QUERY AUDITING AGAINST PARTIAL DISCLOSURE

by

MAYUR MOTGI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

UNIVERSITY OF TEXAS AT ARLINGTON

May 2009

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family and friends for their support and encouragement that made all this possible.

I would like to offer special thanks to my supervising professor Dr. Nan Zhang for his consistent help, support, and guidance throughout this endeavor. His patience, experience, and knowledge have been invaluable throughout my research, and I am truly grateful for this. I would also like to thank Dr. Heng Huang and Dr. Donggang Liu for their encouragement during the course of the project.

I would also like to thank Dr. Bahram Khalili, my graduate advisor and professor, for his help, advice and guidance throughout my M.S.

April 11, 2008

ABSTRACT

QUERY AUDITING AGAINST PARTIAL DISCLOSURE

Mayur Motgi, M.S.

The University of Texas at Arlington, 2008

Supervising Professors: Nan Zhang and Heng Huang

Many government agencies, businesses, and nonprofit organizations need to collect, analyze, and report data about individuals in order to support their short-term and long-term planning activities. Statistical Databases therefore contain confidential information such as income, credit ratings, type of disease, or test scores of individuals. Such data are typically stored online and analyzed using sophisticated database management systems (DBMS) and software packages. On one hand, such database systems are expected to satisfy user requests of aggregate statistics related to non-confidential and confidential attributes. On the other hand, the system should be secure enough to guard against a user's ability to infer any confidential information related to a specific individual represented in the database. A major privacy threat is the adversarial inference of individual (private) tuples from aggregate query answers. Most existing work focuses on the exact disclosure problem, which is inadequate in practice. We propose a novel auditing algorithm for defending against partial disclosure.

We introduce ENTROPY-AUDITING, an efficient query-auditing algorithm for partial disclosure that supports a mixture of common aggregate functions. In particular, we classify aggregate functions into two categories: MIN-like (e.g., MIN and MAX) and SUM-like (e.g., SUM and MEDIAN), and support a combination of them. Our proposed scheme utilizes an exact-auditing algorithm as a primitive function, and supports a combination of queries with various aggregate functions (e.g., SUM, MIN, MAX). We also present a detailed experimental evaluation of our PARTIAL-AUDITING approach.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

A statistical database (SDB) is a database that allows its users to retrieve only aggregate statistics (such as mean or count) over subsets of its data. A statistical database can be viewed as a set of *n* records (or tuples), each record consisting of *m* attribute values. Some of these attributes are used in 'characteristic formulae' for specifying 'query sets', and they are called characteristic attributes. Other attributes, called quantitative attributes, are used to compute statistics. Some of the quantitative attributes are confidential and are to be kept secret for all records in a database. The only queries accepted by the system, concerning attributes are statistical types of queries, such as COUNT, SUM, MEAN, MIN, MAX. Table 1 shows a Hospital Database in which each record corresponds to a patient. The attributes are: Name, City, Sex, Status, Child, Age, Start and Salary. The characteristic attributes are: City, Sex, Status, Child, Age and YOB, while the confidential quantitative attribute is Disease.

Let X be a data set consisting of n entries $X = \{x_1, \ldots, x_n\}$ for the quantitative attribute like salary. A statistical query $q = (Q, f)$ specifies a subset of the data set entries $Q \subseteq [n]$ and a function f. The answer $f(Q)$ to the statistical query

is f applied to the subset of entries $\{x_i \mid i \in Q\}$. Some typical choices for f are sum, max, min, and median.

Table 1.1: Hospital Database

Name	*City*	*Sex*	*Status*	*Child*	*Age*	*YOB*	*Disease*
Emily Walthom	London	F	S	0	21	1988	Typhoid
Jacob Oram	Wellington	M	D	2	40	1969	Measles
Michael Kelso	Luxemberg	M	S	0	27	1982	Blood
Elane King	Amsterdam	F	W	3	54	1955	Yellow
Monica Geller	Yorkshire	F	M	3	45	1964	Small Pox
Daniel Vittori	Surrey	M	M	1	39	1970	AIDS
Christopher Rice	Sussex	M	S	0	23	1986	Typhoid
Arun Nayar	Hampshire	M	M	2	37	1972	Lung
Elizabeth Hannah	Amsterdam	F	S	0	60	1949	Measles

In this thesis, we focus on the online query auditing problem: Suppose that the queries $q_1, \ldots, q_{t-1}$ have already been posed and the answers $a_1, \ldots, a_{t-1}$ have already been given, where each a_j is either the true answer to the query $f_j(Q_j)$ or "denied" for $j = 1, \ldots, t-1$. Given a new query q_t, deny the answer if privacy may be breached, and provide the true answer otherwise.

One may view the online auditing problem as a game between an auditor and an attacker. The auditor monitors an online sequence of queries, and denies queries with the purpose of ensuring that an attacker that issues queries and observes their answers would not be able to 'stitch together' this information in a manner that breaches privacy. A naive and utterly dissatisfying solution is that the auditor would deny the answer to every query. Hence, we modify the game a little, and look for auditors that provide as much information as possible (usually, this would be "large

scale" information, almost insensitive to individual information) while preserving privacy (usually, this would refer to "small-scale" individual information).

Most existing work in online query auditing focuses on the exact disclosure problem, which only prevents a user from inferring the exact values of individual tuples. This problem provides insights for query auditing from the combinatorial perspective but, as widely recognized [6], does not suffice practically because much private information about an individual can still be disclosed without revealing its exact value.

Some recent work addresses the partial disclosure of individual tuples, but only supports one specific type of queries. For example, the partial-disclosure algorithm proposed by Nabar et al[1] only deals with MIN/MAX queries. The simulatable-auditing algorithm proposed by Kenthapadi et al [2] only deals with SUM queries over real-valued data drawn uniformly from a bounded range. A practical workload of aggregate queries, however, usually consists of both SUM and MIN/MAX queries, and thus cannot be handled by existing algorithms.

We introduce the concept of ENTROPY-AUDITING, an algorithm for query auditing against partial disclosure. It supports a combination of common aggregate functions. The aggregate functions are classified as MIN-like(MIN and MAX queries) and SUM-like(SUM and MEDIAN queries). The algorithm does not leak information due to query denials and thus follows the simulatable auditing [1] approach.

The two main ideas of ENTROPY-AUDITING are: using the information theory framework to transform how much information about an individual tuple an aggregate query discloses to how much information about an aggregate query can be inferred from an individual tuple; and the usage of existing exact-disclosure SUM-auditing algorithms as a primitive function to compute the amount of information about an aggregate query answer that can be inferred from an individual tuple. While the transformation enables us to consider different types of aggregate functions separately in query auditing, choosing different underlying exact-disclosure algorithms, allows a flexible tradeoff between the efficiency of query auditing and the availability of aggregate queries.

Our algorithm is the first to support a combination of SUM and MIN/MAX queries in query auditing against partial disclosure. We also describe a comprehensive set of experiments that demonstrate the effectiveness of our query-auditing algorithm.

The rest of this thesis is organized as follows: Next, we give a brief overview of the related work to secure statistical databases against disclosure in chapter 2. We specify our problem and define the performance measure in chapter 3. We introduce our information theory framework in chapter 4. The basic idea of ENTROPY-AUDITING is defined in chapter 5. Auditing SUM queries using an exact-disclosure algorithm as primitive is described in chapter 6. Chapter 7 provides the details of

ENTROPY-AUDITING algorithm. In Chapter 8, we present a detailed experimental evaluation of PARTIAL-AUDITING followed by final remarks in Chapter 9.

CHAPTER 2

RELATED WORK

2.1. Statistical databases

As mentioned earlier, Statistical Databases are ones from which users request statistical queries (Max, Sum, Average etc.). A statistical database usually has a set of n records (or tuples), each record consisting of m attribute values. A typical statistical database is shown in Table 1. The statistical queries are answered using a subset of data in the database. A secure statistical database has data that is private and not visible by everyone. Some of the attributes are confidential and are to be kept secret for all records in a database. The main concern to the privacy of the data is the users compromising the database by asking a series of questions and inferring private information from the answers. The usability of the statistical database is defined as the ratio of number of queries that can be answered without compromising any data to the total number of queries [4].

2.2. Approaches to Secure Statistical Databases

The problem of compromise for statistical databases can be defined as: for a sequence of allowed queries $q_1, \ldots, q_m$, a database is compromised if, for a given value of private data x_i, the query answers remain same irrespective of the values of

other data points all. Overlapping of queries play a major role in the compromise of the database [4]. Now because these aggregate queries are over a subset of data, the problem of compromise is more prominent in statistical databases. The initial approaches for securing statistical databases were divided into 4 categories [5]: *conceptual, query restriction, data perturbation, and output perturbation.*

2.2.1. Conceptual Approach

The conceptual approach [5] describes a framework wherein the security is guaranteed by introducing a concept of smallest non-decomposable subpopulation. These populations always contain zero or more than 2 entities which prevents the information of any entity from being disclosed. The framework also described a lattice model where data is stored in tabular form at different levels of aggregation. Though the framework sounds very promising, its practical application is restricted.

2.2.2. Query restriction

Query restriction is described in 5 methods: *query-set-size control, query-set-overlap control, auditing, cell suppression, and partitioning. Query-set-size-control* restricts the size of the query set i.e. the number of data points that a query refers to. The drawback of this method is that it does not consider the overlapping of queries.

Query-set-overlap-control [5] method restricts the overlapping in the queries. The main reason for the disclosure of information is the overlapping in queries. Thus, control of overlapping is a main concern. This method suffers form the following drawbacks: inability to control cooperative compromise, inability to

release statistics for both sets ad subsets and the requirement of keeping the user profile up to date.

Basic *auditing* [5] of statistical databases involves keeping a log of all the queries answered for a given user and checking if compromise occurs when a new query is answered. This type of auditing is also called online query auditing. There is another version of auditing called offline query auditing that takes a set of queries and checks if the queries are safe to answer [9]. If not, the maximum subset of safe queries is returned [14]. Auditing usually takes a lot of CPU time and storage space when the history of answered queries is stored in the native format. Thus auditing also includes a practical technique for storage and management of the history of answered user queries. Most of the auditing approaches consider representing the query history in the form of a matrix where each row represents an answered query in a reduced format. Using Gaussian elimination, exact disclosure of an entity can be identified efficiently.

Partitioning [5] involves clustering individual entities of the population in a number of mutually exclusive subsets, called atomic populations. This is similar to the conceptual framework described earlier. As long as the atomic population does not contain one entity, disclosure of entity information is prevented. Smaller populations suffer from problems of partial disclosure and robustness.

The method of *cell suppression* involves suppression of cells from the released tables that would lead to disclosure of confidential information. This

process usually involves suppression of other non-confidential cells that might cause disclosure of confidential information(called complimentary suppression). The calculation of complementary suppression is very complex and this method becomes impractical for queries with complex syntax.

2.2.3. Data-Perturbation

Data-Perturbation [5] approaches fall into two main categories, the probability-distribution category and the fixed-data-perturbation category. There is a large risk of bias being introduced in any data-perturbation approach. The methods in probability-distribution category replace the original database by its probability distribution. As this method is equivalent to noise-addition, it introduces sampling bias in query responses. Increasing the size of the database reduces bias but leads to less security of confidential attributes. Thus most of the methods try to find a balance between amount of biasing and level of security. Fixed-Data Perturbation [5] for numerical attributes involves adding perturbation based on a random-perturbation variable to the original database values. Though this method suffers from bias but the advantage it has over probability-distribution methods is that the addition of noise much clearer. This method is appropriate for online statistical databases; the original and perturbed data can be maintained and users can access perturbed data.

2.2.4. Output-Perturbation

The bias problem in the data perturbation approach is less severe in the *Output-Perturbation* [5] approach. This is due to the fact that the selection of query

set is based on the original values, not the perturbed values. Methods include drawing a sample from the query set itself in such a way that lexicographically the same queries produce the same sample from the query set. Varying-Output Perturbation [5] method adds varying perturbation to the data that is used to answer a query. The value of the query set is perturbed. This perturbation is correlated to make the number of queries needed for a disclosure grow exponentially. Output perturbation may take some form of rounding, where the answer to the query is rounded up or down to the nearest multiple of a certain base *b*. The output-perturbation approach suffers from the possibility of a null-query-set situation. Compromised methods such as regression-based ones could easily take advantage of the null-query-set situation.

The random-sample-queries method [5], the varying-output-perturbation method, and the fixed-data-perturbation method, are considered among the most promising security-control methods for on-line, dynamic statistical databases (the most difficult type of statistical databases).

2.3. Query auditing on OLAP

Online Analytical Processing(OLAP) allows quick answers to multi-dimensional analytical queries. OLAP enables a user to easily and selectively extract and view data from different points of view. OLAP data is stored in a multi-dimensional database. Usually two kinds of approaches are considered for privacy protection in OLAP, namely inference control and input/output perturbation.

The perturbation algorithm [6] is usually a randomization algorithm that takes a table T and returns a table T ` with the same number of rows and columns. The algorithm is usually public but the random values used are hidden. So the confidentiality of the data depends on the level of randomness in the algorithm. Thus there is a tradeoff between the level of randomness and the level of security. An efficient algorithm to reconstruct the original data from the perturbed values is also required.

The inference control should ideally eliminate inference of any data point from the answered queries. The optimal boolean query-auditing problem is proved to be co-NP-complete [7]. There is always a tradeoff between query availability and security of the database. Many existing inference control algorithms for OLAP make query answer/reject decisions based on the number of pre-known and sensitive cells [34], [35], [36]. Most only address the exact disclosure of sensitive cells [31], [32], [33], [35] and only support SUM queries [31], [33]–[35].

2.4. Simulatable Auditing

Simulatable auditing [2] is a new model where query denials provably do not leak information. An auditor is simulatable if an attacker, knowing the query-answer history, could make the same decision as the simulatable auditor regarding a newly posed query. Hence, the decision to deny a query does not leak any information beyond what is already known to the attacker. The privacy definition for partial compromises in this model requires that the a-priori probability that a sensitive value

lies within some small interval is not that different from the posterior probability (given the query answers). It assumes that there exists an underlying probability distribution from which the data is drawn. The essence of the definition is that for each data element x_i and small-sized interval I, the auditor ensures that the prior probability that x_i falls in the interval I is about the same as the posterior probability that x_i falls in I given the queries and answers.

The simulatable-auditing algorithm proposed by Kenthapadi et al [2] deals with SUM queries over real-valued data drawn uniformly from a bounded range. The auditing algorithm computes posterior probabilities by utilizing existing randomized algorithms for estimating the volume of a convex polytope. The auditing algorithm does not access the data set while deciding whether to allow the newly posed query q_t(in particular, it does not compute the true answer to q_t). Instead, the auditor draws many data sets according to the underlying distribution, assuming the previous queries and answers, then computes an expected answer a'_t and checks whether revealing it would breach privacy for each of the randomly generated data sets.

The partial-disclosure algorithm proposed by Nabar et al [1] deals with MIN/MAX queries. They follow the simulatable auditing model proposed by Kenthapadi et al [2]. They consider probabilistic compromise for bounded range data where a significant change in the attacker's confidence about the range of a data point constitutes a privacy breach. They try to enhance the robustness of the notion of online auditing by exploring algorithms for auditing new kinds of queries under

different notions of compromise and examining the unexamined dimension of utility. They use the concept of blackbox, which takes as input a set of max queries and corresponding answers and converts them to a synopsis $B_{\max}$ containing predicates of the form [$\max(S_i) = M_i$] and [$\max(S_j) < M_j$].

They show that it suffices to consider just these predicates generated by B in detecting compromise instead of the entire sequence of past queries. The advantage of B is that we can compress a potentially long audit trail to one of size O(n) where n is the number of elements in the dataset. The synopses can be incrementally maintained — when a new query q_t arrives, the old synopsis together with q_t is combined in to a new synopsis in O($|Q_t|$) time. They consider max queries and bags of max and min queries audited under probabilistic compromise. These recent works address the partial disclosure of individual tuples, but only supports either MIN-like or SUM-like queries.

It is clear that these existing works only supports one specific type of queries. A practical workload of aggregate queries, however, usually consists of both SUM and MIN/MAX queries, and thus cannot be handled by existing algorithms. We compare the performance of our algorithm with the algorithms in [1] and [2].

CHAPTER 3

PRELIMINARIES

3.1. Problem Specification

Consider a SDB which consists of n tuples $t_1, \ldots, t_n$. Each tuple has one private attribute and many public attributes. Let $x_1, \ldots, x_n$ be the value of the private attributes for the respective tuples. An adversary tries to infer the value of the private attribute from the answers of the aggregate queries. An aggregate query $q = F(q)$ has an aggregate function F(sum, min, max etc.) applied to a subset of tuples $q \subseteq \{x_1, \ldots, x_n\}$ by specifying the values of the public attributes. These aggregate queries are also called sensitive aggregate queries as not all the information about the tuples used to answer the query is known. We consider the worst-case scenario where an adversary has unlimited computational power.

We consider the online query auditing problem: Suppose that the queries $q_1, \ldots, q_{t-1}$ have already been posed and the answers $a_1, \ldots, a_{t-1}$ have already been given, where each a_j is either the true answer to the query $f_j(Q_j)$ or "denied" for $j = 1, \ldots, t-1$. When the SDB receives q_t, it has no knowledge of the upcoming queries $q_{t+1}, \ldots, q_m$. Following a common assumption in query auditing [5], we start by assuming that an adversary does not collude with others to share the query answers they receive.

Given a new query q_t, deny the answer if privacy may be breached, and provide the true answer otherwise. The definition of privacy breach distinguishes our work from most existing work: Traditionally, a privacy breach is said to occur iff an adversary can (deterministically) infer the exact value of an individual x_i. We define the privacy breach according to a partial disclosure measure: i.e., breach occurs iff the amount of disclosure about x_i exceeds a pre-determined threshold. We shall introduce the partial disclosure measure in the following subsection.

3.2. Privacy and Query Availability Measures

The performance of query auditing should be measured in terms of privacy protection, availability of aggregate queries, and efficiency. We define the measures for partial disclosure and query availability as follows.

Partial Disclosure Measure We consider that each private data point x_i belongs to a given probability distribution. As new queries are received, the probability distribution of the data points change because each query reveals some information of the data points it refers. Intuitively, partial disclosure occurs when an adversary's posterior belief of the probability distribution of a private data point x_i (given the query history) differs from its prior belief (without knowledge of the query history). But this happens for each query answered. Thus, we measure privacy disclosure by the discrepancy between an adversary's prior and posterior belief of the distribution. For any given time, let a user's query history Q_H be the query answers the user has received. We have the following definition of partial disclosure measure:

Definition 1. *For an adversary with query history Q_H, the level of privacy disclosure is defined as:*

$$l_{\mathrm{p}}(Q_{\mathrm{H}}) = \max_i \frac{I(x_i; Q_{\mathrm{H}})}{H(x_i)}. \quad (1)$$

where $H(x_i)$ is the information entropy of x_i, and $I(x_i ; Q_H)$ is the mutual information between x and the query history Q.

The expression on the right measures the percentage of information about the data point that an adversary can infer from Q_{H}. $H(x_i)$ measures the degree of uncertainty in the distribution of x_i and $I(x_i; Q_{\mathrm{H}})$ is the reduction in degree of uncertainty due to access to Q_{H}. We consider the maximum of all the changes in x_i. More the value of $l_{\mathrm{p}}(Q_{\mathrm{H}}) \in (0, 1]$, higher is the information inferred about the sensitive data points from Q_{H}. When exact disclosure occurs, $I(x_i; Q_{\mathrm{H}}) = H(x_i)$ and $l_{\mathrm{p}}(Q_{\mathrm{H}}) = 1$. When $Q_{\mathrm{H}} = \varnothing$, there is $l_{\mathrm{p}}(Q_{\mathrm{H}}) = \max_i I(x_i; Q_{\mathrm{H}}) = 0$.

We consider a maximum acceptable level of privacy disclosure l_0. For the partial disclosure measure above, we say privacy breach has occurred iff

$$l_{\mathrm{p}}(Q_{\mathrm{H}}) > l_0 \quad (2)$$

where $l_0 \in (0, 1]$. The greater l_0 is, the more private information about x can be disclosed. When $l_0 = 1$, a privacy breach occurs iff exact disclosure happens. This reduces the problem to its traditional definition [3].

Query Availability Measure We define the level of query availability $l_{\mathrm{a}} \in [0, 1]$ as the percentage of queries issued by a user that are (correctly) answered by the SDB.

Definition 2. *For a given sequence of queries $Q = \langle q_1, \ldots, q_m \rangle$, the level of query availability is:*

$$l_{\mathrm{a}} = \frac{|\{\mathrm{q}_i | \mathrm{q}_i \textit{ is answered when } \boldsymbol{Q} \textit{ are issued in order}\}|}{m}. \tag{3}$$

Note that l_{a} depends not only on the set of queries issued by the user, but also the order in which the queries are issued.

3.3. Problem Statement for Online Query Auditing

According to the problem specification, the SDB must determine whether an aggregate query is safe to answer based on its issuer's query history and the current aggregate query.

Definition 3. *For a given user C_k and its query history Q_k, a query q issued by C_k is safe iff $\forall x \notin G_k$,*

$$l_{\mathrm{p}}(\mathrm{x}; Q_{\mathrm{k}} \cup \{\mathrm{q}\}) < l_0 \tag{4}$$

Ideally we would want a query-auditing algorithm to deny all unsafe queries and answer all safe queries. It is very difficult to achieve this(not impossible). In fact the special case where $l_0 = 1$ (Exact Disclosure) is itself proved as NP-hard by [9] when the query workload consists of SUM and MIN/MAX queries. So our algorithm denies all unsafe queries and answer as many safe queries as possible, which is a reasonable compromise. Hence we summarize our problem statement as:

Definition 4. (Problem Statement). *The objective of query auditing is:*

– Objective O_1: to reject all unsafe queries, and

– Objective O_2: when O_1 is achieved, to maximize the query availability level l_a.

CHAPTER 4

AN INFORMATION-THEORETIC SCHEME

Every time a new query is received, the computation of $l_p(Q_H)$ would help decide whether the query can be answered or out. This would also help achieve the two objectives outlined in the problem statement, as it would determine whether the incoming query is safe based on the monitored value. In this Chapter, we will present our basic ideas for the monitoring process. In particular, we will first describe the workflow of disclosure estimation, and discuss the computation involved for various aggregate functions.

For the computation of $l_p(Q_H)$, there are three challenges that we must address:

- *Efficiency with a large Q_H*: The input Q_H may contain a large number of queries which prevent $l_p(Q_H)$ from being computed efficiently. A possible solution is to exploit the monotonicity of l_p regarding to Q_H, and to perform the computation incrementally with Q_H.
- *Efficiency for Single x_i*: Note that $l_p(Q_H)$ depends on the value of $I(x_i;Q_H)/H(x_i)$. However, recall that for a given x_i, determining whether $I(x_i;Q_H)/H(x_i) = 1$ (i.e., exact disclosure) is NP-hard. Thus, we may have to tradeoff between efficiency and accuracy of computation for $I(x_i;Q_H)/H(x_i)$.

– *Efficiency with a large number of tuples*: Furthermore, even if $I(x_i;Q_H)/H(x_i)$ can be estimated efficiently, we still need to compute $l_p(Q_H)$ as the maximum level of privacy disclosure for all x_i. Since the SDB may contain a large number of tuples, we must compute l_p more efficiently than simply performing the computation sequentially for each x_i.

We now introduce an information-theoretic scheme that is designed to address these three challenges. To address the first challenge, our idea is to incrementally compute the value of $l_p(Q_H)$. Note that $l_p(Q_H) = 0$ when Q_H is empty. Thus, when the SDB answers a query q, we only need to update $l_p(Q_H)$ by computing the additional amount of information about x_i that one can derive from the new query answer q given the old history Q_H as prior knowledge. This update can be calculated using the difference between the value of l_p for the new query and query history and the value of l_p for the query history only. The expression given below represents the above:

$$l_p(q|Q_H) = l_p(\{q\} \cup Q_H) - l_p(Q_H). \quad (5)$$

Thus, we reduce the problem of computing $l_p(Q_H)$ to the computation of $l_p(q|Q_H)$.

To address the second challenge, our idea is to compute an *upper bound* on $l_p(q|Q_H)$ instead of the exact value. Let $l_{max}(Q_H)$ be such an upper bound. Note that a loose upper bound may prevent future queries from being answered, as the future queries might not satisfy the threshold. Thus, in order to achieve Objective O_2, $l_{max}(Q_H)$ must be tight enough to support an acceptable level of query availability.

The computation of such upper-bound estimate must also be efficient in order to answer (or reject) queries in a timely manner. Thus, the key challenge is to find an efficient approach that generates a *fairly tight* upper-bound estimate $l_{\max}(Q_{\mathrm{H}})$.

Finally, to address the third challenge, we consider a critical information-theoretic transformation as follows:

$$l_{\max}(\mathrm{q}|Q_{\mathrm{H}}) \geq \max_i \frac{I(x_i; \{Q_{\mathrm{H}}, q\}) - I(x_i; Q_{\mathrm{H}})}{H(x_i)} \tag{6}$$

$$= \max_i \frac{H(x_i|Q_{\mathrm{H}}) - H(x|Q_{\mathrm{H}}, q)}{H(x_i)} \tag{7}$$

$$= \max_i \frac{I(x_i; q|Q_{\mathrm{H}})}{H(x_i)} \tag{8}$$

$$= \max_i \frac{I(q; x_i|Q_{\mathrm{H}})}{H(x_i)} \tag{9}$$

$$= \max_i \frac{H(q|Q_{\mathrm{H}}) - H(q|Q_{\mathrm{H}}, x_i)}{H(x_i)}. \tag{10}$$

Intuitively, $H(q|Q_{\mathrm{H}}) - H(q|Q_{\mathrm{H}}, x_{\mathrm{i}})$ in the equation is the additional amount of information about q that can be derived from x_i while given Q_{H} as pre-knowledge. This transformation enables us to change our view from considering how much information about x_i is disclosed by q to how much information about q can be derived from x_i. This helps us consider different types of queries independently. After the transformation, the computation of $l_{\max}(q|Q_{\mathrm{H}})$ can be restated as follows:

Given the current query history Q_H, how much additional information about q can an adversary infer if it knows one more x_i as pre-knowledge?

As we will show later, the restated problem allows us to efficiently derive an upper bound $l_{\max}(q|Q_{\mathrm{H}})$ without computing $I(x_{\mathrm{i}}; Q_{\mathrm{H}} \cup \{q\})$ for every x_i.

In summary, in order to address the three challenges, we have transformed the problem to the efficient computation of

$$l_{\max}(\mathrm{q}|Q_{\mathrm{H}}) \geq \max_{i} \frac{H(q|Q_{\mathrm{H}}) - H(q|Q_{\mathrm{H}}, x_i)}{H(x_i)}. \tag{11}$$

As such, the baseline workflow for the SDB can be stated as follows: The SDB always monitors the cumulative sum of all historic $l_{\max}(q|Q_{\mathrm{H}})$ for answered queries. Let such cumulative sum be $l_{\max}(Q_{\mathrm{H}})$. When the SDB receives a new query q_{j}, it will compute $l_{\max}(q_j|Q_{\mathrm{H}})$, and then determine whether $l_{\max}(Q_{\mathrm{H}}) + l_{\max}(q_j|Q_{\mathrm{H}}) \geq l_0$. If so, the SDB denies the query. Otherwise, it answers the query and updates $l_{\max}(Q_{\mathrm{H}})$ as $l_{\max}(Q_{\mathrm{H}}) + l_{\max}(q_j|Q_{\mathrm{H}})$.

CHAPTER 5

ENTROPY BASED AUDITING

We now describe our basic ideas of computing $l_{max}(q|Q_H)$ for various aggregate functions. In order to do so, we must address two important challenges:

- The first challenge is how to compute a tight bound $l_{max}(q|Q_H)$ for each (type of) aggregate function, such as MIN/MAX and SUM.
- The second challenge is how to deal with Q_H that includes a combination of queries with different aggregate (types of) functions. As we mentioned in the introduction, most existing work deals with one or one type of functions exclusively (e.g., SUM-only [2] or MIN/MAX-only [1]). We need to determine whether different types of queries can be considered separately in the computation of $l_{max}(q|Q_H)$.

For the purpose of this thesis, we divide aggregate functions as MIN/MAX and SUM. In the following, we will first describe the basic ideas for processing MIN/MAX and SUM queries separately (i.e., when the query history is formed exclusively by one type of queries). Then, we will show how to efficiently combine them to compute $l_{max}(q|Q_H)$ for a mixture of MIN/MAX and SUM queries. The detailed computation will be presented in the query auditing algorithms later.

5.1. Auditing MIN/MAX Queries - Direct Usage of Information Entropy as Disclosure Bound

MIN-like functions include MIN and MAX. An important property is that, for query size $|q| \gg 1$, there is $H(q) \ll H(x_i)$. Consider a SDB where x_i is a Boolean attribute with i.i.d. uniform distribution. We have $H(x_i) = \log(2) = 1$, where $\log(\cdot)$ is logarithm with base 2. A query q, which is the MIN of 100 such data points satisfies

$$H(q) = -\frac{1}{2^{|q|}} \cdot \log(\frac{1}{2^{|q|}}) + \left(1 - \frac{1}{2^{|q|}}\right) \cdot \log\left(1 - \frac{1}{2^{|q|}}\right) \approx \frac{100}{2^{100}} \ll 1. \qquad (12)$$

Due to this property, we directly use $\max_i H(q)/H(x_i)$ as $l_{max}(q|Q_H)$ because

$$l_p(q|Q_H) = \max_i \frac{H(q|Q_H) - H(q|Q_H, x_i)}{H(x_i)} \leq \max_i \frac{H(q)}{H(x_i)} \qquad (13)$$

If we use this upper bound in the above example, the SDB can safely answer a large number ($2^{93} \cdot l_0$) of such MIN queries without privacy breach.

When the distribution of x_i is known, the computation of $H(q)$ and $H(x_i)$ is straight-forward. In particular, when x_i are i.i.d. distributed discrete variables,

$$H(q) = -\sum_{v \in \mathcal{V}} |q| \cdot f(v) \cdot (1 - F(v))^{|q|-1} \cdot \log(|q| \cdot f(v) \cdot (1 - F(v))^{|q|-1}) \qquad (14)$$

where ν is the value domain of x_i , $|q|$ is the number of data points included in q, and $f(v)=\Pr\{x_i = v\}$ and $F(v) = \Pr\{x_i < v\}$ are the probability density function (pdf) and cumulative density function (cdf) of x_i , respectively. When the distribution of x_i is unknown, the entropy can be computed by sampling x_i from the dataset. A simple approach is to count the histogram of q and x_i based on the samples.

There might be a question as, in the above example, what will happen if the answer to the MIN query is $q = 1$? The user can then infer that every cell covered by q has a value of 1. This is an example of the extreme-case privacy disclosure problem. We shall address this problem later.

5.2. Auditing SUM Queries - Finding Disjoint Independent Subsets

For SUM queries, the above upper bound cannot be adopted because there may be $H(q) > H(x_i)$. In the above example of Boolean i.i.d. data, a SUM query q over 100 tuples has $H(q) = \log(50\pi e)/2 \approx 4.37 > H(x_i)$. If $H(q)/H(x_i)$ is used as $l_{max}(q|Q_H)$, not even a single SUM query can be answered.

In order to audit a SUM query q, our basic idea is to count the maximum possible number of disjoint independent subsets of X which, given the previous query answers and any possible values of the other data points in X, can independently change the answer to q. We shall introduce in Chapter 6 the formal definition of an independent subset as well as an algorithm for counting the maximum number of disjoint ones (this algorithm uses existing exact-disclosure algorithms as a primitive). In the following, we first present an example of independent subset, and then describe how to compute an upper bound on $l_{max}(q|Q_H)$ based on the number of disjoint independent subsets.

Consider the case where the query history is empty. The SDB receives a query $q = \text{SUM}(x_1, \ldots, x_d)$. For any x_i ($i \in [1, d]$), given all other data points in X, changing x_i also changes q without influencing any query answer in Q_H (in this case,

$Q_H = \varnothing$). Thus, each $\{x_i\}$ ($i \in [1, d]$) is an independent subset. We can find d disjoint independent subsets $\{x_1\}, \ldots, \{x_d\}$.

However, the situation changes after a query, say $x_1 + x_2$, is answered. In this case, $\{x_1\}$ is no longer an independent subset because when x_1 changes, x_2 must also change in order to maintain the value of $x_1 + x_2$ in Q_H. Thus, we can find at most $d-1$ disjoint independent subsets $\{x_1, x_2\}, \{x_3\}, \ldots, \{x_d\}$.

We now describe our basic idea of computing an upper bound on $l_{max}(q|Q_H)$ based on the maximum number of disjoint independent subsets. Suppose that for a given query q and query history Q_H, $B_1, \ldots, B_k$ are disjoint independent subsets. Without loss of generality, we assume each B_i to be minimum - i.e., no subset of B_i is an independent subset. Let $q(B_i) = \text{SUM}(q \cap B_i)$.

Recall from Chapter 4 that key task for computing $l_{max}(q|Q_H)$ is to provide an upper bound on $H(q|Q_H) - H(q|Q_H, x_i)$. We separate the task into two parts: for computing a lower bound on $H(q|Q_H, x_i)$ and an upper bound on $H(q|Q_H)$, respectively.

For the first part, a critical observation is that one x_i can only reveal the value of $q(B_i)$ for a single B_i . All other independent subsets can still freely change the query answer. Thus, the problem is reduced to computing a lower bound on the entropy of the sum of k - 1 independent subsets given Q_H.

Without loss of generality, consider $H(q(B_1) + \cdots + q(B_{k-1})|Q_H)$. Note that for each minimum independent subset B_i, given Q_H and any possible values of the

other data points ($X \backslash B_i$), if a data point in B_i changes, then $q(B_i)$ as well as every other data point in B_i must change. Thus, we have $H(q(B_i)|Q_H) = H(x_i|Q_H) \geq (1-l_{max}(Q_H)) \cdot H(x_i)$. According to the well-known entropy power inequality [24] in information theory,

$$H(q(B_1) + \cdots + q(B_{k-1})|Q_H) \geq H(Z_1^* + \cdots + Z_{k-1}^*), \tag{15}$$

where Z^*_i are independent normal random variables with entropy $H(Z^*_i) = H(q(B_i)|Q_H)$. A normal variable with variance σ^2 has entropy $\ln(\sigma\sqrt{2\pi e})$, while the sum of normal variables with variances $\sigma^2_1, \ldots, \sigma^2_{k-1}$ has variance $\sum_{i=1}^{k-1} \sigma^2_i$. Thus, we can derive a lower bound on $H(q|Q_H, x_i)$ as

$$\begin{aligned} H(q|Q_H, x_i) &\geq H(q(B_1) + \cdots + q(B_{k-1})|Q_H) && (16)\\ &\geq H(Z_1^* + \cdots + Z_{k-1}^*) && (17)\\ &\geq \ln(\sqrt{k-1}) + (1 - l_{max}(Q_H)) \cdot \min_j H(x_j). && (18) \end{aligned}$$

We now consider the second part i.e., an upper bound on $H(q|Q_H)$. Note that without Q_H, q has $|q| - k$ additional independent subsets, each of which has entropy of $H(x_i)$ for a certain $x_i \in X$. Thus, again due to entropy power inequality,

$$e^{2H(q|Q_H)} \geq e^{2H(q|Q_H, x_i)} + e^{2\cdot(1-l_{max}(Q_H)\cdot H(x_i)}. \tag{19}$$

That is,

$$e^{2l_p(q|Q_H, x_i)} \geq 1 + e^{2\cdot(1-l_{max}(Q_H)\cdot H(x_i) - 2H(q|Q_H, x_i)} \tag{20}$$

$$H(q|Q_H) \leq \frac{1}{2} \cdot \ln(e^{2H(q)} - (|q| - k) \cdot e^{2\max_i H(x_i)}). \tag{21}$$

Similar to the case for auditing MIN/MAX queries, $H(q)$ and $H(x_i)$ can be computed either directly from the distribution of x_i or by sampling the dataset X if the distribution is unknown. According to (18) and (21), for a SUM query q, we can derive $l_{\max}(q|Q_H)$ as

$$l_{\max}(q|Q_H) = \frac{1}{2} \cdot \ln(e^{2H(q)} - (|q| - k) \cdot e^{2 \max_i H(x_i)}) - \ln(\sqrt{k-1}) - (1 - l_{\max}(Q_H)) \cdot \min_j H(x_j) \quad (22)$$

For example, when x_i follows i.i.d. normal distribution with entropy $H(x)$, we have $H(q) = \ln(|q|) + H(x)$. Thus,

$$l_{\max}(q|Q_H) = \frac{1}{2} \cdot \ln(2|q|k - k^2) + H(x) - \ln(\sqrt{k-1}) - (1 - l_{\max}(Q_H)) \cdot H(x) \quad (23)$$

5.3. Auditing Mixed Queries - A Simple Cumulative Sum

In the above two subsections, we assume the query history to consist exclusively of MIN/MAX and SUM queries, respectively. The information-theoretic framework also enables the processing of a mixture of MIN/MAX and SUM queries. Note that the computation of $l_{\max}(q|Q_H)$ for MIN/MAX queries (i.e., = $H(q)$) is unaffected by the types of Q_H. For SUM queries, the method is illustrated by the following theorem:

Theorem 1. *Given Q_{HS} and Q_{HM} as the set of SUM and MIN/MAX queries in the query history Q_H, respectively, $\forall q, x_i$,*

$$H(q|Q_{\mathrm{H}}) - H(q|x_i, Q_{\mathrm{H}}) \leq H(q|Q_{\mathrm{HS}}) - H(q|x_i, Q_{\mathrm{HS}}) + H(Q_{\mathrm{HM}}). \quad (24)$$

According to the theorem, while processing a SUM query q, we can first focus on the history of SUM queries and derive $l_{\max}(q|Q_{\mathrm{HS}})$ as discussed in Chapter 5.2. Then, we can compute $l_{\max}(q|Q_{\mathrm{H}}) = l_{\max}(q|Q_{\mathrm{HS}}) + H(Q_{\mathrm{HM}})$, where $H(Q_{\mathrm{HM}})$ is the cumulative sum of $H(q_{\mathrm{M}})$ for all answered MIN/MAX queries q_{M}.

CHAPTER 6

DETAILS OF SUM AUDITING

In this Chapter, we focus on the computation of $l_{\max}(q|Q_{\mathrm{HS}})$ for SUM queries. We first describe the algorithm to compute the number of independent subsets, and then prove the correctness of (22) discussed in Chapter 5.2.

6.1. Reduction to Exact-Disclosure Algorithms for Finding Independent Subsets

We first formally define an independent subset as follows.

Definition 5. *Given the history of SUM queries Q_{HS} and a SUM query q, $B \subseteq X$ is an independent subset iff for any value assignment of the data points $X = \langle v_1 , \ldots , v_n \rangle$, there exists another value assignment $X = v'_1 , \ldots , v'_n$ which satisfies all of the following three conditions.*

C1. $v'_{\mathrm{i}} = v_{\mathrm{i}}$ if $x_i \notin B$.

C2. $\sum_{xi \in q} v_i \neq \sum_{xi \in q} v'_i$.

C3. $\forall q' \in Q_{\mathrm{H}}$, $\sum_{xi \in q'} v_{\mathrm{i}} \neq \sum_{xi \in q'} v'_i = q'$.

According to the definition, each independent subset must be able to independently change q (C2) given the query history (C3) and the other data points (C1). This is consistent with the intuition discussed in subchapter 5.2.

We now show that any exact-disclosure auditing algorithm can be used to determine whether B is an independent subset. Let $Q_{\text{HS}} - X\backslash B$ be the result of removing all data points in $X\backslash B$ from the history of (answered) SUM queries. For example, if $Q_{\text{HS}} = \{x_1 + x_2 + x_3, x_2 + x_3 + x_4\}$, then $Q_{\text{HS}} - \{x_3\} = \{x_1 + x_2, x_2 + x_4\}$. Let x_∞ be a new data point, which has never been included in the SUM query history. Note that we can always construct such a data point even if all data points in X have been covered.

For a given query q and SUM query history Q_{HS}, we first construct a query $q' = \sum (B \cap q) + x_\infty$, and then construct a query history $Q'_{\text{HS}} = Q_{\text{HS}} - X\backslash B$. Then, we transform q' and Q'_{HS} to q'' and Q''_{HS} by replacing each data point $x_i \in X$ (but not x_∞) with the sum of two new data points $x^1_i + x^2_i$. We can now consider a new system with data points $X'' = \{x^1_1, x^2_1, x^1_2, x^2_2, \ldots, x^1_n, x^2_n, x_\infty\}$, SUM query history Q''_{HS}, and a SUM query q''. We have the following theorem:

Theorem 2. *(Reduction to Exact Disclosure) Given Q_{HS} and q, B is independent iff all data points in X'' are safe from exact disclosure given $\{q''\} \cup Q''_{HS}$ as query history.*

Please refer to Appendix C for the detailed proof. The intuition behind the proof is simple - Note that any x^1_i and x^2_i are safe from exact disclosure because they can freely change as long as x^1_i and x^2_i remain the same. Thus, X'' is safe iff x_∞ is safe. If B is independent, then given $Q_{\text{HS}} - X\backslash B$, $\sum(B \cap q)$ must still be able to freely change. As such, x_∞ cannot be derived from $\sum(B \cap q) + x_\infty$ and $Q_{\text{HS}} - X\backslash B$, and is

therefore safe from exact disclosure. On the other hand, if x_∞ is safe, then the value of $\sum(B \cap q)$ must be able to freely change given Q_{HS} and all data points in $X \backslash B$. Thus, B must be independent.

Algorithm 1 Finding Disjoint Independent Subsets

function COMP_k (Q_{HS}, q, $l_{\max}(Q_{\text{HS}})$)
  Replace each x_i in q and Q_{HS} by $x_i^1 + x_i^2$ to form q'' and Q''_{HS}, respectively.
  $X'' \leftarrow \{x_1^1, x_1^2, \ldots, x_n^1, x_n^2, x_\infty\}$. $k \leftarrow 0$.
  repeat
    $S \leftarrow \varnothing$. $X''_0 \leftarrow X''$. $Q''_0 \leftarrow Q''_{\text{HS}}$.
    for each $\{x_i^1, x_i^2\} \subseteq X''_0$ **do**
      if query $\sum(q \cap X'') - \{x_i^1, x_i^2\} + x_\infty$ is exact-safe given $Q''_{\text{HS}} - \{x_i^1, x_i^2\}$ **then**
        $S \leftarrow S \cup \{x_i\}$. $Q''_0 \leftarrow Q''_0 - \{x_i^1, x_i^2\}$. $X''_0 \leftarrow X''_0 \backslash \{x_i^1, x_i^2\}$
      end if
    end for
    if $S \neq \varnothing$ **or** query $\sum(q \cap X'') + x_\infty$ is exact-safe given Q''_{HS} **then**
      $X'' \leftarrow X'' \backslash X''_0$. $Q''_{\text{HS}} = Q''_{\text{HS}} - X''_0$. $k \leftarrow k + 1$.
    end if
  until $S = \varnothing$
  return k
end function

Theorem 2 provides an algorithmic solution for judging whether B is independent. We still need an efficient algorithm to construct an independent subset. This algorithm is enabled by the following *monotone* property of an independence set, which simply follows from its definition:

Theorem 3. *(Monotone) Given query history Q_H and a query q, if $B \subseteq \{x_1, \ldots, x_n\}$ is not independent, then none of its subsets is independent.*

Based on this property, we first construct a (minimal) independent subset B_1 by recursively removing a data point from X until none can be removed while keeping the remaining set to be independent. Then, to construct an independent

subset disjoint with B_1, we consider a new system with data points $X \backslash B_1$, query $q - B_1$, and query history $Q_{HS} - B_1$. We construct a minimal independent subset of the new system as B_2. By performing this process recursively, we obtain a number of disjoint independent subsets $B_1, \ldots, B_k$, and use k as the input to compute $l_{max}(q|Q_{HS})$. Nonetheless, k may not be the maximum possible number of independent subsets. We leave as an open problem how to compute the maximum k.

6.2. Computation of $l_{max}(q|Q_{HS})$

The following theorem summarizes our computation of $l_{max}(q|Q_{HS})$ (i.e., an upper bound on $l_p(q|Q_{HS})$) based on the number of independent subsets.

Theorem 4. *If $B_1, \ldots, B_k$ are disjoint independent subsets, then*

$$l_p(q|Q_{HS}) \leq \max_i \frac{H(q) - \ln(\sqrt{k-1})}{H(x_i)} - (1 - l_{max}(Q_{HS})). \quad (25)$$

Please refer to Appendix B for the proof. Similar to the computation of $l_{max}(q|Q_{HM})$ for MIN/MAX queries discussed in Chapter 5.1, $H(q)$ and $H(x_i)$ can be computed either directly from the distribution of x_i or by sampling the dataset X. The computation of k has been explained in Algorithm 1. The only other parameter, $l_{max}(Q_{HS})$, can be computed as the cumulative sum of $l_{max}(q|Q_{HS})$ for all SUM queries answered in the history.

According to Theorem 2, the greater k is, the smaller $l_{max}(q|Q_H)$ will be, and the more queries can be answered. Intuitively, this shows that if the query is consisted of a large number of independent components, then knowing only one data

point (and thereby one component) will not reveal much information about q, and vice versa. For a given k, the greater $l_{\max}(Q_{\mathrm{HS}})$ is, the greater $l_{\max}(q|Q_{\mathrm{HS}})$ will be. Intuitively, this shows that the more queries are answered in the history, the more information an adversary may extract from a given query.

CHAPTER 7

ALGORITHM ENTROPY-AUDITING

7.1. Detailed Algorithm

Algorithm 2 ENTROPY-AUDITING

1: $l_{\mathrm{HM}} \leftarrow 0$. $l_{\mathrm{HS}} \leftarrow 0$. $Q_{\mathrm{HM}} \leftarrow \varnothing$. $Q_{\mathrm{HS}} \leftarrow \varnothing$.
2: Wait until a query q is received.
3: **if** q $\in Q_{\mathrm{HM}} \cup Q_{\mathrm{HS}}$ **then**
4: Answer q precisely. ▷ Already answered before
5: **else**
6: **if** function of q is MIN or MAX **then**
7: $l_{\max} \leftarrow H(q)/H(x)$.
8: **else if** function of q is SUM **then**
9: $l_{\max} \leftarrow (H(q) - \ln(\sqrt{\textsc{Comp\_}k(Q_{\mathrm{HS}}, q, l_{\mathrm{HS}}) - 1}))/H(x) - (1 - l_{\mathrm{HS}}) + l_{\mathrm{HM}}$.
10: **end if**
11: **if** $l_{\max} + l_{\mathrm{HM}} + l_{\mathrm{HS}} \geq l_0$ **then**
12: Deny q.
13: **else**
14: Answer q precisely.
15: **if** function of q is MIN or MAX **then**
16: $l_{\mathrm{HM}} \leftarrow l_{\mathrm{HM}} + l_{\max}$. $Q_{\mathrm{HM}} \leftarrow Q_{\mathrm{HM}} \cup \{\mathrm{q}\}$.
17: **else**
18: $l_{\mathrm{HS}} \leftarrow l_{\mathrm{HS}} + l_{\max}$. $Q_{\mathrm{HS}} \leftarrow Q_{\mathrm{HS}} \cup \{\mathrm{q}\}$.
19: **end if**
20: **end if**
21: **end if**
22: Goto 2.

Algorithm 2 depicts ENTROPY-AUDITING, our query-auditing algorithm against partial disclosure. Once a query q is received, it is checked against the query history in Step 3, and answered if it has already been answered before. If not, $l_{\max}(q|Q_{\mathrm{H}})$ is computed as $l_{\max}$ in Steps 6 to 10. Step 9 calls Algorithm 1 to compute

the number of disjoint independent subsets for a SUM query. $H(x)$, l_{HM} and l_{HS} represent $\min_i H(x_i)$, $l_{max}(Q_{HM})$ and $l_{max}(Q_{HS})$, respectively. Step 11 determines whether l_0, the maximum acceptance level of privacy disclosure, could be violated if q were answered. If q is answered, Steps 15 to 19 updates the cumulative sum l_{HM}, l_{HS} and the query history.

7.2. Discussions on Privacy and Efficiency of Algorithm ENTROPY-AUDITING

[2] proposed the simulatable auditing model to prevent query denials from revealing private information. PARTIAL-AUDITING takes as input the query history Q_{HM} and Q_{HS}, the received query q, the cumulative sum $l_{max}(Q_{HM})$ and l_{HS} , as well as the distribution of x_i . All these inputs are either known or can be computed by an adversary. Thus, our algorithm satisfies the simulatable auditing model and will not disclose private information through query denials.

The computational complexity of PARTIAL-AUDITING depends on that of the underlying exact-SUM-auditing algorithm, which is invoked for $O(n \cdot |q|)$ times in Algorithm 2. The transparency of PARTIAL-AUDITING to the underlying exact-SUM- auditing algorithm provides a flexibly tradeoff between efficiency and query availability. Note that various existing algorithms for exact-SUM-auditing provide different levels of efficiency and query availability. For example, the algorithm which provides the optimal query availability uses Gaussian elimination which requires $O(n \cdot |Q_{HS}|)$ time, where $|Q_{HS}|$ is the number of answered SUM queries. The

query-size-restriction algorithm requires only O(1) time, but may deny queries that are actually exact-safe. Thus, by choosing which exact-SUM-auditing algorithm to use, the DBA of a SDB can adjust to the application-specific requirements on efficiency and query availability.

For any given exact-SUM-auditing algorithm, when there is a large number of data points (i.e., a large n), the efficiency of PARTIAL-AUDITING can be further improved by adjusting Steps 6 to 10 in the COMP_k function. Instead of removing each x_i from X_0 individually, we can attempt to remove multiple data points at a time. If r data points are removed together at every loop, the exact-SUM-auditing algorithm will only be invoked for $O(n \cdot |q|/r)$ times. However, the value of k may become larger, and the bound $l_{\max}(q|Q_{\mathrm{HS}})$ derived may not be as tight. Again, this allows the DBA to tradeoff between efficiency and query availability.

CHAPTER 8

SIMULATION RESULTS

As mentioned earlier, we compare our algorithm with the recent work in MIN/MAX and SUM queries i.e. the partial-disclosure algorithm proposed by Nabar et al [1] and the simulatable-auditing algorithm proposed by Kenthapadi et al [2]. We consider a database with a specific number of sensitive data points, a given width/range for each sensitive data point x_0 and a given number of queries.

8.1. Experimental Setup

1) *Hardware*: All the experiments were performed on a machine with Intel Core 2 Duo 1.6 GHz processor with 2 GB RAM and Windows Vista operating system. All the algorithms are implemented using C/C++ and Visual Studio.

2) *Datasets*: We consider a database consisting of a set of tuples having a private attribute. The value of the private attributes in the database follows i.i.d. normal distribution with mean of 0 and variance 1. In the initial test we issue 100 queries in a random order. Each query refers to a set of data points, which is generated randomly.

3) *Parameter Settings*: For our algorithm we consider an initial value of 0.5 for security threshold($l_0 \in [0, 1]$). For the partial-disclosure algorithm proposed by Nabar

et al [1], the parameter(λ, δ, γ) are assigned values 0.5,0.5 and 0.1 respectively. While for the simulatable-auditing algorithm proposed by Kenthapadi et al [2], the parameters(λ, α, δ) are assigned 0.5,0.3 and 0.5 respectively.

4) *Performance Measure*: For our experiments there are two measures of performance: number of answered queries compared to the total number of queries posted(query availability) and computation time.

8.2. Comparison with Existing Algorithms

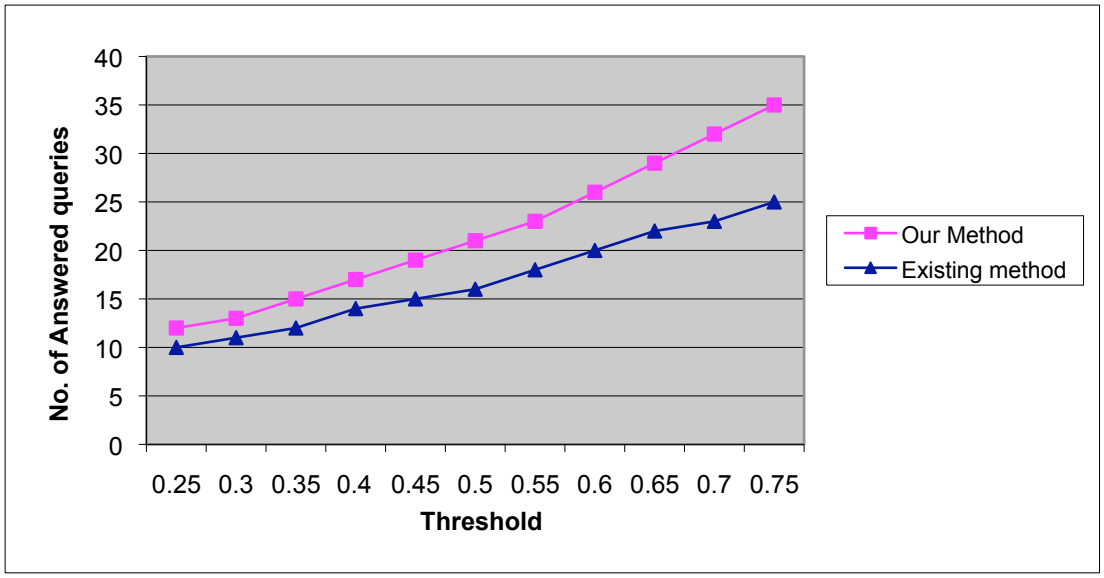

Figure 8.1: No. of queries answered by Existing and Our Method with change in security level.

We know that each query discloses some information about the sensitive data point it refers to. Higher the value of l_0, more information about each sensitive data point x_0 can be disclosed. Thus more queries are answered as the value of l_0 is

increased. Figure 1 shows the increase in number of queries answered with the increase in threshold.

In comparison to the partial-disclosure algorithm proposed by Nabar et al [1], our method performs better because it answers more queries. While the difference is not too much when the threshold value is low as very few queries can be answered anyways, the difference is significant as the threshold takes optimal value of around 0.5. Our algorithm sets a tight bound on the value of l_{max} for each query answered.

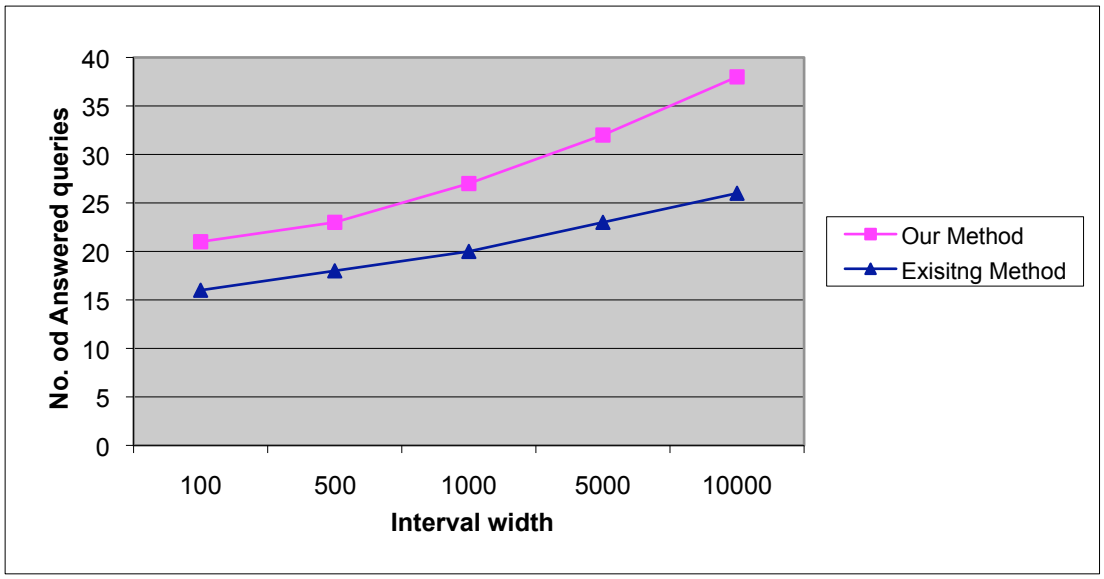

Figure 8.2: No. of queries answered by Existing and Our Method with change in Interval width.

When considering the interval width for each sensitive data point, we can see from Figure 2 that out method performs much better. While the partial-disclosure algorithm proposed by Nabar et al [1] requires the system to keep track of the interval width for each data point using predicates and update it after a query is answered, in our algorithm the entropy calculation of the query and data points takes

care of the interval width. The difference in the number of answered queries increases with the increase in the interval width.

All the above experiments consider 100 queries posted. When considering more than 100 queries, the percentage of answered queries remains almost the same. Figure 3 shows the change in number of answered queries for our method and the partial-disclosure algorithm proposed by Nabar et al [1] with the increase in total number of queries posted. The difference in number of answered queries significantly increases as the total number of queries posted increases.

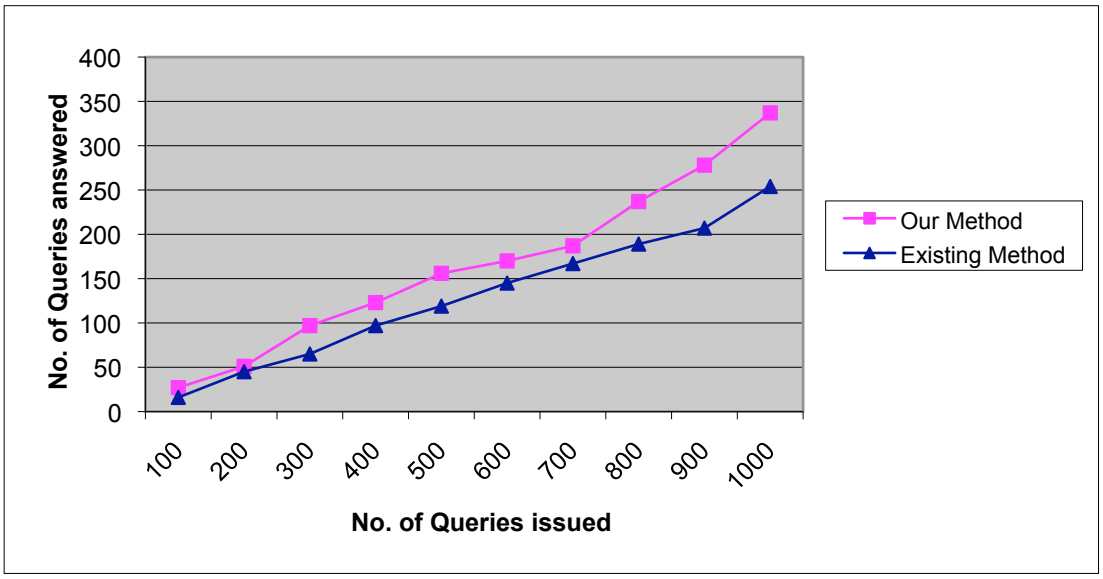

Figure 8.3: No. of queries answered by Existing and Our Method with change in no. of queries issued.

Considering the computation time for our algorithm and the partial-disclosure algorithm proposed by Nabar et al [1], Figure 4 shows the time taken to answer queries for different numbers of queries posted. The time taken increases with the increase in total number of queries issued.

The increase in computation is more gradual in our algorithm because of a consistent way to compute the updates as queries are answered. The partial-disclosure algorithm proposed by Nabar et al [1] takes more time to compute the updated predicates as the number of queries issued increases.

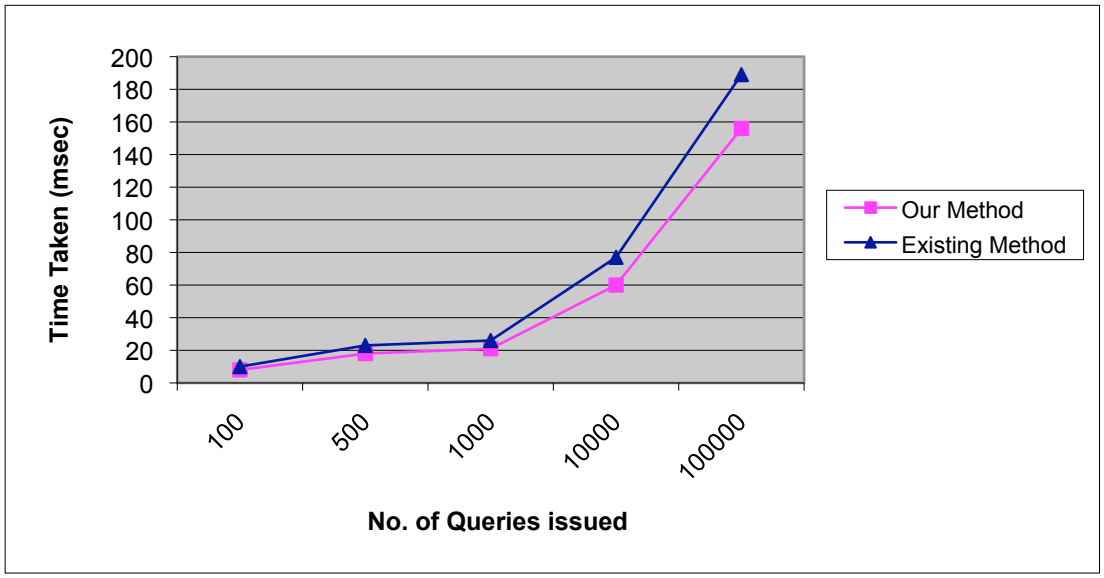

Figure 8.4: Time taken to answer queries with change in no. of queries issued.

When compared with the simulatable-auditing algorithm proposed by Kenthapadi et al [2], we find that our algorithm answers more queries. Figure 5 shows the comparison for the number of queries answered. Our algorithm performs better for all the values of l_p. We can increase the availability of aggregate queries by choosing different underlying exact-disclosure algorithms. Thus, we can have a flexible tradeoff between the efficiency of query auditing and the availability of aggregate queries.

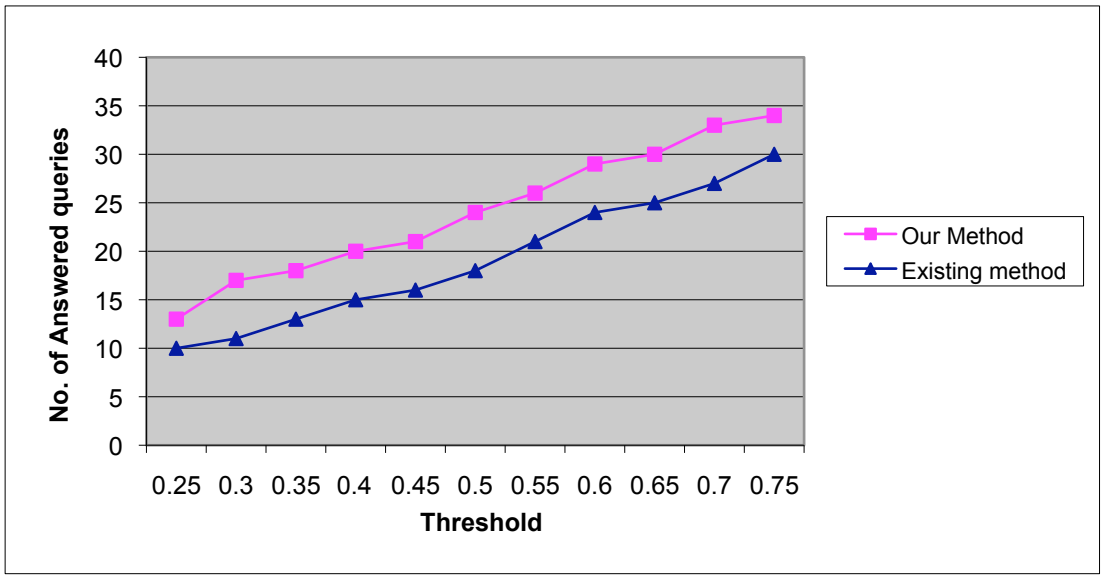

Figure 8.5: No. of queries answered by Existing and Our Method with change in security level l_0.

Now considering the interval width of each sensitive data point, we can see from Figure 6 that the increase in interval width also increases the number of answered queries. The entropy of a sensitive data point increases as we increase its interval width due to increase in randomness of the data point.

Figure 6 also shows the result for the simulatable-auditing algorithm proposed by Kenthapadi et al [2]. It is quite clear that our algorithm performs much better even when there is a change in interval width of the data points.

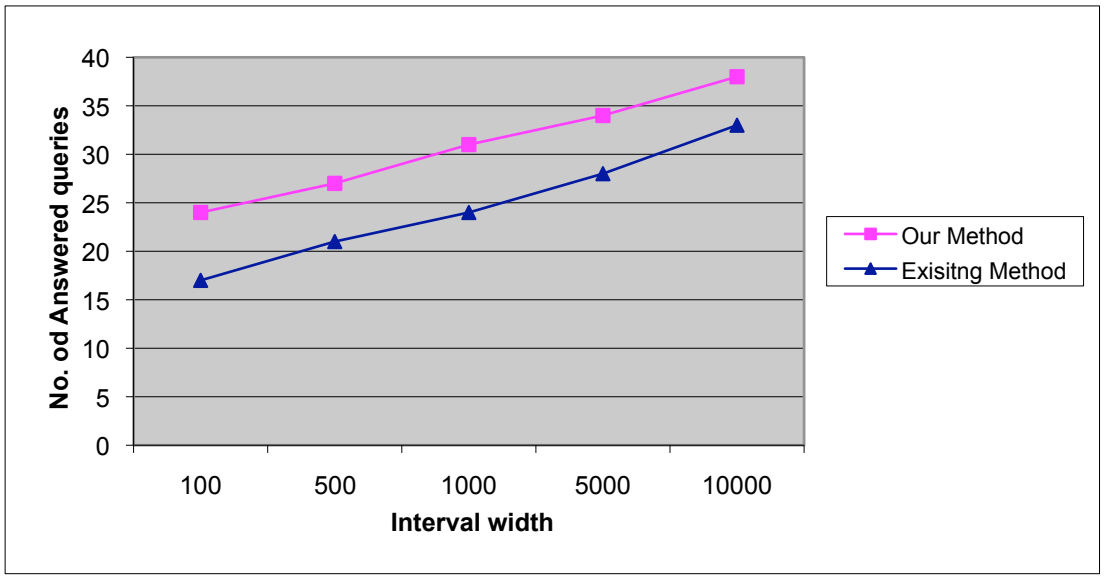

Figure 8.6: No. of queries answered by Existing and Our Method with change in Interval width.

The computation time for our algorithm specifically for SUM queries depends on the exact-SUM-auditing algorithm. Considering the efficient Gaussian elimination method the computation time is shown in Figure 7. The computation time increases gradually for the increase in number of queries posted. This is due to the efficient method to calculate the updated l_{max} using maximum independent subset algorithm. The computation time of the simulatable-auditing algorithm proposed by Kenthapadi et al is also shown in Figure 7 for comparison. Compared to our algorithm, the simulatable-auditing algorithm proposed by Kenthapadi et al takes more time. Their computation time depends on the calculation of the volume for the polytope consisting of the queries. This calculation usually takes time.

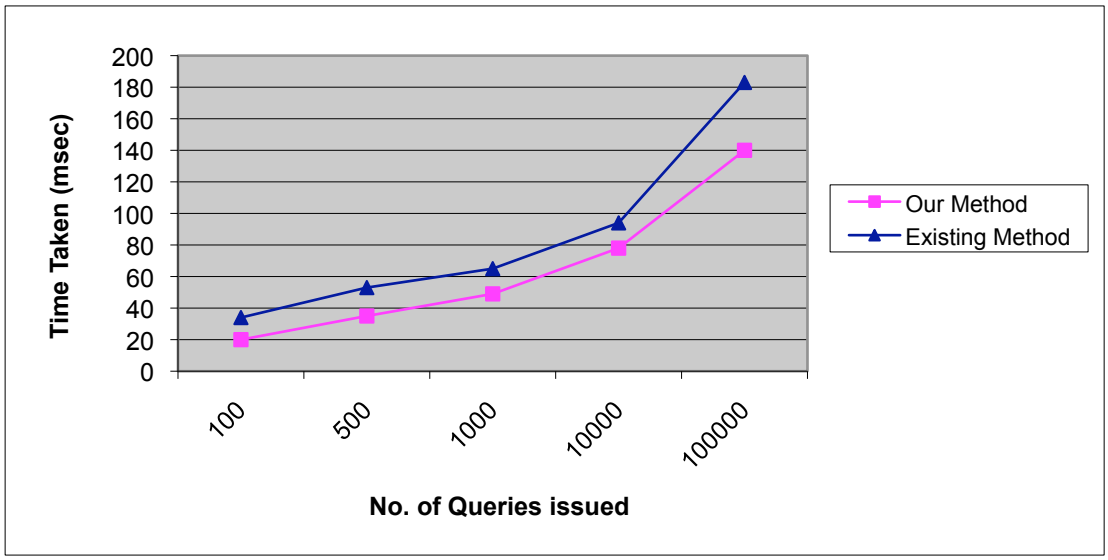

Figure 8.7: Time taken to answer queries with change in no. of queries issued.

Overall there is a significant increase in the performance and efficiency in our method when compared to the existing methods.

8.3. Analysis of ENTROPY-AUDITING

We also evaluated the algorithm when 1) all queries are SUM-like, 2) half of the queries are SUM-like, while the other half are MIN-like, and 3) all queries are MIN-like. We issue 100 randomly generated queries. The query refers to 100 data points in the database, which is also generated randomly. For the case of half SUM-like and half MIN-like queries we randomly select the aggregate function. Figure 8 shows the results for the 3 simulation tests.

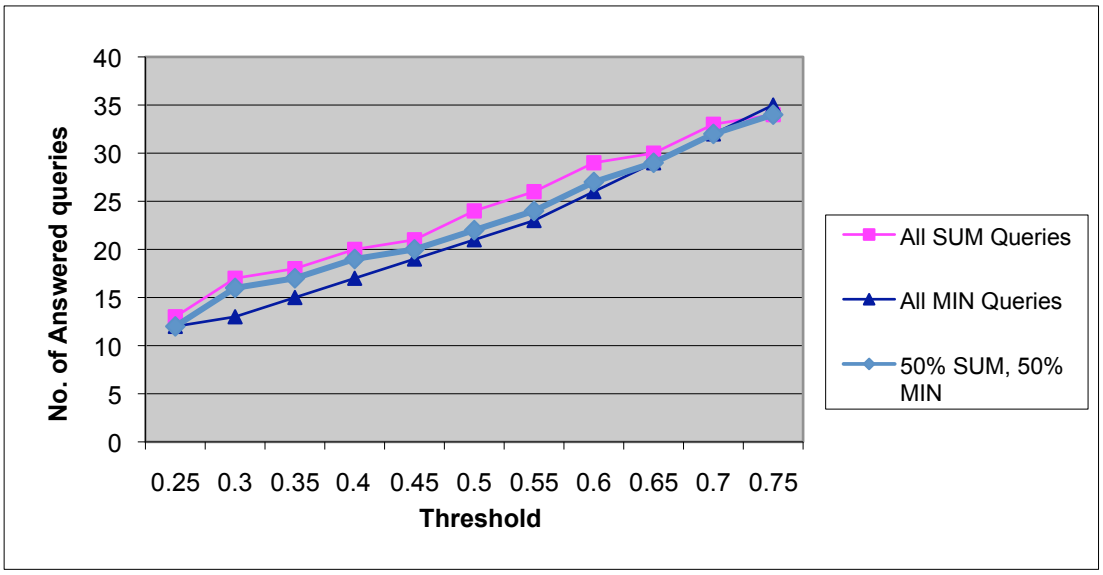

Figure 8.8: Change in no. of queries answered with change in security level for queries with different types aggregate functions on 100 data points.

More queries are answered as the value of l_0 is increased. As we can see, Algorithm 2 answers more SUM-like queries than MIN-like queries when the threshold value is low. The difference is lesser when the value of l_0 is higher. For example, for a high value of l_0=0.75 about 34 queries are answered out of 100 queries issued for both SUM-like and MIN-like aggregate functions. The results for 50% SUM-like queries and 50% MIN-like queries also show that our algorithm can efficiently support a combination of SUM and MIN/MAX queries in query auditing against partial disclosure.

When considering a database with 500 data points, we get the results shown in Figure 9 for the 3 simulation tests mentioned above. 100 queries are generated randomly. The set of data points that the queries refer to is also generated randomly.

For the case of half SUM-like and half MIN-like queries we randomly select the aggregate function.

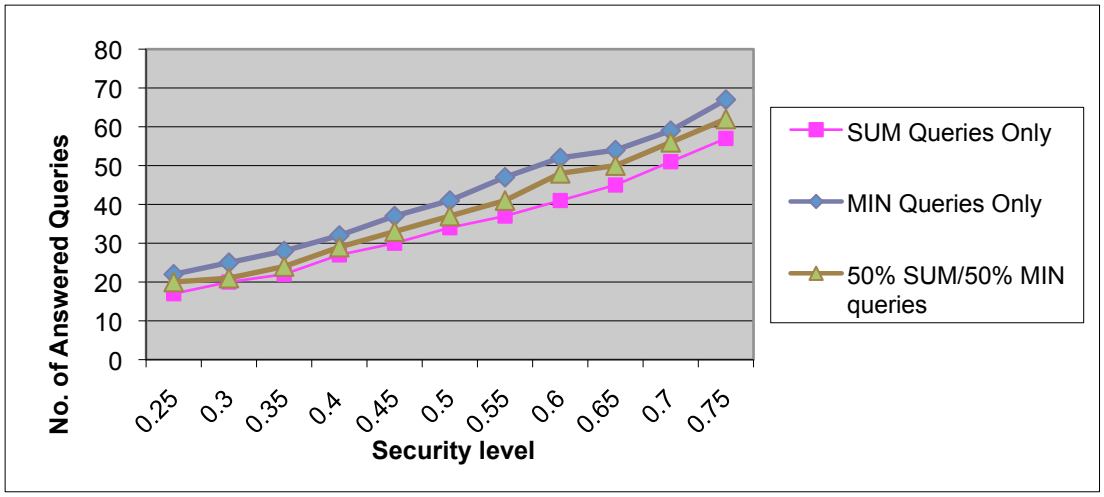

Figure 8.9: Change in no. of queries answered with change in security level for queries with different types aggregate functions on 500 data points.

An interesting observation from the results in Figure 8 and 9 is that the number of MIN-like queries that are answered increases significantly with the increase in the number of data points that the queries refers to. This increase is lesser in case of SUM-like queries. Thus the MIN-like queries depend more on the number of data points that the queries refer to.

CHAPTER 9

FINAL REMARKS

In this thesis, we introduce the concept of ENTROPY-AUDITING for query auditing against partial disclosure in statistical databases. We define the privacy breach according to a partial disclosure measure: i.e., breach occurs iff the amount of disclosure about x_i exceeds a pre-determined threshold. In particular, the auditing algorithm supports queries with MIN-like (e.g., MIN and MAX) and SUM-like (e.g., SUM and MEDIAN) aggregate functions. We proposed the concept of ENTROPY-AUDITING with two main ideas: (a) using the information theory framework to transform how much information about an individual tuple an aggregate query discloses to how much information about an aggregate query can be inferred from an individual tuple, and (b) the usage of existing exact-disclosure SUM-auditing algorithms as a primitive function to compute the amount of information about an aggregate query answer that can be inferred from an individual tuple.

The transformation helps us consider different types of aggregate function separately in query auditing and thus, support combination of queries with different types of aggregate functions. For MIN-like, we calculate the entropy of a given query based on the distribution of the data points and use it to calculate the updated

security threshold. This provides an efficient method to check whether the given query is safe to answer based on the query history.

For SUM-like queries, we use the concept of maximum disjoint subset for calculating the updated security threshold. Choosing different underlying exact-disclosure algorithms for calculating the maximum disjoint subset, allows a flexible tradeoff between the efficiency of query auditing and the availability of aggregate queries.

Our experimental results demonstrate the superiority of our algorithm over existing algorithms. The results also show that our algorithm supports combination of queries with different types of aggregate functions. Our algorithm is the first to support a combination of SUM and MIN/MAX queries in query auditing against partial disclosure.

In the future, we will apply the concept of ENTROPY-AUDITING to other types of queries like SQL queries. We will also consider other types of disclosures that are tough to identify.

APPENDIX A

PROOF OF LEMMA

Lemma 1. (*Upper Bound*)

$$H(q_B|Q_{HS}) \leq H(x_i) = \ln(\sigma\sqrt{2\pi e}). \quad (27)$$

NOTE: for all possible values of $X\backslash B$

Proof. In order to prove the lemma, we will show that given $x_i \in B$, for all V, V' that satisfy C1 and C3, C2 is satisfied iff $v_i \neq v'_i$. It is easy to see that if this condition holds, then given Q_H and $X\backslash B$, q changes iff x changes. Thus, (??) always holds.

Necessity: If C2 is satisfied and $\exists x_i \in B$ with $v_i = v'_i$, then $B\backslash x_i$ is independent because V and V' satisfy C1 - C3 for $B\backslash x_i$. This contradicts our assumption that B is minimal independent. Thus, when C2 is satisfied, $v_i \neq v'_i$ for all $x_i \in B$.

Sufficiency: Suppose that $\sum_{xj \in q} v_j = \sum_{xj \in q} v'_j$ while $v_i \neq v'_i$. Since B is independent and the data points have no range limit, it is always possible to find $V^0 = \langle v^0_1 , \ldots , v^0_n \rangle$ such that V and V^0 satisfy C1 - C3. We construct $V^c = \langle v^c_1 , \ldots , v^c_n \rangle$ as

$$v_j^c = v_j + (v'_j - v_j) \cdot \frac{v_i^0 - v_i}{v'_i - v_i}. \quad (28)$$

for all $j \in [1, n]$. Consider V^0 and V^c . Note that $\forall x_j \notin B$, $v_j = v'_j = v^0_j$. According to (35), $v^c_j = v_j = v^0_j$. Also note that $v^c_i = v^0_i$. Thus, for a subset $B' = B\backslash x_i$, V^0 and V^c satisfy C1. Since $\sum_{xj \in q} v = \sum_{xj \in q} v'$,

$$\sum_{x_j \in q} v_j^c = \sum_{x_j \in q} v_j + \left(\sum_{x_j \in q} v'_j - \sum_{x_j \in q} v_j \right) \cdot \frac{v_i^0 - v_i}{v'_i - v_i} = \sum_{x_j \in q} v_j \neq \sum_{x_j \in q} v_j^0. \quad (29)$$

Thus, V^0 and V^c satisfy C2. V^0 and V^c also satisfy C3 due to the definition of V^c. Thus, $B' = B \backslash x_i$ is independent. This contradicts our assumption that B is minimal independent. Thus, if $\exists x_i \in B$ with $v_i \neq v'_i$, then C2 must hold for V and V'.

APPENDIX B

PROOF OF THEOREM 4

Proof. Let $q_i = \sum (B_i \cap q)$. Let $B_0 = X \setminus (B_1 \cup \cdots \cup B_k)$. We have

$$H(q|Q_H) = H(q_1 + \cdots + q_k + \sum (B_0 \cap q)|Q_H) \geq H(q_1 + \cdots + q_k|Q_H, B_0). \quad (30')$$

We first derive an lower bound on $H(q_1 + \cdots + q_k | Q_H, B_0)$. Without loss of generality, suppose that $x_1, \ldots, x_k$ belong to $B_1, \ldots, B_k$, respectively. According to Lemma 1, for all $i \in [1, k]$ and all possible values of $X \setminus B_i$,

$H(q_i | X \setminus B_i, Q_H) = H(x_i | X \setminus B_i, Q_H)$. Thus, $H(q_i | Q_H) = H(x_i | Q_H)$. Since $H(x_i | Q_H) \geq l_H$, we have $H(q_i | Q_H) \geq l_H$. Let $y_1, \ldots, y_k$ be k random variables generated i.i.d. from Gaussian distribution with variance of $e^{2/H}/(2\pi e)$. Since the entropy of Gaussian distribution with variance σ^2 is $\log(\sigma\sqrt{2\pi e})$ where $\log(\cdot)$ is the natural logarithm, we have $H(y) = 1$ and

$$H(y_1 + \cdots + y_k) = \log \sqrt{k} + l_H. \quad (31)$$

Note that all constraints imposed by Q_H on the values of q_i ($I \in [1, k]$) can be expressed as linear equations. Also note that with $x_1, \ldots, x_n$ having Gaussian distribution, the posterior distribution of q_i given these linear equations is also Gaussian. Since $H(x_i | Q_H) \geq l_H$ for all $i \in [1, n]$, if $H(q_1 + \cdots + q_k | Q_H, B_0) <$ $H(y_1 + \cdots + y_k)$, then given Q_H and B_0, there must exist a linear equation

$$q_1 + \cdots + q_k = a_1 x_{i_1} + \cdots + a_j x_{i_h} + c \quad (32)$$

with $h < k$ and $i_1, \ldots, i_h \in [1, n]$. Note that when (39) holds, there is at least one q_j ($j \in [1, h]$) that does not contain any of $x_{i1}, \ldots, x_{ih}$. That is, q_j can be derived from

$X \backslash B_j$ based on (39). This contradicts our assumption that B_j is independent (C2). Thus, $H(q_1 + \cdots + q_k \,|Q_H, B_0) \geq \log \sqrt{k} + l_H$.

We now derive an upper bound on $l(q)$. As we can see, given any single data point x_i ($i \in [1, n]$), at most one of the k minimal independent subsets can be determined. Similar to the steps above, we can prove that

$$H(q|x, Q_H) \geq H(y_1 + \cdots + y_{k-1}) = \log \sqrt{k-1} + l_H. \tag{33}$$

Suppose that given x and Q_H, the variance of q is σ_0^2. Since $H(q|x, Q_H) = \log(\sigma\sqrt{2\pi e})$,

$$\sigma_0 \geq \frac{l_H \sqrt{k-1}}{\sqrt{2\pi e}}. \tag{34}$$

Since $\max_{i \in [1,k]} |B_i| = s$, the variance of q given (only) Q_H is at most $\sigma^2_0 + s\sigma^2$. Thus,

$$l(q) = \log \frac{\sqrt{\sigma_0^2 + s\sigma^2}}{\sigma_0} \leq \frac{1}{2} \log \left(1 + \frac{2\pi e s \sigma_x^2}{l_H^2 (k-1)}\right). \tag{35}$$

APPENDIX C

PROOF OF THEOREM 2

Proof. *Necessity*: If B is independent, then there are V and V' that satisfy C1 - C3. Recall that in the proof of Lemma 1 we have proved that given $X\backslash B$, $v_i \neq v'_i$ for all $x_i \in B$. Also note that x_∞ never appears in Q_H. Thus, no $x_i \in B$ can be derived from $Q_H - X\backslash B$ and $\sum (B \cap q) + x_\infty$. As such, the only possible way for $q_B + x_\infty$ to be exact-unsafe is if x_∞ can be compromised. That is, q_B can be derived from $Q_H - X\backslash B$. This is impossible because V and V' satisfy C2 (i.e., $\sum_{x_i \in (B \cap q)} v_i \neq \sum_{x_i \in (B \cap q)} v'_i$).

Sufficiency: If $\sum_{x_i \in (B \cap q)} x_i + x_\infty$ is exact-safe, given $Q_H - X\backslash B$, $\sum_{x_i \in (B \cap q)} x_i$ must have at least two possible values because otherwise x_∞ can be inferred from $Q_H - X\backslash B$. Note that no data point $x_i \in (X \backslash B)$ appears in $Q_H - X\backslash B$. Thus, given $X\backslash B$, there must exist V and V' that satisfy C1 - C3. That is, B is independent.

REFERENCES

[1] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards robustness in query auditing. s.l. : VLDB, 2006, pp. 151–162.

[2] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. s.l. : PODS, 2006, pp. 118-127.

[3] David Dobkin, Anita K. Jones, Richard J. Lipton. Secure Databases: Protection Against User Influence. s.l. : ACM, 1979, pp. 97-106.

[4] Ljiljana Brankovic, Peter Norak, Mirka Miller, Graham Wrightson. Usability of Compromise-Free Statistical Databases. s.l. : IEEE Computer Society, 1997, pp. 144-154.

[5] N. R. Adam, and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. s.l. : ACM Computing Surveys, 1989, pp. 515-556.

[6] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. s.l. : SIGMOD, 2005, pp. 251– 262.

[7] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. s.l. : PODS, 2000, pp. 86-91.

[8] F. Y. Chin, G. Ozsoyoglu. Auditing and Inference Control in Statistical Databases. s.l. : IEEE press, 1982, pp. 574-582.

[9] F. Y. Chin. Security problems on inference control for SUM, MAX, and MIN queries. s.l. : ACM, 1986, pp. 451 - 464.

[10] I. Dinur, K. Nissim. Revealing information while preserving privacy. s.l. : PODS, 2003, pp. 202-210.

[11] Cynthia Dwork, Kobbi Nissim. Privacy-Preserving Datamining on Vertically Partitioned Databases. *24th Annual International Cryptology Conference (CRYPTO 2004)*. s.l. : Springer, 2004, Vol. 3152, pp. 528-544.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. s.l. : Springer, 2006, pp. 265–284.

[13] Martin Dyer, Alan Frieze, Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. s.l. : ACM, 1991, Vol. 38, pp. 1–17.

[14] F. Chin, and G. Osoyoglu. Auditing and Inference Control in Statistical Databases. s.l. : IEEE, 1982, pp. 574-582.

[15] L. Beck. A security machanism for statistical database. ACM Trans. Database Syst., 5(3): pp. 316–3338, 1980.

[16] M. Cryan and M. Dyer. A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. ACM Press, 2002, pp 240–249.

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

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

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

[20] R. Agrawal and R. Srikant. Privacy-preserving Data Mining. In Proc. of ACM SIGMOD, pp. 439–450, 2000

[21] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. In PODS, pp 211–222, 2003.

[22] F. Chin, G. Osoyoglu. Security in Partitioned Dynamic Statistical Databases, Proc. IEEE COMPSAC, pp. 594–601.

[23] T. Cormen, C. Leiserson, R. Rivest. Introduction to Algorithms, McGraw-Hill, Boston, 1990.

[24] Thomas M. Cover, Joy A. Thomas. Elements of Information Theory, Second Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 1991.

[25] A. Friedman, L. Hoffman. Towards a Fail-safe Approach to Security and Privacy, Proc. IEEE Symp. on Security and Privacy, 1980.

[26] J. Traub, Y. Yemini, H. Wozniakowksi. The Statistical Security of a Statistical Database, ACM TODS, 9, 4 pp. 672–679, 1984.

[27] C. K. Liew, U. J. Choi, and C. J. Liew. A data distortion by probability distribution. ACM Transactions on Database Systems, 10(3), 1985.

[28] K. Hoffman and R. Kunze. Linear algebra. Prentice-Hall Inc, 1971.

[29] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In Proc. Advances in Cryptology – EUROCRYPT 2004, 2004.

[30] Y. Lindell and B. Pinkas. Privacy preserving data mining. In CRYPTO, 2000.

[31] Y. Li, H. Lu, and R. H. Deng. Practical inference control for data cubes (extended abstract). in Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006, pp. 115–120.

[32] L. Wang, S. Jajodia, and D. Wijesekera. Securing OLAP data cubes against privacy breaches. in Proceedings of the 25th IEEE Symposium on Security and Privacy, 2004, pp. 161–175.

[33] L. Wang, Y. Li, D. Wijesekera, and S. Jajodia. Precisely answering multi-dimensional range queries without privacy breaches. in Proceedings of the 8th European Symposium on Research in Computer Security, 2003, pp. 100–115.

[34] L. Wang, D. Wijesekera, and S. Jajodia. Cardinality-based inference control in sum-only data cubes. in Proceedings of the 7th European Symposium on Research in Computer Security, 2002, pp. 55–71.

[35] L. Wang, D. Wijesekera, and S. Jajodia. Cardinality-based inference control in data cubes. Journal of Computer Security, vol. 12, no. 5, pp. 655 – 692, 2004.

[36] N. Zhang, W. Zhao, and J. Chen. Cardinality-based inference control in OLAP systems: An information theoretic approach. in Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP, 2004, pp. 59–64.

BIOGRAPHICAL INFORMATION

Mayur Motgi was born May 1985, India. He received his Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University, India in June 2006.

Prior to pursuing his Masters he worked at Oracle India Pvt. Ltd., Bangalore, India in the field of Enterprise Resource Planning (ERP) for a period of one year. In Fall of 2007, he started his graduate studies in computer science. He received his Masters in Computer Science from the University of Texas at Arlington, in May 2009. His research interests include databases, computer networks and robotics.