

is indistinguishable from the distribution of the tuple

$$\begin{pmatrix} x_1 & \ldots & x_m & x_{m+1} & \ldots & x_n \\ f_e(x_1) & \ldots & f_e(x_m) & z_{m+1} & \ldots & z_n \end{pmatrix},$$

where $0 \leqslant m \leqslant n$, $\forall i:\ x_i, z_i \in_r \operatorname{Dom} \mathcal{F}$, and $e \in_r \operatorname{Key} \mathcal{F}$.

Proof. Let us denote by $\mathcal{D}^n_m$ the distribution of the lower tuple; the upper tuple's distribution is thus $\mathcal{D}^n_n$.

From Lemma 3, for all $j = m{+}1 \ldots n$, the distributions $\mathcal{D}^n_j$ and $\mathcal{D}^n_{j-1}$ are indistinguishable. (The first j columns of $\mathcal{D}^n_j$ are identical to $\mathcal{D}_j$ of Lemma 3, the first j columns of $\mathcal{D}^n_{j-1}$ are identical to $\mathcal{D}_{j-1}$ of Lemma 3, and the last $n-j$ columns of $\mathcal{D}^n_{j-1}$ and $\mathcal{D}^n_j$ are just uniformly random numbers.)

Since $\mathcal{D}^n_{j-1}$ and $\mathcal{D}^n_j$ are indistinguishable for $\forall j = m{+}1 \ldots n$, and because n is bounded by a polynomial, $\mathcal{D}^n_n$ is also indistinguishable from any $\mathcal{D}^n_m$ (where $0 \leqslant m \leqslant n$). Let $\mathcal{A}_k$ be an algorithm that pretends to distinguish $\mathcal{D}^n_n$ from $\mathcal{D}^n_m$, and returns true or false. Now

$$\begin{aligned} &\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_n^n] \,-\, \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_m^n] \qquad (5.1)\\ &= \sum_{j=m+1}^{n} \Big(\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_j^n] \,-\, \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_{j-1}^n] \Big) \end{aligned}$$

Here k is the number of bits in the tuples. Consider any polynomial $p(k)$; we want to prove that $\exists k_0\ \forall k \geqslant k_0$ the difference (5.1) is bounded by $1/p(k)$. Let $p'(k) = n\,p(k)$, which is also a polynomial. We have $\forall j = m{+}1 \ldots n\ \exists k_j\ \forall k \geqslant k_j$ the j-th difference in the telescoping sum is bounded by $1/p'(k)$. Now set $k_0 = \max_j k_j$, and we are done:

$$\sum_{j=m+1}^{n} \Big(\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_j^n] \,-\, \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_{j-1}^n] \Big) \;<\; \\ <\; \sum_{j=m+1}^{n} \frac{1}{p'(k)} \;<\; \frac{n}{n\,p(k)} \;=\; \frac{1}{p(k)}.$$

Computational indistinguishability between $\mathcal{D}_n^n$ and $\mathcal{D}_m^n$ is proven. □

Statement 13. *The intersection protocol is secure if both parties are semi-honest. In the end, S learns only the size $|V_R|$, and R learns only the size $|V_S|$ and the intersection $V_S \cap V_R$.*

Proof. We use a standard proof methodology from multi-party secure computation [83]. If, for any V_S and V_R, the distribution of the S's view of the protocol (the information S gets from R) cannot be distinguished from a simulation of this view that uses only V_S and $|V_R|$, then clearly S cannot learn anything from the inputs it gets from R except for $|V_R|$. Note that the simulation only uses the knowledge S is supposed to have at the end of the protocol, while the distinguisher also uses the inputs of R (i.e., V_R), but not R's secret keys (i.e., e_R). It is important that the distinguisher be unable to distinguish between the simulation and the real view even given R's inputs: this precludes the kind of attack that broke the protocol given in Section 5.2.1.

The simulator for S (that simulates what S receives from R) is easy to construct. At Step 3 of the protocol, the only step where S receives anything, the simulator generates $|V_R|$ random values $z_i \in_r \mathrm{Dom}\,\mathcal{F}$ and orders them lexicographically. In the real protocol, these values equal $f_{e_R}(h(v))$ for $v \in V_R$. Assuming that, for all $v \in V_R$, the hashes $h(v)$ are distributed uniformly at random (random oracle model), by Lemma 4 the distributions

$$\underbrace{\begin{pmatrix} x_1 & \dots & x_m \\ f_{e_R}(x_1) & \dots & f_{e_R}(x_m) \end{pmatrix}}_{x_i=h(v_i),\, v_i\in V_R} \quad \text{and} \quad \underbrace{\begin{pmatrix} x_1 & \dots & x_m \\ z_1 & \dots & z_m \end{pmatrix}}_{x_i=h(v_i),\, v_i\in V_R},$$

where $\forall i:\ z_i \in_r \mathrm{Dom}\,\mathcal{F}$, are indistinguishable. Therefore the real and simulated views for S are also indistinguishable.

The simulator for R (that simulates what R gets from S) will use V_R, $V_S \cap V_R$ and $|V_S|$; it also knows the hash function h. However, it does not have $V_S - V_R$. The simulator chooses a key $\tilde{e}_S \in_r \mathrm{Key}\,\mathcal{F}$. In Step 4(a), the simulation creates Y_S as follows:

- First, for values $v_i \in V_S \cap V_R$, the simulation adds $f_{\tilde{e}_S}(h(v_i))$ to Y_S.
- Next, the simulation adds $|V_S - V_R|$ random values $z_i \in_r \mathrm{Dom}\,\mathcal{F}$ to Y_S.

In Step 4(b), the simulation uses the key $\tilde{e}_S$ to encrypt each $y \in Y_R$.

Since e_S (real view) and $\tilde{e}_S$ (simulation) are both chosen at random, their distributions are identical. According to Lemma 4, one cannot distinguish between the distribution of

$$\left(\underbrace{\begin{matrix} x_1 & \dots & x_m \\ f_{\tilde{e}_S}(x_1) & \dots & f_{\tilde{e}_S}(x_m) \end{matrix}}_{x_i=h(v_i),\, v_i\in V_R} \;\; \underbrace{\begin{matrix} x_{m+1} & \dots & x_n \\ f_{\tilde{e}_S}(x_{m+1}) & \dots & f_{\tilde{e}_S}(x_n) \end{matrix}}_{x_i=h(v_i),\, v_i\in V_S-V_R}\right)$$

and the distribution of

$$\left(\underbrace{\begin{matrix} x_1 & \dots & x_m \\ f_{\tilde{e}_S}(x_1) & \dots & f_{\tilde{e}_S}(x_m) \end{matrix}}_{x_i=h(v_i),\, v_i\in V_R} \;\; \underbrace{\begin{matrix} x_{m+1} & \dots & x_n \\ z_{m+1} & \dots & z_n \end{matrix}}_{x_i=h(v_i),\, v_i\in V_S-V_R}\right).$$

The real view corresponds to the upper matrix, and the simulated view to the lower matrix. Therefore the real view and the simulated view are also indistinguishable, and the statement is proven. $\square$

5.3 Equijoin Protocol

We now extend the intersection protocol so that, in addition to $V_S \cap V_R$, R learns some extra information $\mathrm{ext}(v)$ from S for values $v \in V_S \cap V_R$, but does not learn $\mathrm{ext}(v)$ for for $v \in V_S - V_R$. To compute the join $T_S \bowtie T_R$ on attribute A, we have $\mathrm{ext}(v)$ contain all the records of S's table where $T_S.A = v$, i.e., $\mathrm{ext}(v)$ contains the information about the other attributes in T_S needed for the join.

5.3.1 Idea Behind Protocol

A simple, but incorrect, solution would be to encrypt the extra information $\mathrm{ext}(v)$ using $h(v)$ as the encryption key. Since, in our intersection protocol, $h(v)$ could not be discovered by R except for $v \in V_R$ (and similarly for S), one might think that this protocol would be secure. While it is true that $h(v)$ cannot be discovered from Y_R or Y_S, $h(v)$ can be discovered from the encryption of ext(v). For any arbitrary value v, R can compute $h(v)$ and try decrypting all the $\mathrm{ext}(v)$ using $h(v)$ to learn whether or not $v \in V_S$. In fact, if the domain is small, R can exhaustively go over all possible values and completely learn both V_S and $\mathrm{ext}(v)$ for $v \in V_S$.

Rather then encrypt the extra information with $h(v)$, we will encrypt it with a key $\kappa(v) = f_{e'_S}(h(v))$, where e'_S is a second secret key of S. The problem now is to allow R to learn $\kappa(v)$ for $v \in V_R$ without revealing V_R to S. We do this as follows: R sends $f_{e_R}(h(v))$ to S, and S sends back $f_{e'_S}(f_{e_R}(h(v)))$ to R. R can now apply $f_{e_R}^{-1}$ to the latter to get

$$f_{e_R}^{-1}(f_{e'_S}(f_{e_R}(h(v)))) \;=\; f_{e_R}^{-1}(f_{e_R}(f_{e'_S}(h(v)))) \;=\; f_{e'_S}(h(v)).$$

Note that R only gets $f_{e'_S}(h(v))$ for $v \in V_R$, not for $v \in V_S - V_R$.

5.3.2 Encryption Function K

We now formally define the encryption function $K(\kappa, \mathrm{ext}(v))$ that encrypts $\mathrm{ext}(v)$ using the key $\kappa(v)$. K is defined to be a function

$$K : \; \mathrm{Dom}\,\mathcal{F} \;\times\; V_{\mathrm{ext}} \;\rightarrow\; C_{\mathrm{ext}}$$

with two properties:

1. Each function $K_\kappa(x) \equiv K(\kappa, x)$ can be efficiently inverted (decrypted) given κ;
2. "Perfect Secrecy" [149]: For any $\mathrm{ext}(v)$, the value of $K_\kappa(\mathrm{ext}(v))$ is indistinguishable from a fixed (independent of $\mathrm{ext}(v)$) distribution $\mathcal{D}_{\mathrm{ext}}$ over C_{ext} when $\kappa \in_r \mathrm{Dom}\,\mathcal{F}$.

Example 2 Let $\mathcal{F}$ be the power function over quadratic residues modulo a safe prime, as in Example 1. If the extra information $\mathrm{ext}(v)$ can also be encoded as a quadratic residue (i.e., $V_{\mathrm{ext}} = \mathrm{Dom}\,\mathcal{F}$), the encryption $K_\kappa(\mathrm{ext}(v))$ can be just a multiplication operation:

$$K_\kappa(\mathrm{ext}(v)) \;=\; \kappa \cdot \mathrm{ext}(v).$$

The multiplication can be easily reversed given κ, and if κ is uniformly random then $\kappa \cdot \mathrm{ext}(v)$ is also uniformly random (independently of $\mathrm{ext}(v)$).

5.3.3 Equijoin Protocol

Let V_S be the set of values (without duplicates) that occur in $T_S.A$, and let V_R be the set of values that occur in $T_R.A$. For each $v \in V_S$, let $\text{ext}(v)$ be all records in T_S where $T_S.A = v$.

1. Both S and R apply hash function h to their sets:

   $X_S = h(V_S)$ and $X_R = h(V_R)$.

   R chooses its secret key $e_R \in_r \text{Key}\,\mathcal{F}$, and S chooses two secret keys: $e_S, e'_S \in_r \text{Key}\,\mathcal{F}$.

2. R encrypts its hashed set: $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.

3. R sends to S its encrypted set Y_R, reordered lexicographically.

4. S encrypts each $y \in Y_R$ with both key e_S and key e'_S, and sends back to R 3-tuples

   $\langle y,\, f_{e_S}(y),\, f_{e'_S}(y)\rangle$

   $= \langle f_{e_R}(h(v)),\, f_{e_S}(f_{e_R}(h(v))),\, f_{e'_S}(f_{e_R}(h(v)))\rangle.$

5. For each $v \in V_S$, S does the following:

   (a) Encrypts the hash $h(v)$ with e_S, obtaining $f_{e_S}(h(v))$.

   (b) Generates the key for extra information using e'_S:

   $\kappa(v) = f_{e'_S}(h(v))$.

   (c) Encrypts the extra information:

   $c(v) = K(\kappa(v), \text{ext}(v))$.

   (d) Forms a pair $\langle f_{e_S}(h(v)),\, c(v)\rangle$

   $= \langle f_{e_S}(h(v)),\, K(f_{e'_S}(h(v)), \text{ext}(v))\rangle.$

   The pairs are then shipped to R in lexicographical order.

6. R applies $f_{e_R}^{-1}$ to all entries in the 3-tuples received at Step 4, obtaining $\langle h(v), f_{e_S}(h(v)), f_{e'_S}(h(v))\rangle$ for all $v \in V_R$.

7. R sets aside all pairs $\langle f_{e_S}(h(v)), K(f_{e'_S}(h(v)), \mathrm{ext}(v))\rangle$ received at Step 5 whose first entry occurs as a second entry in a 3-tuple $\langle h(v), f_{e_S}(h(v)), f_{e'_S}(h(v))\rangle$ from Step 6. Using the third entry $f_{e'_S}(h(v)) = \kappa(v)$ as the key, R decrypts $K(f_{e'_S}(h(v)), \mathrm{ext}(v))$ and gets $\mathrm{ext}(v)$. The corresponding v's form the intersection $V_S \cap V_R$.

8. R uses $\mathrm{ext}(v)$ for $v \in V_S \cap V_R$ to compute $T_S \bowtie T_R$.

5.3.4 Proofs of Correctness and Security

Statement 14. *Assuming there are no hash collisions, S learns $|V_R|$, and R learns $|V_S|$, $V_S \cap V_R$, and $\mathrm{ext}(v)$ for all $v \in V_S \cap V_R$.*

Proof. This protocol is an extension of the intersection protocol, so it allows R to determine $V_S \cap V_R$ correctly. Since R learns the keys $\kappa(v)$ for values in the intersection, R also gets $\mathrm{ext}(v)$ for $v \in V_S \cap V_R$. $\square$

Next we prove that R and S do not learn anything besides the above. We first extend Lemma 4 as follows.

Lemma 5. *For polynomial n, the distributions of the following two $3\times n$-tuples*

$$\begin{pmatrix} x_1 & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_n) \\ f_{e'}(x_1) & \dots & f_{e'}(x_n) \end{pmatrix} \quad \textit{and} \quad \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \\ z_1 & \dots & z_n \end{pmatrix},$$

are computationally indistinguishable, where $\forall i: \ x_i, y_i, z_i \in_r \mathrm{Dom}\,\mathcal{F}$, and $e, e' \in_r \mathrm{Key}\,\mathcal{F}$.

Proof. Let us denote the left distribution by $\mathcal{D}_1$, the right distribution by $\mathcal{D}_2$, and the following "intermediate" distribution by $\mathcal{D}_3$:

$$\begin{pmatrix} x_1 & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_n) \\ z_1 & \dots & z_n \end{pmatrix}$$

The first and third line in the tuples for $\mathcal{D}_1$ and $\mathcal{D}_3$ are distributed like $\mathcal{D}_n^n$ and $\mathcal{D}_0^n$ (from Lemma 4) respectively. The second line in both $\mathcal{D}_1$ and $\mathcal{D}_3$ can be obtained from the first line by applying f_e with random key e. Therefore, since $\mathcal{D}_n^n$ and $\mathcal{D}_0^n$ are indistinguishable by Lemma 4, distributions $\mathcal{D}_1$ and $\mathcal{D}_3$ are also indistinguishable.

Analogously, the first and second lines in $\mathcal{D}_3$ and $\mathcal{D}_2$ are distributed like $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ respectively. The third line in both $\mathcal{D}_3$ and $\mathcal{D}_2$ can be obtained by using random numbers for the z_i's. Therefore, by Lemma 4, $\mathcal{D}_3$ and $\mathcal{D}_2$ are also indistinguishable.

Finally, since both $\mathcal{D}_1$ and $\mathcal{D}_2$ are indistinguishable from $\mathcal{D}_3$, they themselves are indistinguishable. $\square$

The following lemma will be used in the proof for the security of the join protocol to show that the real and simulated views for R are indistinguishable. $\mathcal{D}'_1$ corresponds to the real view (for R), while $\mathcal{D}'_2$ corresponds to the simulated view. The first t columns correspond to $V_S - (V_S \cap V_R)$, the next $m - t$ columns to $V_S \cap V_R$, and the last $n - m$ columns to $V_R - (V_S \cap V_R)$.

Lemma 6. *For polynomial m, t, and n, and any $c_i \in V_{\text{ext}}$, the two distributions $\mathcal{D}'_1$ and $\mathcal{D}'_2$ of the $4 \times n$-tuple*

$$\begin{pmatrix} x_1 & \dots & x_t & x_{t+1} & \dots & x_m & x_{m+1} & \dots & x_n \\ y_1 & \dots & y_t & y_{t+1} & \dots & y_m & y_{m+1} & \dots & y_n \\ & & & z_{t+1} & \dots & z_m & z_{m+1} & \dots & z_n \\ \xi_1 & \dots & \xi_t & \xi_{t+1} & \dots & \xi_m & & & \end{pmatrix}$$

such that

- *For* $\mathcal{D}'_1$, $\forall i\,:\, x_i \in_r \operatorname{Dom}\mathcal{F}$, $y_i = f_e(x_i)$, $z_i = f_{e'}(x_i)$, *and* $\xi_i = K(f_{e'}(x_i), c_i)$ *where* $e, e' \in_r \operatorname{Key}\mathcal{F}$*;*
- *For* $\mathcal{D}'_2$, $\forall i\,:\, x_i, y_i, z_i \in_r \operatorname{Dom}\mathcal{F}$*, and*
  - $i = 1\ldots t\,:\ \xi_i$ *is independent random with distribution* $\mathcal{D}_{\text{ext}}$*,*
  - $i = t{+}1\ldots m\,:\ \xi_i = K(z_i, c_i)$

are computationally indistinguishable. (In both $\mathcal{D}'_1$ *and* $\mathcal{D}'_2$*, the positions corresponding to* $z_1 \ldots z_t$ *and* $\xi_{m+1} \ldots \xi_n$ *are blank.)*

Proof. Denote by $\mathcal{D}'_3$ the following "intermediate" distribution:

$$\forall i\,:\ x_i, y_i, z_i \in_r \operatorname{Dom}\mathcal{F} \text{ and } \xi_i = K(z_i, c_i).$$

Note that the z_i for $i = 1\ldots t$ are not included in the tuple, even though they are used to generate $K(z_i, c_i)$.

The only difference between the two distributions $\mathcal{D}'_2$ and $\mathcal{D}'_3$ is that, for $i = 1\ldots t$, we replace ξ_i distributed as $\mathcal{D}_{\text{ext}}$ with $K(z_i, c_i)$ where $z_i \in_r \operatorname{Dom}\mathcal{F}$; the rest of the matrix is independent and stays the same. Since z_i is not a part of the matrix for $i = 1\ldots t$, by Property 2 of encryption $K(\kappa, c)$, distributions $\mathcal{D}'_2$ and $\mathcal{D}'_3$ are indistinguishable.

Next we use Lemma 5 to show that distributions $\mathcal{D}'_1$ and $\mathcal{D}'_3$ are also indistinguishable. We define function $Q(M)$ that takes a $3\times n$ matrix M (from Lemma 5) and generates a $4\times n$ matrix M' as follows:

1. The first 3 rows of M' are the same as the first 3 rows of M, except that the values corresponding to $z_1, \ldots, z_t$ in M' are left blank.
2. The fourth row of M' is generated by taking $\xi_i = K(z_i, c_i)$ where z_i is the corresponding value of the third row of M.

If M is distributed like $\mathcal{D}_1$ of Lemma 5, $Q(M)$ corresponds to $\mathcal{D}'_1$. If M is distributed like $\mathcal{D}_2$, $Q(M)$ corresponds to $\mathcal{D}'_3$. Since by Lemma 5, $\mathcal{D}_1$ and $\mathcal{D}_2$ are indistinguishable, and $Q(M)$ is computable in polynomial time, $\mathcal{D}'_1$ and $\mathcal{D}'_3$ are also indistinguishable.

Finally, since both $\mathcal{D}'_1$ and $\mathcal{D}'_2$ are indistinguishable from $\mathcal{D}'_3$, they themselves are indistinguishable. □

Statement 15. *The join protocol is secure if both parties are semi-honest. At the end of the protocol, S learns only $|V_R|$; R learns only $|V_S|$, $V_S \cap V_R$, and $\operatorname{ext}(v)$ for all $v \in V_S \cap V_R$.*

Proof. As in the proof of Statement 13, we will construct simulators of each party's view of the protocol, such that each simulator is given only what the party is supposed to learn, and such that the distribution of the real view is indistinguishable from the distribution of the simulated view.

The simulator for S is identical to that in Statement 13, since S gets exactly the same input from R as in the intersection protocol. Hence the proof from Statement 13 directly applies.

The simulator for R (that simulates what R receives from S) can use h, e_R, V_R, $V_S \cap V_R$, $\operatorname{ext}(v)$ for $v \in V_S \cap V_R$, and $|V_S|$. Let

$$\begin{aligned} V_S &= \{v_1, \ldots, v_t, v_{t+1}, \ldots, v_m\} \text{ and} \\ V_R &= \{v_{t+1}, \ldots, v_m, v_{m+1}, \ldots, v_n\}. \end{aligned}$$

So $t = |V_S - V_R|$, $m = |V_S|$, and $n = |V_S \cup V_R|$. Note that the simulator does not know the values in $V_S - V_R$.

In Step 4, the simulator generates n random numbers $y_i \in_r \operatorname{Dom} \mathcal{F}$, $i = 1 \ldots n$ as the simulated values for $f_{e_S}(h(v_i))$, and an additional n random numbers $z_i \in_r \operatorname{Dom} \mathcal{F}$

as the simulated values for $f_{e'_S}(h(v_i))$. The simulation then uses key e_R to create

$$\langle f_{e_R}(h(v_i)),\ f_{e_R}(y_i),\ f_{e_R}(z_i)\rangle$$

for $i = t{+}1 \ldots m$. These triplets are ordered lexicographically and comprise the simulated view for Step 4.

In Step 5, the simulator creates the pairs as follows:

- For values $v_{t+1}, \ldots, v_m$ from $V_S \cap V_R$, the simulator encrypts $\mathrm{ext}(v_i)$ as $\xi_i = K(z_i, \mathrm{ext}(v_i))$; then it forms pairs $\langle y_i, \xi_i\rangle$;
- For $i = 1 \ldots t$, the simulator creates $|V_S - V_R|$ additional pairs $\langle y_i, \xi_i\rangle$ where ξ_i have distribution $\mathcal{D}_{\text{ext}}$ over C_{ext}, i.e., y_i and ξ_i are random values from their respective domains.

These pairs are sorted lexicographically and comprise the simulated view for Step 5.

Setting $x_i = h(v_i)$, the real view corresponds to distribution $\mathcal{D}'_1$ of the matrix in Lemma 6, while the simulation corresponds to distribution $\mathcal{D}'_2$ of the matrix. The only difference is that some variables appear in the view encrypted by f_{e_R}, which makes the view a efficiently-computable function of the matrix. Since these $\mathcal{D}'_1$ and $\mathcal{D}'_2$ are indistinguishable, the simulation is also indistinguishable from the real view. $\square$

5.4 Intersection and Join Size Protocols

5.4.1 Intersection Size

We now show how the intersection protocol can be modified, such that R only learns the intersection size, but not which values in V_R were present in V_S. (Simply applying the intersection protocol would reveal the set $V_R \cap V_S$, in addition to the intersection size.) Recall that in Step 4 of the intersection protocol, S sends back to R the values of

$y \in Y_R$ together with their encryptions made by S. These encryptions are paired with the unencrypted y's so that R can match the encryptions with R's values. If instead S sends back to R only the lexicographically reordered encryptions of the y's and not the y's themselves, R can no longer do the matching.

Intersection Size Protocol

We now present the protocol for intersection size. (Steps 1 through 3 are the same as in the intersection protocol.)

1. Both S and R apply hash function h to their sets:

   $X_S = h(V_S)$ and $X_R = h(V_R)$.

   Each party randomly chooses a secret key:

   $e_S \in_r \text{Key}\,\mathcal{F}$ for S and $e_R \in_r \text{Key}\,\mathcal{F}$ for R.

2. Both parties encrypt their hashed sets:

   $Y_S = f_{e_S}(X_S) = f_{e_S}(h(V_S))$ and

   $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.

3. R sends to S its encrypted set $Y_R = f_{e_R}(h(V_R))$, reordered lexicographically.

4. (a) S ships to R its set $Y_S = f_{e_S}(h(V_S))$, reordered lexicographically.

   (b) S encrypts each $y \in Y_R$ with S's key e_S and sends back to R the set $Z_R = f_{e_S}(Y_R) = f_{e_S}(f_{e_R}(h(V_R)))$, reordered lexicographically.

5. R encrypts each $y \in Y_S$ with e_R, obtaining

   $Z_S = f_{e_R}(f_{e_S}(h(V_S)))$.

6. Finally, R computes intersection size $|Z_S \cap Z_R|$, which equals $|V_S \cap V_R|$.

Proofs of Correctness and Security

Statement 16. *Assuming there are no hash collisions, S learns the size $|V_R|$ and R learns the size $|V_S|$ and the size $|V_S \cap V_R|$.*

Proof. The proof is very similar to that for Statement 12. Since f_{e_S} and f_{e_R} commute and are bijective, assuming that hash function h has no collisions on $V_S \cup V_R$,

$$|V_S \cap V_R| \;=\; f_{e_R}(f_{e_S}(h(V_S))) \cap f_{e_S}(f_{e_R}(h(V_R))).$$

Therefore R recovers the correct size $|V_S \cap V_R|$. $\square$

Statement 17. *The intersection size protocol is secure if both parties are semi-honest. At the end of the protocol, S learns only the size $|V_R|$, and R learns only the sizes $|V_S|$ and $|V_S \cap V_R|$.*

Proof. We use the same methodology as in the proofs of Statement 13 and 15.

The simulator for S's view of the intersection size protocol is identical to that in Statement 13, since S gets exactly the same input from R as in the intersection protocol. Hence the proof from Statement 13 directly applies.

The simulator for R's view of the protocol is allowed to use V_R, the hash function h, e_R, and the numbers $|V_S \cap V_R|$ and $|V_S|$; however, it has neither $V_S - V_R$ nor $V_S \cap V_R$. Let

$$V_S \;=\; \{v_1,\, \ldots,\, v_t,\, v_{t+1},\, \ldots,\, v_m\} \text{ and}$$
$$V_R \;=\; \{v_{t+1},\, \ldots,\, v_m,\, v_{m+1},\, \ldots,\, v_n\}.$$

So $t = |V_S - V_R|$, $m = |V_S|$, and $n = |V_S \cup V_R|$.

The simulator generates n random numbers $y_1, \ldots, y_n \in_r \operatorname{Dom} \mathcal{F}$ which play the role of $f_{e_S}(h(v))$ for all $v \in V_S \cup V_R$. The key e_S is not simulated, and no decision is made about which y_i stands for which $f_{e_S}(h(v))$. In Step 4(a), the simulation creates Y_S as

$$Y_S \;=\; \{y_1, \ldots, y_m\}.$$

In Step 4(b), the simulation generates Z_R by taking set $\{y_{t+1}, \ldots, y_n\}$ and encoding it with f_{e_R}:

$$Z_R \;=\; \{f_{e_R}(y_{t+1}), \ldots, f_{e_R}(y_n)\}.$$

We now show that the distribution of R's real view in the protocol is computationally indistinguishable from the distribution of R's simulated view.

According to Lemma 4, the distributions $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ of the following matrix M:

$$\begin{pmatrix} x_1 & \ldots & x_n \\ y_1 & \ldots & y_n \end{pmatrix}$$

where

- $\mathcal{D}_0^n:\; \forall i: x_i, y_i \in_r \operatorname{Dom} \mathcal{F}$;
- $\mathcal{D}_n^n:\; \forall i: x_i \in_r \operatorname{Dom} \mathcal{F},\, y_i = f_{e_S}(x_i),\, e_S \in_r \operatorname{Key} \mathcal{F}$;

are indistinguishable. Given $x_i = h(v_i)$, consider the following function $Q(M)$:

$$Q(M) \;=\; \langle h,\, e_R,\, Y_S,\, Z_R \rangle,$$

where

$$\begin{aligned} h \;&:=\; \text{a function on } V_S \cup V_R \text{ s.t. } \forall i:\; h(v_i) = x_i; \\ e_R \;&:=\; \text{a random key}; \\ Y_S \;&:=\; \{y_1, \ldots, y_m\}; \\ Z_R \;&:=\; \{f_{e_R}(y_{t+1}), \ldots, f_{e_R}(y_n)\}. \end{aligned}$$

If M is distributed according to $\mathcal{D}_0^n$, then $Q(M)$ corresponds to the simulated view of server R. If M's distribution is $\mathcal{D}_n^n$, then

$$\begin{aligned} y_i &= f_{e_S}(x_i) = f_{e_S}(h(v_i)), \\ f_{e_R}(y_i) &= f_{e_R}(f_{e_S}(x_i)) = f_{e_S}(\, f_{e_R}(h(v_i))\,), \end{aligned}$$

and $Q(M)$ is distributed like the real view of R. Since from Lemma 4, $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ are indistinguishable, and Q is computable in polynomial time, the simulated view $Q(\mathcal{D}_0^n)$ and the real view $Q(\mathcal{D}_n^n)$ are also indistinguishable. □

5.4.2 Equijoin Size

To evaluate equijoin size, we follow the intersection size protocol, except that we allow V_R and V_S to be multi-sets, i.e., contain duplicates, and then compute the join size instead of the intersection size in Step 6. However, R can now use the number of duplicates of a given value to partially match values in Y_R with their corresponding encryptions in Z_R. We now characterize exactly what R and S learn in this protocol (besides $|V_R|$, $|V_S|$ and $|V_R \bowtie V_S|$).

To start with, R learns the distribution of duplicates in V_S, and S learns the distribution of duplicates in V_R. To characterize what else R learns, let us partition the values in V_R based on the number of duplicates, i.e., in a partition $V_R(d)$, each $v \in V_R(d)$ has d duplicates. Then, for each partition, R learns $|V_R(d) \cap V_S(d')|$ for each partition $V_S(d')$ of V_S. Thus if all values have the same number of duplicates (e.g., no duplicates as in our intersection protocol), R only learns $|V_R \cap V_S|$. At the other extreme, if no two values have the same number of duplicates, R will learn $V_R \cap V_S$.

5.5 Cost Analysis

5.5.1 Protocols

Let

- each encrypted codeword (in $\mathrm{Dom}\,\mathcal{F}$) be k bits long,
- C_h denote the cost of evaluating the hash function,
- C_e denote the cost of encryption/decryption by $\mathcal{F}$ (e.g., exponentiation "$x^y \bmod p$" over k-bit integers),
- C_K denote the cost of encryption/decryption by K (e.g., encoding/decoding as a quadratic residue and multiplication), and
- $n \log n \cdot C_s$ be the cost of sorting a set of n encryptions.

We assume the obvious optimizations when computing the computation and communication costs. For example, in the join protocol, we assume that the protocol does not decrypt y to $h(v)$ in Step 6, but uses order preservation for matching. Also, in all the protocols, S does not retransmit the y's back but just preserves the original order.

Computation The computation costs are:

- *Intersection:* $(C_h + 2C_e)(|V_S| + |V_R|) + 2C_s|V_S| \log |V_S|$
  $+ 3C_s|V_R| \log |V_R|$
- *Join:* $C_h(|V_S| + |V_R|) + 2C_e|V_S| + 5C_e|V_R|$
  $+ C_K(|V_S| + |V_S \cap V_R|) + 2C_s|V_S| \log |V_S|$
  $+ 3C_s|V_R| \log |V_R|$

We can assume $C_e \gg C_h$, $C_e \gg C_K$, and $nC_e \gg n \log n \cdot C_s$, so these formulae can be approximated by

- *Intersection:* $2C_e(|V_S| + |V_R|)$

- *Join:* $2C_e|V_S| + 5C_e|V_R|$

Communication The communication cost is:

- *Intersection:* $(|V_S| + 2|V_R|) \cdot k$ bits
- *Join:* $(|V_S| + 3|V_R|) \cdot k + |V_S| \cdot k'$ bits, where k' is the size of the encrypted $\mathrm{ext}(v)$.

Both the intersection size and join size protocols have the same computation and communication complexity as the intersection protocol.

5.5.2 Applications

We now estimate the execution times for the applications in Section 5.1.1.

For the cost of C_e (i.e., cost of $x^y \bmod p$), we use the times from [127]: 0.02s for 1024-bit numbers on a Pentium III (in 2001). This corresponds to around 2×10^5 exponentiations per hour. We assume that communication is via a T1 line, with bandwidth of 1.544 Mbits/second (≈ 5 Gbits/hour).

Encrypting the set of values is trivially parallelizable in all three protocols. We assume that we have P processors that we can utilize in parallel: we will use a default value of $P = 10$.

Selective Document Sharing

Recall that we have two databases D_R and D_S, where each database contains a set of documents, and a document consists of a set of significant words. We wish to find all pairs of documents $d_R \in D_R$ and $d_S \in D_S$ such that, for some similarity function f and threshold τ, $f(|d_R \cap d_S|, |d_R|, |d_S|) > \tau$. For example, f could be $|d_R \cap d_S|/(|d_R| + |d_S|)$.

Implementation We call the intersection size protocol for each pair of documents $d_R \in D_R$ and $d_S \in D_S$ to get $|d_R \cap d_S|$, $|d_R|$ and $|d_S|$; we then compute the similarity function f.

For S, in addition to the number of documents $|D_S|$, this protocol also reveals for each document to $d_R \in D_R$, which documents in D_S matched d_R, and the size of $|d_R \cap d_S|$; similar additional information is leaked for R.

Cost Analysis For a given pair of documents d_R and d_S, the computation time is $(|d_R| + |d_S|) \cdot 2C_e$, and the data transferred is $(|d_R| + 2|d_S|) \cdot k$ bits. Thus the total cost is:

- Computation: $|D_R| \cdot |D_S| \cdot (|d_R| + |d_S|) \cdot 2C_e$.
- Communication: $|D_R| \cdot |D_S| \cdot (|d_R| + 2|d_S|) \cdot k$.

Assume $|D_R| \approx 10$ documents, $D_S \approx 100$ documents, and $|d_R| = |d_S| \approx 1000$ words. Then the computation time will be $4 \times 10^6 \cdot C_e/P \approx 2$ hour. The data transferred will be $3 \times 10^6 \cdot k \approx 3$ Gbits ≈ 35 minutes.

Medical Research

Recall that we wish to get the answer to the query

> `select` pattern, reaction, count(*)
>
> `from` T_R, T_S
>
> `where` T_R.id = T_S.id and T_S.drug = true
>
> `group by` T_R.pattern, T_S.reaction

where T_R and T_S are tables in two different enterprises.

Implementation Figure 5.2 shows the implementation algorithm. We use a slightly modified version of the intersection size protocol where Z_R and Z_S are sent to T, the researcher, instead of to S and R. Note that whenever we have, say, $(V_R - V_R')$ inside IntersectionSize, the set difference is computed locally, and the result is the input to the

V_R := ids in T_R.

V'_R := subset of V_R that match the DNA sequence.

V_S := ids in T_S that took the drug.

V'_S := subset of V_S with adverse reaction.

T gets IntersectionSize(V'_R, V'_S).

T gets IntersectionSize($V'_R, (V_S - V'_S)$).

T gets IntersectionSize($(V_R - V'_R), V'_S$).

T gets IntersectionSize($(V_R - V'_R), (V_S - V'_S)$).

Figure 5.2: Algorithm for Medical Research Application

protocol.

Cost Analysis The combined cost of the four intersections is $2(|V_R| + |V_S|) \cdot 2C_e$, and the data transferred is $2(|V_R| + |V_S|) \cdot 2k$ bits. If $|V_R|$ = $|V_S|$ = 1 million, the total computation time will be $8 \times 10^6 \cdot C_e/P \approx$ 4 hours. The total communication time will be $8 \times 10^6 \cdot k \approx 8$ Gbits $\approx$ 1.5 hours.

5.6 Circuit-Based Protocols

For comparison, we estimate the computation and communication cost of intersection and join protocols obtained using Yao's cryptographic protocol for oblivious circuit evaluation [171]. Let V_S and V_R contain w-bit values. Consider a function $f(\vec{x}, \vec{y})$ that takes vectors $\vec{x}$ and $\vec{y}$ (of size $w \cdot |V_S|$ and $w \cdot |V_R|$ respectively) as inputs and returns a vector $\vec{z}$ (of size $|V_R|$) that shows which of R's values also belong to V_S. This function can be represented by a circuit of boolean gates. We shall use the semi-honest variant of the protocol. According to it, S hardwires its input $\vec{x}$ into the circuit and supplements each possible encrypted bit value at each circuit wire with its own random key (used for decrypting the next gate's output and its key). The protocol has two major steps:

Coding R's input: For each bit of $\vec{y}$, R engages with S in a 1-out-of-2 oblivious transfer protocol [127, 134] and gets the corresponding supplemental keys;

Computing the circuit: For each gate, R receives a table from S and, using the keys for the gate's inputs, computes the output and its key. In the process, R applies a pseudorandom function twice per each output wire.

To get $f(\vec{x}, \vec{y})$, R gets the tables that allow it to decrypt the wires with the output of the circuit.

5.6.1 Cost Analysis

Let the keys be k_0 bits long, and C_r be the cost of pseduorandom function evaluation. We assume that $w = 32$ (recall that w is the size in bits of the input values), $k_0 = 64$, and $|V_R| = |V_S| = n$.

Coding the Input

Let C_{ot} be the computation cost of each oblivious transfer, and C'_{ot} its communication cost. An efficient protocol for oblivious transfers is given in [127]. For any integer $l > 0$, this paper gives a protocol with amortized cost

$$C_{\text{ot}} = \frac{1}{l} \cdot C_e + \frac{2^l}{l} \cdot C_\times; \quad C'_{\text{ot}} = \frac{2^l}{l} \cdot k_0$$

where $C_\times$ is the cost of multiplication. Assume that $C_e = 1000\, C_\times$; then the best choice with respect to the computation time is $l = 8$, and the costs become

$$C_{\text{ot}} = 0.157\, C_e; \quad C'_{\text{ot}} = 32\, k_0.$$

Cost The computation cost of coding the input is

$$w \cdot |V_R| \cdot C_{\text{ot}} = 32 \times n \times 0.157\, C_e \approx 5\, n\, C_e$$

and the communication cost is

$$w \cdot |V_R| \cdot C'_{\text{ot}} = 32 \times n \times 32\, k_0 \approx 6.5 \times 10^4\, n$$

Evaluating the Circuit

Let $C(w, |V_S|, |V_R|)$ be the total number of gates required for the circuit. We estimate lower bounds on the number of gates required for a brute force algorithm, and a more efficient partitioning algorithm.

Let G_e be the number of gates required to compare two w-bit numbers in the circuit to determine whether they are equal. Let G_l be the number of gates required to determine which number is less than (or equal to) the other.

Brute Force Circuit Consider a circuit that compares every number in V_R with every number in V_S, and then merges the results to output just the numbers in V_R that were equal to at least one number in V_S. The number of gates $C(w, |V_S|, |V_R|)$ in this circuit is greater than

$$|V_R| \cdot |V_S| \cdot G_e.$$

Partitioning Circuit We can assume that each set V_R and V_S is given to the circuit in the form of an ordered array, with all duplicates removed. Instead of comparing all pairs of numbers, we can split these arrays into m intervals (non-interleaving subarrays) of size $|V_R|/m$ and $|V_S|/m$. For ease of exposition, we assume that $|V_R| = |V_S| = n$, and that n is a power of m.

Out of all possible m^2 pairs of subarrays, with one subarray from V_S and the other from V_R, only at most $2m-1$ pairs may have a nonempty intersection; the others are pairs of non-interleaving subarrays. To see this, note that in a pair of interleaving subarrays the beginning of one subarray must be within the interval spanned by the other. There is at most one pair per one such "internal beginning." There are $2m$ subarrays in both V_S and V_R, each having only one beginning; and the smallest beginning is always "wasted," thus limiting the number of interleaving pairs to $2m-1$.

The circuit has to choose the $2m-1$ interleaving pairs of subarrays and then use recursion to compute set intersections within these pairs. To check whether a pair of subarrays interleaves, we need to compare the smallest and largest numbers of these subarrays, thus making 2 comparisons. There are m^2 pairs, so we need $2m^2$ comparisons, and hence $2m^2G_l$ gates. Some more gates are needed to reroute the subarrays and combine the recursive outputs, but we shall ignore them in our estimation.

Please see more details in Section 5.6.3.

Let $f(n)$ be the cost of the circuit. Then

$$\begin{aligned} f(n) &> 2m^2G_l + (2m-1)\cdot f(n/m) \\ f(1) &= G_e \end{aligned}$$

Note that the above formula should be considered a lower bound for the cost of the circuit (using this algorithm); the actual cost will be significantly higher. Let $c = 2m^2G_l$;

then

$$
\begin{aligned}
f(n) &> c + (2m-1)\cdot f(n/m) \\
&> c + (2m-1)\cdot(c + (2m-1)f(n/m^2)) \\
&> c \cdot \sum_{i=0}^{\log_m n-1} (2m-1)^i + (2m-1)^{\log_m n} f(1) \\
&= c \cdot \frac{(2m-1)^{\log_m n}-1}{(2m-1)-1} + (2m-1)^{\log_m n} G_e \\
&= c \cdot \frac{n^{\log_m(2m-1)}-1}{2m-2} + n^{\log_m(2m-1)} G_e \\
&> \left(\frac{c}{2m-2} + G_e\right)\cdot(n^{\log_m(2m-1)}-1)
\end{aligned}
$$

Substituting back the value for c, we get

$$
f(n) > \left(\frac{m^2}{m-1} G_l + G_e\right)\cdot(n^{\log_m(2m-1)} - 1)
$$

Two w-bit numbers can be checked for equality using $2w-1$ binary gates and compared using $5w-3$ gates. Setting $G_l = 5w-3$ and $G_e = 2w-1$ gives

$$
f(n) > \left(\frac{m^2}{m-1}(5w-3) + (2w-1)\right)\cdot(n^{\log_m(2m-1)}-1)
$$

Brute Force vs. Partitioning Circuit Let $|V_R| = |V_S| = n$. (As before w = 32 and k_0 = 64.) Then, for the partitioning circuit, we get the following values for $C(w, |V_R|, |V_S|) = f(n)$ for the optimal value of m:

n	m	$f(n)$
10,000	11	2.3×10^8
1 million	19	7.3×10^{10}
100 million	32	1.9×10^{13}

The brute force circuit does much worse, with $C(w, |V_R|, |V_S|)$ equal to 6.3×10^9, $6.3\times$

10^{13}, and 6.3×10^{17} respectively.

Cost The computation cost of circuit evaluation is at least

$$2C_r \cdot C(w, |V_S|, |V_R|)$$

and the communication cost is at least

$$4k_0 \cdot C(w, |V_S|, |V_R|).$$

Then the computation cost of circuit evaluation is

$$2C_r \cdot C(w, |V_S|, |V_R|)$$

and the communication cost is

$$4k_0 \cdot C(w, |V_S|, |V_R|) = 256 \cdot C(w, |V_S|, |V_R|).$$

We now discuss the number of gates required for two different constructions of the circuit.

The computation cost for the partitioning circuit is $2C_r f(n)$, and the communication cost is $4k_0 f(n) = 256 f(n)$.

5.6.2 Comparison with Our Protocol

Computation We get the following computation costs:

n	Circuit		Our Protocol
	Input (OT)	Evaluation	
10^4	$5\times10^4 C_e$	$4.7\times10^8 C_r$	$4\times10^4 C_e$
10^6	$5\times10^6 C_e$	$1.5\times10^{11} C_r$	$4\times10^6 C_e$
10^8	$5\times10^8 C_e$	$3.8\times10^{13} C_r$	$4\times10^8 C_e$

The cost of just coding the input (i.e., the cost of the oblivious transfers) is slightly higher than the cost of our protocol. Therefore the total cost of the circuit (relative to our protocol) will depend on the ratio of C_e to C_r. While $C_e \gg C_r$, there are 10^4 to 10^5 as many calls to C_r as there are to C_e. Thus our protocol will be substantially faster if $C_r > C_e/10000$, and slightly faster otherwise.

Communication The difference in communication costs is even more dramatic:

n	Circuit		Our Protocol
	Input (OT)	Circuit	
10^4	6.5×10^{10}	6.0×10^{10}	3×10^7
10^6	6.5×10^{12}	1.8×10^{13}	3×10^9
10^8	6.5×10^{14}	4.9×10^{15}	3×10^{11}

Our protocol requires between 10^3 and 10^4 times less communication. For n = 1 million, the communication time for the circuit-based protocol is 144 days (using a T1 line), versus 0.5 hours for our protocol. The communication times make the circuit-based protocols impractical for database-size applications.

5.6.3 Partitioning Circuit: Details

Input: Two arrays, X and Y, that contain n k-bit numbers. All numbers in X are distinct, and the array is sorted so that $1 \leqslant i < j \leqslant n$ implies $X_i < X_j$; the same is true for array Y. We assume $k \ll n \ll 2^k$.

Output: A vector V of n bits such that $V_i = 1$ if and only if the k-bit number Y_i belongs also to array X.

Statement 18. *For any fixed $\varepsilon > 0$, there is a Boolean circuit $C(X, Y)$ that computes vector V on all valid inputs X and Y and has $O(k \cdot n^{1+\varepsilon} \log n)$ binary logical gates.*

More precisely,

$$C(X,Y) \;=\; O(k\cdot 4^{1/\varepsilon})\cdot n^{1+\varepsilon}\cdot(1+\varepsilon\,\log n)$$

Proof. Let $m = \lceil 2^{1/\varepsilon}\rceil$. Split both arrays X and Y into m consecutive intervals of numbers, each interval containing $\lfloor n/m\rfloor$ numbers:

$$X \;=\; I_1^X\cup I_2^X\cup\ldots\cup I_m^X,\quad Y \;=\; I_1^Y\cup I_2^Y\cup\ldots\cup I_m^Y.$$

If n is not divisible by m, make a special simple circuit that compares the last $\nu = n \bmod m$ numbers of X and Y with all numbers in the opposite array. If this circuit compares each number with each number in the trivial way, it needs $2\cdot 2k\nu n < 4kmn$ gates[5] and generates a vector that can be later joined with the rest using another n gates. Thus, it remains to compare the first $\lfloor n/m\rfloor\cdot m$ numbers of X and Y, i. e. numbers within the intervals.

Since the arrays X and Y are sorted, the intervals are also sorted, and for $1\leqslant i < j\leqslant m$ we have $I_i^X < I_j^X$ and $I_i^Y < I_j^Y$, meaning that all numbers in, say, I_i^X are smaller than all numbers in I_j^X. Note that, for any i,j, intersection $I_i^X\cap I_j^Y$ is empty when either $I_i^X < I_j^Y$ or $I_i^X > I_j^Y$. Let $(I_i^X)_s$ be the s-th number of I_i^X; then $I_i^X < I_j^Y$ if and only if $(I_i^X)_{\lfloor n/m\rfloor} < (I_j^Y)_1$.

Let us prove that there are at most $2m-1$ pairs (I_i^X, I_j^Y) such that neither $I_i^X < I_j^Y$ nor $I_i^X > I_j^Y$ (*incomparable pairs*). For any such pair, we have

$$(I_i^X)_1 \leqslant (I_j^Y)_1 \leqslant (I_i^X)_{\lfloor n/m\rfloor} \;\text{ or} \tag{5.2}$$

$$(I_j^Y)_1 \leqslant (I_i^X)_1 \leqslant (I_j^Y)_{\lfloor n/m\rfloor}, \tag{5.3}$$

depending on which interval starts first. In case (5.2) we say that interval I_j^Y has *internal beginning*, in case (5.3) interval I_i^X has internal beginning in the pair. Each interval can

[5]Two k-bit numbers can be checked for equality using $2k-1$ binary gates and for inequality using $5k-3$ gates.

have internal beginning in only one pair, and there are $2m$ intervals, so there are at most $2m$ incomparable pairs. The interval with the smallest first number (either I_1^X or I_1^Y) cannot have internal beginning, unless $(I_1^X)_1 = (I_1^Y)_1$ when both I_1^X and I_1^Y have internal beginning in the same pair. Therefore, in fact we have at most $2m-1$ incomparable pairs. It is easy to see that $2m-1$ is achievable.

We shall build our circuit by recursion. The circuit for $n < m$ is done by comparing all numbers in X with all numbers in Y and needs $2kn^2 < 2km^2$ gates. Now let us assume that $n \geqslant m$ and denote the number of gates in the circuit for n numbers by $f(n)$. By recursion, we assume that we have built the circuits for less than n numbers.

Our circuit contains three sections: preamble, middle part, and conclusion. In the preamble, we compute the $2m-1$ pairs of incomparable intervals (some pairs may be duplicates or nonsense). The middle part contains $2m-1$ recursive modules that compute intersection for each incomparable pair. The conclusion computes the final result.

As we know from above, each incomparable pair of intervals has at least one interval with internal beginning. So, for each interval starting from I_2^X and I_2^Y (the first intervals are treated differently) we find the pair for which this interval has internal beginning and feed the pair into its recursive circuit in the middle part. WLOG., consider the interval I_i^X. For each interval I_j^Y of Y, we check property (5.3) ($10k$ gates) and, if not satisfied, zero the interval ($k\lfloor n/m \rfloor$ gates) before joining it to the middle-part input wires by OR-gates (another $k\lfloor n/m \rfloor$ gates). Overall, the number of gates for all considered intervals is

$$(2m-2) \cdot m \cdot (10k + 2k \left\lfloor \frac{n}{m} \right\rfloor).$$

For the first two intervals, I_1^X and I_1^Y, we check which of $(I_1^X)_1$ or $(I_1^Y)_1$ is larger ($5k$ gates) and send the corresponding interval to the middle part ($3k\lfloor n/m \rfloor$ gates). To find a pair to it, we check condition (5.2) with I_1^Y and (5.3) with I_1^X for all intervals and zero the intervals from the wrong array or which do not satisfy the condition. The number of gates used is

$$3k\left\lfloor\frac{n}{m}\right\rfloor + 2m\cdot(10k + 2k\left\lfloor\frac{n}{m}\right\rfloor).$$

The middle part consists of $2m-1$ recursive circuits, therefore the number of gates within it equals

$$(2m-1)\cdot f(\lfloor n/m\rfloor).$$

The circuits output $\lfloor n/m\rfloor$-bit vectors that show the numbers in the Y-interval given to the circuit that occur in the X-interval given to the circuit. Some circuits have the same Y-interval, or have nonsense inputs (when one input is an all-zero bit vector).

The conclusion brings together all the vectors formed in the middle part. First of all, we zero all output from a recursive circuit if the input was invalid. This happens when a certain interval, say I_i^X, does not have internal beginning in any interval I_j^Y, as checked in the preamble. For each pair of intervals, we need m more OR-gates in the preamble and $\lfloor n/m\rfloor$ gates to zero the output. Now, the outputs of the $m-1$ pairs of intervals where a predetermined Y-interval has internal beginning can readily be incorporated to the final output by OR-gates, since their place in the final vector is known ($(m-1)\lfloor n/m\rfloor$ gates overall).

For the m other pairs of intervals where an X-interval (if any) was predetermined, their intersection outputs may go to any of the m sections in the final output vector, depending on which Y-interval is in the pair. So, we join each of these outputs to all m sections in the final output (using $m\lfloor n/m\rfloor$ OR-gates), but zero all copies except the right one (using $m\lfloor n/m\rfloor$ AND-gates). The right copy is chosen in the preamble, it is the one that satisfies condition (5.3). So, the conclusion has overall complexity

$$(2m-1)(m+\left\lfloor\frac{n}{m}\right\rfloor) + (m-1)\left\lfloor\frac{n}{m}\right\rfloor + m\cdot 2m\left\lfloor\frac{n}{m}\right\rfloor$$

The circuit is thus defined. Now we turn to the estimation of its complexity (the number of gates). Let us bring together all the terms: preamble, middle part, conclusion, and handling non-integer n/m:

$$\begin{aligned} f(n) \;=\; & (2m-2)\cdot m\cdot(10k+2k\left\lfloor\frac{n}{m}\right\rfloor)+ \\ & +3k\left\lfloor\frac{n}{m}\right\rfloor+2m\cdot(10k+2k\left\lfloor\frac{n}{m}\right\rfloor)+ \\ & +(2m-1)\cdot f(\lfloor n/m\rfloor)+ \\ & +(2m-1)(m+\left\lfloor\frac{n}{m}\right\rfloor)+(m-1)\left\lfloor\frac{n}{m}\right\rfloor+ \\ & +m\cdot 2m\left\lfloor\frac{n}{m}\right\rfloor+4kmn+n \end{aligned}$$

It follows that

$$f(n) \;<\; c_1\, n+c_2+(2m-1)\cdot f(\lfloor n/m\rfloor),$$

where

$$c_1=(8k+2)\,m+4+3\,\frac{k}{m}; \quad c_2=(20k+2)\,m^2.$$

As a basis for recursion, we have

$$n<m \implies f(n)<2km^2<c_2.$$

Let us prove that

$$f(n) \;<\; (c_1+c_2)\cdot n^{\log_m(2m-1)}\cdot(1+\log_m n) \tag{5.4}$$

The proof is, of course, by induction. For $0<n<m$, we have

$$f(n)<c_2<(c_1+c_2)\cdot n^{\log_m(2m-1)}\cdot(1+\log_m n).$$

Now, assume that we have (5.4) for $\lfloor n/m \rfloor$, prove it for n:

$$\begin{aligned}
f(n) \;&<\; c_1\, n + c_2 \;+\; (2m-1)\cdot f(\lfloor n/m \rfloor) \\
&<\; c_1\, n + c_2 \;+\; (2m-1)(c_1 + c_2)\,\cdot \\
&\qquad \cdot \left(\frac{n}{m}\right)^{\log_m(2m-1)} (1 + \log_m \frac{n}{m}) \\
&=\; c_1\, n + c_2 \;+\; (c_1 + c_2)\cdot n^{\log_m(2m-1)}\,\cdot \\
&\qquad \cdot (1 + \log_m n - 1) \\
&=\; (c_1 + c_2)\cdot n^{\log_m(2m-1)} \cdot (1 + \log_m n) \;+ \\
&\qquad + c_1\, n + c_2 \;-\; (c_1 + c_2)\cdot n^{\log_m(2m-1)}
\end{aligned}$$

Since $2m - 1 \geqslant m$, we have $\log_m(2m-1) \geqslant 1$ and

$$(c_1 + c_2)\cdot n^{\log_m(2m-1)} \;\geqslant\; (c_1 + c_2)\, n$$

The induction step is proven, and with it the bound (5.4). It remains to recall that $m = \lceil 2^{1/\varepsilon} \rceil$:

$$\begin{aligned}
\log_m(2m-1) &< 1 + \frac{1}{\log m} \leqslant 1 + \frac{1}{\log(2^{1/\varepsilon})} = 1 + \varepsilon; \\
\log_m n &= \frac{\log n}{\log m} \leqslant \frac{\log n}{\log(2^{1/\varepsilon})} = \varepsilon\, \log n.
\end{aligned}$$

Therefore, finally,

$$f(n) \;<\; c\cdot n^{1+\varepsilon}\cdot(1 + \varepsilon\, \log n)$$

where

$$c \;=\; c_1 + c_2 \;=\; O(km^2) \;=\; O(k\cdot 4^{1/\varepsilon}).$$

□

Chapter 6

Conclusions

Here I would like to address the question of usefulness of the results presented in this dissertation, both for theory of privacy preserving information sharing and for practical applications, as well as outline some possible directions of future work.

For Chapter 3, the main contributions are: defining privacy breaches and showing a way to randomize itemsets so that privacy breaches are limited; developing a mathematical apparatus for the estimation of support and confidence of association rules from randomized data (and of their precision); making experiments with real-life data to evaluate practicality of the approach. The idea of hiding a small piece of valuable information within a large volume of noise looks quite general and may be useful for randomizing other types of data. Of course, it causes a blowup in the size of the record, but this may be treated by using pseudorandomness as explained in Section 4.4. The use of matrices and partial supports in the recovery of aggregate statistics is quite convenient and simplifies derivations, as well as provides some extra insight at the randomization process. The real-life data experiments have demonstrated that, even though there are lots of near-zero frequency itemsets, we do not see many of them among the false positives, which is a very encouraging indication. Of course, the requirement of having many millions of short itemsets for a good-quality recovery while preserving privacy is somewhat tough and limits practical usefulness of the method, but there are ways to possibly improve performance. If most respondents have long transactions, the server may ask them only about small subsets of their transactions, asking different questions to different respondents and choosing the questions most helpful for the recovery of frequent associations. Future work may also consider using randomization and cryptography simultaneously, to get the strengths of both without the weaknesses of either.

In Chapter 4 we presented a new definition of privacy breaches, separated them into two classes, and developed a general approach, called amplification, that provably limits breaches. Amplification can be used to limit privacy breaches with respect to any single-record property. More importantly, unlike earlier approaches, this approach does not require knowledge of the data distribution to provide privacy guarantees. The simplicity of amplification condition allows to apply it in various sophisticated settings, such as randomization of data structures, compression and pseudorandomness, randomization on top of other secure multi-party operations such as fast but approximate query answering. Conceptually, amplification condition is a way to restrict the disclosure capability of a randomization operator by looking just at the operator itself. Other such measures may exist, better suited for various specific applications or privacy definitions. Studying the properties of these measures is an important future direction.

Chapter 5 identified information integration with minimal sharing as a new area for future database research. It presented novel protocols for three key operations: intersection, intersection size, and equijoin and proved that these protocols disclose minimal information apart from the query result. It also showed how new applications can be built using the proposed protocols. An equijoin and set intersection is a common database operation, so the protocols are practically useful, especially if the encryption is done in a specialized hardware module. Some interesting directions for future research include: analyzing the tradeoff between the additional information being disclosed and efficiency; implementing other common database operations in a secure way, and developing methods to combine several operations together without disclosing intermediate results; providing security against malicious attackers, in particular against an attack through malicious input substitution (e.g., a solution might be a zero-knowledge proof of the input's origin); exploring the possibility to compile database queries into efficient Boolean circuits.

BIBLIOGRAPHY

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

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th Symposium on Principles of Database Systems*, Santa Barbara, California, USA, May 2001.

[3] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 86–97, San Diego, California, USA, June 9–12 2003.

[4] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington DC, USA, May 26–28 1993.

[5] R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, August 20–23 2002.

[6] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[7] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Implementing P3P using database technology. In *Proc. of the 19th Int'l Conference on Data Engineering*, Bangalore, India, March 2003.

[8] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *Proc. of the 12th Int'l World Wide Web Conference*, Budapest, Hungary, May 2003.

[9] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, 1996.

[10] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, September 1994.

[11] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. Research Report RJ 9839, IBM Almaden Research Center, San Jose, California, June 1994.

[12] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of the 19th ACM SIGMOD Conference on Management of Data*, Dallas, Texas, USA, May 2000.

[13] S. Agrawal, V. Krishnan, and J. R. Haritsa. On addressing efficiency concerns in privacy-preserving mining. In *Proceedings of the 9th International Conference on Database Systems for Advances Applications (DASFAA 2004)*, volume 2973 of *Lecture Notes in Computer Science*, pages 113–124, Jeju Island, Korea, March 17–19 2004. Springer.

[14] N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):777–780, September 1987.

[15] S. Ajmani, R. Morris, and B. Liskov. A trusted third-party computation service. Technical Report MIT-LCS-TR-847, MIT, May 2001.

[16] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[17] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 10–20, Philadelphia, Pennsylvania, USA, May 31–June 2 1999.

[18] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 20–29, Philadelphia, Pennsylvania, USA, May 22–24 1996.

[19] M. J. Atallah and W. Du. Secure multi-party computational geometry. Technical Report TR 2001-48, CERIAS, Purdue University, Providence, Rhode Island, USA, August 8–10 2001. The 7th International Workshop on Algorithms and Data Structures.

[20] V. Atluri and J. Hale, editors. *Research Advances in Database and Information Systems Security*. Kluwer Academic Publishers, Seattle, Washington, USA, July 25–28 1999. Thirteenth Working Conference on Database Security.

[21] R. Bayardo. Efficiently mining long patterns from databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Seattle, Washington, 1998.

[22] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, U.S.A., 1993.

[23] J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93, Workshop on the Theory and Application of Cryptographic Techniques, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, Lofthus, Norway, May 23–27 1993. Springer-Verlag.

[24] D. Boneh. The decision diffie-hellman problem. In *Proceedings of the 3rd International Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, Portland, Orgeon, USA, June 21–25 1998. Springer-Verlag.

[25] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. In D. Coppersmith, editor, *Advances in cryptology – CRYPTO'95: 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465, Santa Barbara, California, USA, August 27–31 1995. Springer-Verlag.

[26] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. CRC Press, Boca Raton, Florida, USA, 1984.

[27] Business Week. *Privacy on the Net*, March 2000.

[28] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Advances in Cryptology – EUROCRYPT '99*, 1592:402–414, 1999.

[29] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security*. Addison Wesley, 1995.

[30] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data, 2004. Cryptology ePrint Archive: Report 2004/051.

[31] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[32] G. Chen and S. Keller-McNulty. Estimation of identification disclosure risk in microdata. *Journal of Official Statistics*, 14(1):79–95, 1998.

[33] F. Y. Chin and G. Ozsoyoglu. Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, SE-8(6):574–582, April 1982.

[34] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of 29th ACM Symposium on Theory of Computing*, pages 304–313, 1997.

[35] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the Association for Computing Machinery*, 45(6):965–981, November 1998.

[36] C. A. W. Citteur and L. C. R. J. Willenborg. Public use microdata files: Current practices at national statistical bureaus. *Journal of Official Statistics*, 9(4):783–794, 1993.

[37] C. Clifton and D. Marks. Security and privacy implications of data mining. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 15–19, May 1996.

[38] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, 1986.

[39] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking: Principles and Practice*. Morgan Kaufmann, 2001.

[40] L. H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75(370):377–395, April 1980.

[41] L. Cranor, J. Reagle, and M. Ackerman. Beyond concern: Understanding net users' attitudes about online privacy. Technical Report TR 99.4.3, AT&T Labs–Research, April 1999.

[42] L. F. Cranor, editor. *Special Issue on Internet Privacy*. Comm. ACM, 42(2), Feb. 1999.

[43] U. Dayal and H.-Y. Hwang. View definition and generalization for database integration in a multidatabase system. *IEEE Transactions on Software Eng.*, 10(6):628–645, 1984.

[44] D. Defays and N. Anwar. Micro-aggregation: a generic method. In *Proceedings of the 2-nd International Symposium on Statistical Confidentiality*, pages 69–78. Luxembourg: Office for Official Publications of the European Communities, 1995.

[45] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society, Series B*, 39:1–38, 1977.

[46] D. Denning, P. Denning, and M. Schwartz. The tracker: A threat to statistical database security. *ACM Transactions on Database Systems*, 4(1):76–96, March 1979.

[47] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[48] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22-nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '03)*, pages 202–210, San Diego, California, USA, June 10–12 2003.

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

[50] A. Dobra, S. E. Fienberg, and M. Trottini. Assessing the risk of disclosure of confidential categorical data. In *Proceedings of the 7th Valencia International Meeting on Bayesian Statistics*, 2002.

[51] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 21st ACM SIGMOD Conference on Management of Data*, Madison, Wisconsin, USA, June 3–6 2002.

[52] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, January 2002.

[53] J. Domingo-Ferrer, A. Oganian, and V. Torra. Information-theoretic disclosure risk measures in statistical disclosure control of tabular data. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM '02)*, pages 227–231, Edinburgh, Scotland, July 24–26 2002.

[54] P. Doyle, J. I. Lane, J. J. M. Theeuwes, and L. V. Zayatz, editors. *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*. Elsevier, 2001.

[55] W. Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *2001 ACSAC: Annual Computer Security Applications Conference*, pages 102–110, New Orleans, Louisiana, USA, December 10–14 2001.

[56] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003)*, Washington DC, USA, August 24–27 2003.

[57] G. T. Duncan, R. Krishnan, R. Padman, P. Reuther, and S. Roehrig. Cell suppression to limit content-based disclosure. In *Proceedings of the 30th Hawaii International Conference on System Sciences*, volume 3, pages 552–560. IEEE Computer Society Press, 1997.

[58] G. T. Duncan and S. Mukherjee. Optimal disclosure limitation strategy in statistical databases: Deterring tracker attacks through additive noise. *Journal of the American Statistical Association*, 95(451):720–729, 2000.

[59] G. T. Duncan and R. W. Pearson. Enhancing access to microdata while protecting confidentiality. *Statistical Science*, 6:219–239, 1991.

[60] The Economist. *The End of Privacy*, May 1999.

[61] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.

[62] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors. *Management of Heterogeneous & Autonomous Database Systems*. Morgan Kaufmann, 1999.

[63] V. Estivill-Castro and L. Brankovic. Data swapping: Balancing privacy against precision in mining for logic rules. In M. K. Mohania and A. M. Tjoa, editors, *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK-99)*, volume 1676 of *Lecture Notes in Computer Science*, pages 389–398, Florence, Italy, August 30 – September 1 1999. Springer-Verlag.

[64] Eurogramme, editor. *Statistical Data Confidentiality: Proceedings of the Joint Eurostat/UN-ECE Work Session on Statistical Data Confidentiality*. Eurostat, Thessaloniki, Greece, March 1999.

[65] European Union. *Directive on Privacy Protection*, October 1998.

[66] T. Evans, L. Zayatz, and J. Slanta. Using noise for disclosure limitation of establishment tabular data. *Journal of Official Statistics*, 14(4):537–551, 1998.

[67] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 205–210, Santa Barbara, California, USA, 1982.

[68] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22-nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 211–222, San Diego, California, USA, June 9–11 2003.

[69] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 217–228, Edmonton, Alberta, Canada, July 23–26 2002.

[70] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules (invited journal version). *Information Systems*, 29(4):343–364, June 2004.

[71] Federal Committee on Statistical Methodology, USA. *Report on Statistical Disclosure Limitation Methodology*, May 1994. Statistical Policy Working Paper 22.

[72] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, Crete, Greece, July 8–12 2001.

[73] I. Fellegi. On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337):7–18, March 1972.

[74] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology, Proceedings of Crypto 86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[75] S. E. Fienberg. Confidentiality and disclosure limitation methodology: Challenges for national statistics and statistical research. Technical Report TR-668, Department of Statistics, Carnegie-Mellon University, 1997.

[76] S. E. Fienberg and U. E. Makov. Confidentiality, uniqueness, and disclosure limitation for categorical data. *Journal of Official Statistics*, 14(4):385–397, 1998.

[77] S. E. Fienberg, U. E. Makov, and R. J. Steele. Disclosure limitation using perturbation and related methods for categorical data. *Journal of Official Statistics*, 14(4):485–502, 1998.

[78] S. E. Fienberg and J. McIntyre. Data swapping: Variations on a theme by dalenius and reiss. In J. Domingo-Ferrer and V. Torra, editors, *Privacy in Statistical Databases. Proceedings of CASC Project International Workshop, PSD 2004*, volume 3050 of *Lecture Notes in Computer Science*, pages 14–29, Barcelona, Spain, June 9–11 2004. Springer-Verlag.

[79] L. Franconi and J. Stander. Microaggregation and model based methods for disclosure limitation and their application to business microdata. In *The 2-nd Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality*, 2001.

[80] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – Proceedings of Eurocrypt '2004, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, May 2–6 2004. Springer-Verlag.

[81] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Journal of Official Statistics*, 9(2):383–406, 1993.

[82] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000.

[83] O. Goldreich. Secure multi-party computation, June 1998. Working Draft, will appear in Volume 2 of “Foundations of Cryptography”.

[84] O. Goldreich. *Foundations of Cryptography: Volume I, Basic Tools*. Cambridge University Press, August 6 2001.

[85] O. Goldreich. *Foundations of Cryptography: Volume II, Basic Applications*. Cambridge University Press, May 10 2004.

[86] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[87] S. Goldwasser and M. Bellare. Lecture notes on cryptography, August 2001. Lecture notes for a summer course on cryptography, taught by the authors at the MIT in 1996–2001.

[88] J. M. Gouweleeuw, P. Kooiman, L. C. R. J. Willenborg, and P.-P. de Wolf. Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14(4):463–478, 1998.

[89] L. M. Haas, R. J. Miller, B. Niswonger, M. T. Roth, P. M. Schwarz, and E. L. Wimmers. Transforming heterogeneous data with database middleware: Beyond integration. *IEEE Data Engineering Bulletin*, 22(1), 1999.

[90] S. Halevi, R. Krauthgamer, E. Kushilevitz, and K. Nissim. Private approximation of np-hard functions. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 550–559, Hersonissos, Greece, 2001.

[91] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications*. Springer Verlag, August 1999. 440 pp.

[92] J. J. Hickey and A. Nogin. Fast tactic-based theorem proving. In J. Harrison and M. Aagaard, editors, *Theorem Proving in Higher Order Logics: 13th International Conference (TPHOLs 2000)*, volume 1869 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 2000.

[93] B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1-st ACM Conference on Electronic Commerce*, pages 78–86, Denver, Colorado, USA, November 1999.

[94] P. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[95] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 279–288, Edmonton, Alberta, Canada, July 23–26 2002.

[96] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260, 2001.

[97] S. Jajodia and R. Sandhu. Polyinstantiation integrity in multilevel relations. In *IEEE Symp. on Security and Privacy*, 1990.

[98] N. Jefferies, C. Mitchell, and M. Walker. A proposed architecture for trusted third party services. In *Cryptography Policy and Algorithms Conference*. Springer LNCS v 1029 pp 98–104, July 1995.

[99] N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding : Steganography and Watermarking - Attacks and Countermeasures*, volume 1 of *Advances in Information Security*. Kluwer Academic Publishers, 2001.

[100] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *Proceedings of the Cryptographer's Track at the RSA Conference 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262, San Jose, CA, USA, February 18–22 2002. Springer-Verlag.

[101] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Madison, Wisconsin, USA, June 2 2002.

[102] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining*, Melbourne, FL, November 2003.

[103] J. J. Kim and W. E. Winkler. Masking microdata files, 1995.

[104] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. On the value of private information. In *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge*, 2001.

[105] J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proceedings of the 19-th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 86–91, Dallas, Texas, USA, 2000.

[106] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the IEEE Symp. on Found. of Computer Science (FOCS)*, 1997.

[107] D. Lambert. Measures of disclosure risk and harm. *Journal of Official Statistics*, 9(2):313–331, 1993.

[108] C. Landwehr. Formal models of computer security. *ACM Computing Surveys*, 13(3):247–278, 1981.

[109] Y. Lindell. Secure multiparty computation for privacy preserving data mining. IBM T.J. Watson Research Center, USA.

[110] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

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

[112] R. J. A. Little. Statistical analysis of masked data. *Journal of Official Statistics*, 9(2):407–426, 1993.

[113] K. Liu, H. Kargupta, and J. Ryan. Multiplicative noise, random projection, and privacy preserving data mining from distributed multi-party data, 2003.

[114] T. F. Lunt, editor. *Research Directions in Database Security*. Springer-Verlag, 1992.

[115] F. J. C. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1978. 762 pp.

[116] W. Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall PTR, first edition, July 25 2003.

[117] P. B. Massell. Assessing the statistical disclosure risk of a demographic microdata file. In *1999 Research Conference, Federal Committee on Statistical Methodology*, 1999.

[118] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, November 1996. 304 pp.

[119] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 16 1996.

[120] R. Merkle. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Department of Electrical Engineering, Stanford University, 1979.

[121] D. Micciancio. Oblivious data structures: Applications to cryptography. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 456–464, El Paso, Texas, May 4–6 1997.

[122] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*, pages 575–586, Paris, France, 2004.

[123] R. A. Moore. Controlled data-swapping techniques for masking public use microdata sets. Technical Report SRD Report RR 96-04, U.S. Bureau of the Census, 1996.

[124] S. Mukherjee and G. T. Duncan. Disclosure limitation through additive noise data masking: Analysis of skewed sensitive data. In *Proceedings of the 30th Hawaii International Conference on System Sciences*, volume 3, pages 581–586. IEEE Computer Society Press, 1997.

[125] M. Naor and K. Nissim. Communication complexity and secure function evaluation. Technical Report TR01-062, Electronic Colloquium on Computational Complexity, September 2001.

[126] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the 31st Symposium on Theory of Computer Science*, pages 245–254, Atlanta, GA, USA, May 1–4 1999.

[127] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington DC, USA, January 7–9 2001.

[128] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1-st ACM Conference on Electronic Commerce*, pages 129–139, Denver, Colorado, USA, November 1999.

[129] Office of the Information and Privacy Commissioner, Ontario. *Data Mining: Staking a Claim on Your Privacy*, January 1998.

[130] W. Ogata and K. Kurosawa. Oblivious keyword search. *Journal of Complexity*, 20:356–371, April/June 2004.

[131] B. Preneel. Analysis and design of cryptographic hash functions. Ph.D. dissertation, Katholieke Universiteit Leuven, 1992.

[132] O. Pretzel. *Error-Correcting Codes and Finite Fields*. Oxford University Press, 1992. 398 pp.

[133] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[134] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[135] T. E. Raghunathan, J. P. Reiter, and D. B. Rubin. Multiple imputation for statistical disclosure limitation. *Journal of Official Statistics*, 19(1), 2003.

[136] J. P. Reiter. Satisfying disclosure restrictions with synthetic data sets. *Journal of Official Statistics*, 18(4), 2002.

[137] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[138] S. J. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, August 2002.

[139] Z. Rjaskova. Electronic voting schemes, April 2002. M.Sc. Thesis, Comenius University, Bratislava.

[140] M. Rotenberg. The privacy law sourcebook 2000: United states law, international law, and recent developments, 2000. Electronic Privacy Information Center.

[141] M. Rotenberg. Fair information practices and the architecture of privacy. *Stanford Technology Law Review*, 1, 2001.

[142] D. B. Rubin. Discussion: Statistical disclosure limitation. *Journal of Official Statistics*, 9(2), 1993.

[143] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[144] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[145] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, Seattle, Washington, USA, June 1–3 1998.

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

[147] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, second edition, October 18 1995.

[148] A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. Technical Memo MIT-LCS-TM-125, Laboratory for Computer Science, MIT, February 1979.

[149] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28-4:656–715, 1949.

[150] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[151] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In S. Goldwasser, editor, *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science (FOCS '94)*, pages 124–134, Santa Fe, New Mexico, USA, November 20–22 1994. IEEE Computer Society Press.

[152] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[153] A. Shoshani. Statistical databases: Characteristics, problems and some solutions. In *Proc. of the Eighth Int'l Conference on Very Large Databases*, pages 208–213, Mexico City, Mexico, September 1982.

[154] C. J. Skinner and D. J. Holmes. Estimating the re-identification risk per record in microdata. *Journal of Official Statistics*, 14(4):361–372, 1998.

[155] S. W. Smith and D. Safford. Practical private information retrieval with secure coprocessors. Research Report RC 21806, IBM, July 2000.

[156] P. Stachour and B. Thuraisingham. Design of LDV: A multilevel secure relational database management system. *IEEE Trans. Knowledge and Data Eng.*, 2(2):190–209, 1990.

[157] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, second edition, February 27 2002.

[158] D. R. Stinson. *Cryptography: Theory and Practice*, chapter 4. Chapman & Hall/CRC, second edition, 2002.

[159] M. Szydlo. Merkle tree traversal in log space and time. In *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 2–6 2004.

[160] K. Thearling. Data mining and privacy: A conflict in making. *DS**, March 1998.

[161] Time. *The Death of Privacy*, August 1997.

[162] T. M. Truta, F. Fotouhi, and D. Barth-Jones. Disclosure risk measures for microdata. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management (SSDBM 2003)*, pages 15–22, Cambridge, MA, USA, July 9–11 2003. IEEE Computer Society.

[163] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23–26 2002.

[164] S. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, March 1965.

[165] A. Westin. E-commerce and privacy: What net users want. Technical report, Louis Harris & Associates, June 1998.

[166] A. Westin. Privacy concerns & consumer choice. Technical report, Louis Harris & Associates, Dec. 1998.

[167] A. Westin. Freebies and privacy: What net users think. Technical report, Opinion Research Corporation, July 1999.

[168] G. Wiederhold. Intelligent integration of information. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, Washington, DC, 1993.

[169] L. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*, volume 155 of *Lecture Notes in Statistics*. Springer-Verlag, 2001.

[170] Y. Xia, Y. Yang, Y. Chi, and R. R. Muntz. Mining association rules with non-uniform privacy concerns. Technical Report CSD-TR 040015, University of California at Los Angeles, 2004.

[171] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27rd Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Canada, October 1986.

[172] L. Zayatz. Sdc in the 2000 u.s. decennial census. In J. Domingo-Ferrer, editor, *Inference Control in Statistical Databases: From Theory to Practice*, volume 2316 of *Lecture Notes in Computer Science*, pages 193–202. Springer-Verlag, 2002.

[173] L. Zayatz, R. Moore, and B. T. Evans. New directions in disclosure limitation at the census bureau. In *Proceedings of the Government Statistics Section, Joint Statistical Meetings*. American Statistical Association, 1996.