Privacy-Preserving OLAP: An Information-Theoretic Approach

Nan Zhang, *Member, IEEE,* Wei Zhao, *Fellow, IEEE*

Abstract—We address issues related to the protection of private information in Online Analytical Processing (OLAP) systems, where a major privacy concern is the adversarial inference of private information from OLAP query answers. Most previous work on privacy-preserving OLAP focuses on a single aggregate function and/or addresses only exact disclosure, which eliminates from consideration an important class of privacy breaches where partial information, but not exact values, of private data is disclosed (i.e., partial disclosure). We address privacy protection against both exact and partial disclosure in OLAP systems with mixed aggregate functions. In particular, we propose an information-theoretic inference control approach that supports a combination of common aggregate functions (e.g., COUNT, SUM, MIN, MAX, MEDIAN) and guarantees the level of privacy disclosure not to exceed thresholds predetermined by the data owners. We demonstrate that our approach is efficient and can be implemented in existing OLAP systems with little modification. It also satisfies the simulatable auditing model and leaks no private information through query rejections. Through performance analysis, we show that compared with previous approaches, our approach provides more effective privacy protection while maintaining a higher level of query-answer availability.

Index Terms—Online Analytical Processing (OLAP), privacy, information theory.

1 Introduction

Online analytical processing (OLAP) is one of the most popular decision support and knowledge discovery techniques in business-intelligence systems [20]. However, it is a challenge to enable OLAP on private data without violating the data owners' privacy. Traditional studies on database security provide access control and data sanitization (e.g., removal of personal-identifiable information) tools for the relational back-end of OLAP systems [19], [21], [25], [31]. Nonetheless, since an essential feature of OLAP is to compute the multi-dimensional aggregates of stored data, a major privacy concern in OLAP is the adversarial inference of (individual) private data points from the aggregate information. This inference problem cannot be fully addressed by access control and data sanitization techniques.

In this paper, we address privacy protection against adversarial inference in an OLAP system consisting of a data warehouse server and a number of users. The data warehouse server holds private data, and is supposed to answer OLAP queries [20] issued by users on the multi-dimensional aggregates of private data. A user may not have the right to access all individual data points in the data warehouse, but might be allowed to issue OLAP queries on the aggregates of data for which it has no right to access. For example, in a hospital system, the accounting department (as a user) can access each patient's financial data, but not the patients' medical records (an actual HIPAA requirement [33]). Nonetheless, the accounting department may query aggregate information related to the medical records, such as the total expense for patients with Alzheimer's disease.

In an OLAP system, a privacy breach occurs if a user can infer certain information about a private data point for which it has no right to access from the query answers it receives as well as the data that it has the right to access. Such privacy breach is referred to as the inference problem [9]. Protection against the inference problem has been extensively investigated in the literature of related areas (e.g., statistical databases [1], [9], [10], [16], [22], query auditing [13], [23], [24], [29], [30]), which laid solid foundation for studies in OLAP. Privacy protection in OLAP faces unique challenges, such as larger queries, mixed aggregate functions, and increased demand for shorter response time [34].

There are two types of methods that have been proposed to prevent inference problems from happening in OLAP systems: inference control (i.e., online query auditing in OLAP) [26], [34], [35], [37] and input/output perturbation [4], [16], [32]. With the inference control approach, after receiving a query from a user, the data warehouse server determines whether answering the query may lead to an inference problem, and then either rejects the query or answers it precisely. The input/output perturbation approach either perturbs (input) data stored in the data warehouse server with random noise and answers every query with an estimation, or adds random noise to the (output) query answers in order to preserve privacy. Existing studies support COUNT [4] and SUM [32] queries for input perturbation, and general aggregation functions for output perturbation [16].

- *Nan Zhang is with the Department of Computer Science, The George Washington University, Washington, DC 20052, USA. Email: nzhang10@gwu.edu.*
- *Wei Zhao is with The University of Macau, Taipa, Macau SAR, China. Email: weizhao@umac.mo.*

Digital Object Indentifier 10.1109/TKDE.2010.25 1041-4347/10/$26.00 © 2010 IEEE

While both approaches can be very useful for privacy protection, precise query answers (without perturbation) are often preferred when important decisions need to be made based on the data [29]. Thus, we focus on the inference control approach in this paper[1]. Most existing solutions for inference control in OLAP share an assumption that a privacy breach occurs if and only if a user can infer the *precise* value of a private data point stored in the data warehouse server (i.e., exact disclosure) [35], [37]. This assumption eliminates from consideration an important class of privacy breaches where *partial* information of a private data point is disclosed (i.e., partial disclosure), and, as is demonstrated by previous work [5], [22], [23], [26], [28], [29], does not suffice for many practical applications. For example, although a malicious user cannot infer the precise age of a person, it may be able to estimate the age within error of one year. In practice, many people would consider this to be a violation of their privacy. As such, existing solutions for inference control in OLAP cannot satisfy the privacy-protection requirements of many real-world systems.

In this paper, we address privacy protection against both exact and partial disclosure in OLAP systems. We first define a continuous measure on the partial disclosure of privacy information, which can be used (with precautions) by the data owners to specify their maximum acceptable (or tolerable) level of privacy disclosure. Then, we propose an inference control approach based on an information-theoretic formulation of OLAP queries, which quantitatively measures the amount of private information disclosed by each query. Basically, our approach maintains an estimated upper bound on the amount of private information disclosed by all answered queries, and rejects a query iff such an estimation would exceed the owner-specified threshold. We shall demonstrate that our approach has the following features to distinguish itself from previous approaches:

- By exploiting the common information-theoretic properties of various aggregate functions, we address a combination of aggregation functions including the most common ones such as SUM, AVG, COUNT, MIN, MAX, etc, in contrast to the SUM-only [7], [26], [35], [36], [37], COUNT-only [4], or MIN/MAX-only [29] approaches in previous work.
- Our approach is effective against both exact and partial disclosure. To the best of our knowledge, our approach is the first to guarantee the level of privacy disclosure not to exceed owner-specified thresholds in OLAP systems with queries of mixed aggregate functions. Furthermore, our inference control algorithm satisfies the simulatable auditing model [23] and leaks no private information through rejections.
- While providing rigid privacy protection, our approach also maintains a high level of query availability. In particular, we derive theoretical lower bounds on the percentage of queries answered by our approach, and use simulation results to show that our approach can answer substantially more queries than existing approaches while providing better privacy protection.

1. Our proposed scheme can also be seamlessly integrated with the input/output perturbation approach for scenarios where the input data or the query answers can be perturbed to improve the performance of both approaches. Due to space limitations, we present such an integration in the technical report [41].

The idea of using information theory to model inference control was first used in our work for exact disclosure [40]. Significant differences between our work in this paper and [40] include:

- The difference in objective of privacy protection: We are dealing with both exact and partial privacy disclosure in this paper, instead of exact disclosure only in [40].
- The difference in supported OLAP queries: The inference control algorithm proposed in [40] only allows $(n-1)$- and n-dimensional queries for n-dimensional data cubes, while this paper supports queries with all dimensionalities.

The rest of the paper is organized as follows. We briefly review the background in Section II. In Section III, we present the system settings, performance measures, and problem statement. Also in this section, we define a continuous measure on the partial disclosure of privacy information. In Section IV, we introduce an information-theoretic formulation of OLAP queries, and identify what aggregate functions we support and do not support with our formulation. Then, we propose an inference control approach in Section V, followed by theoretical analysis in Section VI. We present simulation results on the performance of our approach in Section VII. In Section VIII, we discuss various extensions to our approach. We compare our results with related work in Section IX, and conclude the paper with final remarks in Section X.

2 Background

2.1 OLAP System Model

We consider an OLAP system where the data warehouse server stores data in an n-dimensional data cube, in order to support aggregate queries on an attribute-of-interest over selected data [20]. We refer to such attribute-of-interest as the measure attribute. Besides the measure attribute, there are n dimension attributes, each of which is represented by a dimension of the data cube. Each (base) cell of the data cube is the value of the measure attribute for the corresponding value combination of dimension attributes. For example, Table 1 shows a 2-dimensional data cube with a measure attribute of sales and two dimension attributes of product and time. Each cell of the data cube in Table 1 is the sales amount (i.e., measure attribute) of a product (e.g., Book) in a month (e.g., April). As in real cases, some cells in the data cube can be missing or not applicable (i.e., N/A in Table 1).

By changing the *group by* clause and the aggregation function, a user of the data warehouse can query various

TABLE 1
An Example of Inference Problem

	April	May	June	July	Sum
Book	10	12	15	7	$q_5 = 47$
CD	20	23	27	N/A	$q_6 = 70$
DVD	23	35	16	36	$q_7 = 110$
Game	N/A	25	30	14	$q_8 = 69$
Sum	$q_1 = 53$	$q_2 = 95$	$q_3 = 88$	$q_4 = 57$	

aggregates (e.g., COUNT, SUM, MIN, MAX, MEDIAN) of the measure attribute for different subcubes of the data cube. The answer to an OLAP query is the output of aggregation function on all cells in the subcube. We say that a subcube is h-dimensional if and only if it covers all values for (each of) h dimension attributes in the data cube. We refer to a query on an h-dimensional subcube of the data cube as an h-dimensional query. For example, query q_5 (i.e., SUM of row "Book") in Table 1 is a 1-dimensional SUM query that covers a 1-dimensional subcube of $\mathrm{Book} \times \{\mathrm{April}, \mathrm{May}, \mathrm{June}, \mathrm{July}\}$. Note that the answer to an h-dimensional query can also be considered as a cell in the corresponding $(n-h)$-dimensional cuboid [20] of the data cube. As is commonly assumed in the literature, we consider *skeleton queries*[2] [35], [38] to be the *granularity* of queries issued by the users [35], [37], [40]. However, note that this does *not* indicate that our algorithm only supports skeleton queries. Instead, we focus on the processing of skeleton queries in the upcoming discussions only for the ease of understanding. Our proposed algorithms readily apply to non-skeleton queries, as we shall discuss in Section 5.5.

2.2 Privacy Requirements

Due to privacy concerns, the owner of a data cube may not want a user to have access to all the information stored in it. Privacy requirements on a data cube may be defined over the cells in the data cube, and/or the data tuples in the relational back-end of it. In this paper, we choose a cell as the granularity for specification of privacy requirements due to our focus on the processing of OLAP queries. Nonetheless, as evidenced by our privacy measure in Section 3, privacy requirements on tuples can also be transformed to requirements on data cube cells. In our model, a user does not have the right to access all cells in the data cube. We refer to all cells that a user cannot access as *sensitive cells* for the user.

A user should not be capable of compromising the value of a sensitive cell from the query answers it receives. There are two measures for the compromise of a sensitive cell: exact and partial disclosure. While exact disclosure occurs iff the exact value of cell is known to a user, partial disclosure occurs whenever a user has a significant change between its prior and posterior belief on the value of a cell. We shall introduce the formal definitions for exact and partial disclosure in Section 3.3.

2. A query is a skeleton query if and only if the query covers either all values of a dimension attribute or only one value of it [35]. For example, a query on a row or a column of the 2-dimensional data cube is a skeleton query. In Table 1, each q_i is the answer to a 1-dimensional skeleton query with function of SUM.

2.3 External Knowledge

In practice, besides receiving aggregate query answers, a user may also learn certain information about a sensitive cell from sources outside the OLAP system. We refer to such information as the *external knowledge* of the user. There are various types of external knowledge which may lead to the disclosure of sensitive cells. For example, if a user knows that the maximum employee salary of a company belongs to Alice, its CEO, then the salary of Alice can be inferred from just one aggregate query on the maximum salary in the data cube. Even a MAX query on the salary of a subset of the employees can be privacy-divulging, as the answer can be used by an adversary as a lower-bound estimate on Alice's salary. It is difficult, if not possible, to have a universal model that captures all types of external knowledge. The proper modeling of external knowledge is still an open problem drawing considerable attention from data privacy studies [8], [27].

In this paper, we consider the type of external knowledge that is most commonly assumed and addressed in the literature of privacy-preserving OLAP [36], [37], [40] - a subset of cells in the data cube are known by the users as pre-knowledge. To ensure the security of sensitive cells, we adopt a conservative assumption that if a cell is not sensitive to (i.e., can be accessed by) a user, then the user knows the value of the cell as pre-knowledge. We refer to such non-sensitive cells as pre-known cells of the user. Table 2 shows the access privileges of a user on Table 1. For the ease of understanding, we count all pre-known cells as 0 in the computation of query answers (i.e., q_1 to q_8) in the table.

TABLE 2
An Example of Inference Problem

	April	May	June	July	Sum
Book	10	Known	15	Known	$q_5 = 25$
CD	20	Known	27	Known	$q_6 = 47$
DVD	Known	35	16	36	$q_7 = 87$
Game	Known	25	Known	14	$q_8 = 39$
Sum	$q_1 = 30$	$q_2 = 60$	$q_3 = 58$	$q_4 = 50$	

The formal definition of inference problem will be introduced in the next section. Intuitively, inference problem occurs if a user can infer certain information about a sensitive cell from the received query answers as well as the pre-known cells. For example, inference problem may occur in Table 2, in terms of the exact disclosure of sensitive cell $\langle$DVD, June$\rangle$, if the user learns query answers q_1, q_3, q_5, and q_6: The user can infer the sales amount of DVD in June by computing $\langle \text{DVD, June} \rangle = q_1 + q_3 - (q_5 + q_6) = 16$.

3 System Model

In this section, we introduce the system models for privacy protection in OLAP. In particular, we present the

system settings, models of privacy intrusion attacks and defensive countermeasures, performance measures for privacy disclosure and query availability, as well as the formal problem definition of inference control in OLAP.

3.1 System settings

Let there be one data warehouse server and a number of users $C_1, \ldots, C_U$ in the system. For each user C_k ($k \in [1, U]$), let G_k be the set of cells in the data cube that C_k has the right to access. We refer to G_k as the set of *pre-known cells* of C_k. Since the data warehouse server needs to properly enforce access control on the sensitive cells, we assume that it knows G_k for $k \in [1, U]$ as pre-knowledge.

To model the processing of OLAP queries in many real systems, we follow an online setting of the inference control problem [23], [29] which assumes that each user issues multiple OLAP queries sequentially, and the data warehouse server has no knowledge of the upcoming queries. In Section 8.2, we shall discuss an extension of our results to cases where a user can submit multiple queries in a batch. We shall show that while all results in the paper still apply to these cases, the simultaneous submission of multiple queries may allow us to further optimize the performance of inference control.

We denote a query q by a 2-tuple $\langle \mathtt{F}, \{x_1, \ldots, x_{|\mathrm{q}|}\}\rangle$, where $\mathtt{F}$ is the aggregate function of q, $|\mathrm{q}|$ is the number of cells (including both pre-known and sensitive cells) covered by q, and $x_1, \ldots, x_{|\mathrm{q}|}$ are the covered cells. If a cell x is covered by q, we say that $x \in \mathrm{q}$. For each query q, let q be the correct answer to q. For a given time, let Q_k be the set of queries q for which user C_k has received answers.[3] We refer to Q_k as the *query history* of C_k at the given time. Both the data warehouse server and C_k knows Q_k.

3.2 Models of Privacy Intrusion Attacks and Defensive Countermeasures

A user C_k can be an adversary intending to compromise the private information about sensitive cells x ($x \notin G_k$) that it has no right to access. In order to do so, C_k may launch an *inference attack* by inferring private information about x from the pre-known cells in G_k as well as the historic query answers in Q_k. Throughout this paper, we make a worst-case assumption that each (malicious) user is computationally unbounded in that its inference attack may compromise the maximum possible private information about x ($x \notin G_k$) from Q_k and G_k irrespective of the computational cost. In the majority of the paper, we follow a common assumption in the literature [1] that the users do not share G_k and Q_k with each other. Nevertheless, we shall extend our approach to address the threats from colluding users in Section 8.1.

3. With a slight abuse of notation but without introducing ambiguity, we also use Q_k to denote the set of query answers received by C_k.

In order to defend against inference attacks, the data warehouse server may employ an *inference control component* to control the query answers issued to the users. When the data warehouse server receives a query q from a user C_k, the inference control component of the data warehouse server either rejects the query or provides the correct answer by transmitting q to C_k. The decision is made based on the received query q as well as the pre-known set G_k and the query history Q_k of the user.

3.3 Performance Measures

An inference control approach should be measured by its ability to protect privacy and answer OLAP queries. We define the measures for privacy disclosure and query availability respectively, as follows.

3.3.1 Privacy Disclosure Measure

We define a continuous measure for privacy disclosure in order to capture both exact and partial disclosure of private information. For each cell x, we assume that the prior distribution of x is public information to both the data warehouse server and the users. This is a reasonable assumption in practice because many attributes, such as age and salary, have underlying probability distributions that both the data warehouse server and the users can learn from external knowledge [23]. We restrict our discussion in this paper to discrete distributions of x - we assume a simple discretization of continuous data to resemble discrete values. For the sake of simplicity, we also assume that a user cannot directly compromise information about a sensitive cell x from the pre-known cells in G_k (i.e., $H(x) = H(x|G_k)$ where $H(\cdot)$ is information entropy [12]), because otherwise no inference control approach can prevent such disclosure from happening. Note that 1) this does not mean that we assume the cells to be independent (we shall propose in Section 5.2 an algorithm that supports interdependent cells), and 2) when information about a sensitive cell x can indeed be inferred from the pre-known set G_k, we can always adjust our definition of the privacy disclosure measure accordingly by changing the belief of the data warehouse server and the users on the prior distribution of x.

Definition 1. *For a given user C_k and its query history Q_k, the level of privacy disclosure on a sensitive cell $x \notin G_k$ is defined as:*

$$l_{\mathrm{p}}(x; Q_k) = \frac{I(x; Q_k|G_k)}{H(x)}. \quad (1)$$

where $H(x)$ is the information entropy of x, and $I(x; Q_k|G_k)$ is the mutual information between x and Q_k given G_k.

Note that since G_k does not change during the query answering process, we do not include it as a variable in $l_{\mathrm{p}}(x; Q_k)$. Please refer to information theory textbook [12] for the formal definitions of information entropy and mutual information. Intuitively, $H(x)$ measures the amount of information in (i.e., the degree of uncertainty

of) x, while $I(x;Q_k|G_k)$ measures the amount of additional information about x that one can infer from Q_k given G_k as pre-knowledge. Due to our assumption of $H(x) = H(x|G_k)$, the privacy disclosure level $l_{\mathrm{p}}(x;Q_k) \in [0,1]$ measures the percentage of information about x that C_k can infer from Q_k and G_k. For example, when C_k is able to infer the precise value of x, there is $I(x;Q_k|G_k) = H(x)$ and $l_{\mathrm{p}}(x;Q_k) = 1$. When C_k can infer no information about x, there is $I(x;Q_k|G_k) = l_{\mathrm{p}}(x;Q_k) = 0$. The greater $l_{\mathrm{p}}(x;Q_k)$ is, the more information about x can be inferred by C_k from Q_k and G_k.

In order to provide an intuitive explanation of our privacy disclosure measure, we show that a data owner can specify its privacy requirements on the posterior probability $P(x|Q_k,G_k)$ in terms of an upper bound on $l_{\mathrm{p}}(x;Q_k)$. Suppose that x is a discrete variable with domain set of $V(x)$. We have the following theorem:

Theorem 3.1. *For all users C_k, cells x, and values $x_0 \in V(x)$, no user can have expected probability of $\Pr\{x = x_0|Q_k,G_k\} > p_1$ or $\Pr\{x = x_0|Q_k,G_k\} < p_2$, where $p_1 > 1/|V(x)|$ and $p_2 < 1/|V(x)|$, if*

$$l_{\mathrm{p}}(x;Q_k) \leq 1 + \min_{i\in\{1,2\}} \frac{(1-p_i)\cdot \log \frac{1-p_i}{|V(x)|-1} + p_i \log p_i}{H(x)}. \quad (2)$$

Due to space limitations, please refer to [41] for the proof. As we can see from the theorem, the more difference there is between $\Pr\{x = x_0|Q_k,G_k\}$ and $1/|V(x)|$, the greater $l_{\mathrm{p}}(x;Q_k)$ will be. For example, when the prior distribution of x is uniform on $\{0,1\}$, no user can have expected probability of $\Pr\{x = 1|Q_k,G_k\} > 3/4$ or $< 1/4$ if $l_{\mathrm{p}}(x;Q_k) \leq 0.188$. Similar to the transformation from $P(x|Q_k,G_k)$ to $l_{\mathrm{p}}(x;Q_k)$ demonstrated in the theorem, a data owner can also transform many other probabilistic notions of privacy requirements, such as the probabilistic compromise measure [23] and the ρ_1-to-ρ_2 privacy measure [17], to upper bounds on $l_{\mathrm{p}}(x;Q_k)$.

Note that when we adopt the information-theoretic measure of $l_{\mathrm{p}}(x;Q_k)$ to capture the partial disclosure of private information, we also introduce an inherent problem of information-theoretic measures found in existing work [17]: Since an information-theoretic measure quantifies the *average* amount of disclosed information, there may exist extreme-case privacy disclosure with a small probability of occurrence that cannot be captured by information-theoretic measures. We shall further elaborate on such extreme-case privacy disclosure by examples shown in Section 4.2. In addition, we provide a solution in the technical report [41]. Nonetheless, we argue that the information-theoretic measure $l_{\mathrm{p}}(x;Q_k)$ is still suitable for measuring privacy disclosure in OLAP because 1) as we shall show in Section 4.2, the extreme-case privacy disclosure only occurs with a rather small probability, and 2) as we shall show in [41], when such extreme-case privacy disclosure occurs, it can be effectively eliminated by integrating inference control with input/output perturbation.

3.3.2 Query Availability Measure

Another important performance measure for privacy protection in OLAP is the system's ability to provide accurate answer queries to the users (i.e., the utility of inference control [29]). Since an inference control approach either rejects a query or answers it correctly, for a given user C_k, we define the query availability level $l_{\mathrm{a}}(C_k) \in [0,1]$ as the percentage of queries issued by C_k that are (correctly) answered by the data warehouse server. Note that the query availability level depends not only on the set of pre-known cells G_k and the queries issued by C_k, but also on the order in which the queries are issued. Formally, we have the following definition:

Definition 2. *Let $\vec{Q} = \langle \mathrm{q}_1, \ldots, \mathrm{q}_m \rangle$ be a sequence (i.e., ordered list) of queries issued by C_k. The query availability level for C_k is defined as:*

$$l_{\mathrm{a}}(C_k) = \frac{|\{\mathrm{q}_i | \mathrm{q}_i \textit{ is answered when } C_k \textit{ issues } \vec{Q}\}|}{m}. \quad (3)$$

3.4 Problem Statement

Based on the performance measures defined in the above subsection, we can now formally state the problem of inference control. As we can see, the objective of inference control in OLAP is to answer as many queries as possible without violating the privacy requirements imposed by the data owners. It is impractical to guarantee that absolutely no private information will be disclosed because almost every query answer that covers a sensitive cell x may disclose a certain (albeit small) percentage of private information about x (i.e., $I(q;x|G_k) > 0$ for $x \in \mathrm{q}$ and $x \notin G_k$). For example, a MIN query answer $q = v$ on 100 cells always reveals that every cell included in the query has value of at least v, and thus may disclose certain information about each cell. As such, we allow the owners of private cells to specify an *upper bound* on the level of privacy disclosure as follows.

Definition 3. *For a given cell x, the maximum acceptable level of privacy disclosure on x is an upper bound $l(x) \in (0,1]$ specified by the owner of x such that for each user C_k which does not know x as pre-knowledge (i.e., $x \notin G_k$), the data warehouse server must guarantee that at any time, the query history Q_k of the user satisfies*

$$l_{\mathrm{p}}(x;Q_k) < l(x). \quad (4)$$

Due to the definition, the greater $l(x)$ is, the more private information about x is allowed to be disclosed by Q_k. An intuitive reference point for $l(x)$ is $l(x) = 1$, in which case a privacy breach occurs if and only if the exact value of x can be learned by a user C_k with $x \notin G_k$. This essentially reduces the problem to the exact disclosure investigated in previous work [35], [37], [40]. When $l(x) < 1$, recall that Theorem 3.1 provides guidelines for a data owner to properly transform its privacy requirements on the posterior probability $P(x|Q_k,G_k)$ to an upper bound on $l_{\mathrm{p}}(x;Q_k)$.

Note that although we allow data owners the freedom to assign arbitrary $l(x)$ to their cells, some of the bounds, if not carefully chosen, may not be meaningful in real systems due to the interdependency between values of different cells. For example, consider the case where two cells x_1 and x_2 always have the same value (i.e., $x_1 \equiv x_2$). If the owner of x_1 specifies $l(x_1) = l_1$, any bound on x_2 with $l(x_2) > l_1$ is meaningless because by the time a user C_k reaches $l_{\mathrm{p}}(x_2; Q_k) \geq l(x_2)$, the user must have already violated the privacy of x_1 (i.e., $l_{\mathrm{p}}(x_1; Q_k) = l_{\mathrm{p}}(x_2; Q_k) \geq l(x_2) > l_1$). In this paper, we address such "inconsistent" owner-specified thresholds by designing inference control approaches that satisfy the maximum acceptable disclosure levels imposed on *all* cells. This way, our design always follows the most restrictive bound such as l_1 in the above example.

Based on the definition of maximum acceptable privacy disclosure level, we can now define the *safety* of a query:

Definition 4. *For a given user C_k and its query history Q_k, a query q issued by C_k is safe if and only if $\forall x \notin G_k$,*

$$l_{\mathrm{p}}(x; Q_k \cup \{\mathrm{q}\}) < l(x). \tag{5}$$

According to the definition, a query is safe if and only if by answering the query, the data warehouse server will not violate the maximum acceptable level of privacy disclosure imposed by the data owners. Recall that the data warehouse server has no knowledge about future queries. Hence, an ideal inference control approach should answer a query *if and only if* the query is safe. Nonetheless, it is difficult, if not impossible, to achieve this ideal objective efficiently. Indeed, a special case of the problem, where $l(x) = 1$ for all cells x, has been proven to be NP-hard in statistical databases with the presence of mixed SUM and MAX queries [9], [14]. In order to design efficient inference control approaches, a common compromise is to reject all unsafe queries, and to answer as many safe queries as possible. Formally, the problem of inference control is stated as follows.

Definition 5. (Problem Statement). *For a given user C_k, the objectives of inference control on processing queries from C_k include:*

- *Objective O_1: to reject all unsafe queries submitted by C_k, and*
- *Objective O_2: to maximize the query availability level $l_{\mathrm{a}}(C_k)$ when O_1 is achieved.*

In the following sections, we shall propose inference control algorithms based on the problem statement.

4 Our New Approach

In this section, we present the basic idea of our information-theoretic approach for inference control. We first introduce the basic workflow of our approach. Then, we define two categories of aggregate functions, namely MIN-like and SUM-like functions, based on their information-theoretic properties. We use examples to illustrate our basic ideas of dealing with these two types of functions respectively. The detailed algorithms of our approach will be presented in Section 5.

4.1 Basic Workflow

We first briefly introduce the basic workflow of our inference control approach. Recall that Objective O_1 of inference control is to guarantee $l_{\mathrm{p}}(x; Q_k) < l(x)$ for all sensitive cells $x \notin G_k$. Also note that we cannot directly compute $l_{\mathrm{p}}(x; Q_k)$ for each and every cell x because of the large number of possible cells (which grows exponentially with the dimensionality n). Thus, in order to achieve Objective O_1, we propose to maintain an *upper-bound estimate* of $\max_x l_{\mathrm{p}}(x; Q_k)$ as $l_{\max}(Q_k)$ for each user C_k, such that $\forall x \notin G_k$, $l_{\max}(Q_k) \geq l_{\mathrm{p}}(x; Q_k)$. As such, when a new query q is received from C_k, the data warehouse server only needs to compute $l_{\max}(Q_k \cup \{\mathrm{q}\})$ based on $l_{\max}(Q_k)$, Q_k, G_k, and q, and answers q if and only if $l_{\max}(Q_k \cup \{\mathrm{q}\}) < l(x)$. If q is answered, the server needs to update $l_{\max}(Q_k)$ to $l_{\max}(Q_k \cup \{\mathrm{q}\})$.

Since a loose upper bound on $l_{\max}(Q_k)$ may prevent future queries from being answered, in order to achieve Objective O_2 of inference control, the estimated upper bound on $l_{\max}(Q_k)$ 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 on $l_{\max}(Q_k)$. We focus on our ideas for tackling this challenge in the rest of this subsection.

Note that $l_{\mathrm{p}}(x; Q_k) = 0$ when Q_k is empty. Thus, in order to generate an upper-bound estimate on $l_{\max}(Q_k \cup \{\mathrm{q}\})$, we only need to compute

$$l_{\max}(\mathrm{q}|Q_k) = \max_x (l_{\mathrm{p}}(x; Q_k \cup \{\mathrm{q}\}) - l_{\mathrm{p}}(x; Q_k)). \tag{6}$$

which is (intuitively) the additional amount of information about x that one can derive from the query answer q given Q_k and G_k are pre-knowledge. A critical step for efficiently computing a (fairly) tight upper bound on $l_{\max}(\mathrm{q}|Q_k)$ is to change the view of computing $l_{\max}(\mathrm{q}|Q_k)$ as follows.

$$l_{\max}(\mathrm{q}|Q_k) = \max_x \frac{I(x; \{Q_k, q\}|G_k) - I(x; Q_k|G_k)}{H(x)} \tag{7}$$

$$= \max_x \frac{I(x; q|Q_k, G_k)}{H(x)} \tag{8}$$

$$= \max_x \frac{I(q; x|Q_k, G_k)}{H(x)} \tag{9}$$

$$= \max_x \frac{H(q|Q_k, G_k) - H(q|Q_k, G_k, x)}{H(x)}. \tag{10}$$

Intuitively, $H(q|Q_k, G_k) - H(q|Q_k, G_k, x)$ in the equation is the additional amount of information about q that can be derived from x while given Q_k and G_k as pre-knowledge. This transformation is critical in that it

enables us to change our view from considering how much information about x is disclosed by q to how much information about q can be derived from x. After the transformation, the computation of $l_{\max}(\mathrm{q}|Q_k)$ can be restated as follows:

Given G_k and the current value of Q_k, how much additional information about q can C_k learn if it knows one more cell as pre-knowledge?

As we shall show in the next subsection, the restated problem allows us to efficiently derive an upper bound on $l_{\max}(\mathrm{q}|Q_k)$ (for all x) without computing $l_{\mathrm{p}}(x; Q_k \cup \{\mathrm{q}\})$ for each and every $x \notin G_k$. In particular, we shall use some examples to illustrate our basic idea of computing $l_{\max}(\mathrm{q}|Q_k)$ for various types of aggregate functions. The detailed computation of $l_{\max}(\mathrm{q}|Q_k)$ will be illustrated in the inference control algorithms presented in Section 5.

We would like to note that the basic workflow introduced above is not limited to inference control in OLAP. More generally, if we consider x as a sensitive data point and q as an aggregate query over such data points, then the above workflow readily applies to the traditional problem of inference control for statistical databases [1]. Nonetheless, the following discussions on the actual processing of various aggregate functions, especially SUM-like ones, are specific to an OLAP system. Due to space limitations, we discuss in the technical report [41] the extension of our results to the generic inference control problem.

4.2 Basic Ideas for Various Aggregate Functions

The computation of $l_{\max}(q|Q_k)$ critically depends on the aggregate function of q (e.g., MIN or SUM). In this subsection, we shall introduce two categories of common aggregate functions, namely MIN-like and SUM-like functions, each of which is composed of functions that share common characteristics in the computation of $l_{\max}(q|Q_k)$. For each category, we shall present our basic ideas for computing $l_{\max}(q|Q_k)$ of queries with aggregate functions in that category.

4.2.1 MIN-like Functions

MIN-like functions include MIN and MAX. A common property of MIN-like functions is, for query size $|\mathrm{q}| \gg 1$, there is $H(q) \ll H(x)$. We demonstrate this property in the following example:

Suppose that each cell x is chosen uniformly at random from 0 and 1. Consider query q which is the minimum of 100 such cells (i.e., $|\mathrm{q}| = 100$). Suppose that all $\log(\cdot)$ in the paper is logarithm with base 2. We have $H(x) = \log(2) = 1$, and $H(q) \approx 100/2^{100} \ll 1$.

With this property of $H(q)$ being extremely small with large $|q|$, we can compute an upper bound on $l_{\max}(\mathrm{q}|Q_k)$ as follows: $\forall Q_k$ and G_k, we have

$$l_{\max}(\mathrm{q}|Q_k) \leq \max_x \frac{H(q)}{H(x)} \approx \frac{1}{2^{93}}. \quad (11)$$

That is, due to the upper bound derived in (11), we can safely answer about $2^{93} \cdot l(x)$ of such MIN queries without violating the privacy requirements of data owners. As we can see, this lower bound is tight enough for many practical OLAP applications. Thus, the property of MIN-like functions can enable us to derive effective upper bounds of $l_{\max}(\mathrm{q}|Q_k)$ as $\max_x(H(q)/H(x))$. As we shall show in Section 5, this is our basic idea of dealing with MIN-like functions in the proposed inference control algorithms. Note that such computation is extremely efficient (with complexity of $O(1)$) and does not require the storage of any historical queries (other than the original value of $l_{\max}(Q_k)$ in order to properly update it to $l_{\max}(Q_k \cup \{q\})$).

There may be a question, as in the above example, of what will happen if the answer to the MIN query is $q = 1$ when $|q| = 100$? 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 mentioned in Section 3.3. Note that extreme-case privacy disclosure has a small probability of happening, but (once it happens) can result in serious privacy breaches. Since the probability of $q = 1$ is extremely small, our information-theoretic privacy measure (for the average case) cannot capture such disclosure. Our approach may be extended to prevent such extreme-case disclosure from happening. Due to space limitations, we present such an extension in the technical report [41].

4.2.2 SUM-like Functions

SUM-like functions include COUNT, SUM, AVG, MEDIAN, MODE, and STANDARD DEVIATION.

Our basic idea of processing a SUM-like query is to count the number of cells covered by the query that can independently change the query answer without influencing (or violating) the historic queries answers and the prior knowledge of the adversaries. Therefore, we define SUM-like functions as non-MIN-like aggregate functions that share the following property: For a given query q, when a sensitive cell $x \in \mathrm{q}$ changes its value from x_0 to x_1, for any other cell $y \in \mathrm{q}$, there always exists a pair of values y_0 and y_1, such that when y also changes from y_0 to y_1 and no other cell in q changes, the query answer q remains the same. As we can see, the definition of SUM-like queries mandates that a cell can change without alternating a SUM-like query answer that covers it. For example, when the function is SUM, there exists $y_1 = x_0 + y_0 - x_1$. Note that MIN and MAX do not satisfy this condition. For example, when the minimum cell x covered by a MIN query changes from x_0 to $x_0 - 1$, the query answer q can never remain the same no matter how the other cells change their values.

We now introduce our basic idea to compute $l_{\max}(\mathrm{q}|Q_k)$ for SUM-like functions. In order to do so, we first define the *maximum self-supportive subset* of a SUM-like query q. Based on the definition of a SUM-like query, a self-supportive subset further isolates a set of cells that can independently change a SUM-like query answer.

Definition 6. *Given Q_k and q, a subset Ω of cells covered by q (i.e., $\Omega \subseteq \mathrm{q}$) is self-supportive iff Ω consists of sensitive cells only (i.e., $\Omega \cap G_k = \varnothing$), and for all $x \in \Omega$, there exists a corresponding set of sensitive cells $S(x)$ such that*

1) $x \in S(x)$,
2) $\forall x_1 \in \Omega$ *with* $x_1 \neq x$, $S(x) \cap S(x_1) = \varnothing$, *and*
3) *When x changes its value and all cells in the data cube except those in $S(x)$ remain unchanged, all answers to SUM-like queries in Q_k can still remain the same.*

A self-supportive subset Ω of q is maximum self-supportive iff there is no self-supportive subset Ω' of q such that $|\Omega| < |\Omega'|$.

To illustrate the definition of self-supportive subset, consider the example shown in Table 2 of Section 2. When $Q_k = \{q_1, q_3, q_5\}$ and $\mathrm{q} = \mathrm{q}_6$, {CD/June} is self-supportive with S(CD/June) = {CD/June, DVD/June} because S(CD/June) $\cap\, \mathrm{q}$ = CD/June, and when CD/June changes its value, say from 27 to 29, no query answer in Q_k will change if DVD/June changes accordingly from 16 to 14. {CD/April} is also self-supportive with S(CD/April) = {CD/April, Book/April, Book/June, DVD/June}. Nevertheless, q_6 itself ({CD/April, CD/June}) is not self-supportive because S(CD/April) and S(CD/June) have either DVD/June or CD/April in common, which violates Condition 2 of Definition 6. Thus, both {CD/June} and {CD/April} are maximum self-supportive subsets of q.

Let n_{q} be the size of a maximum self-supportive subset of q. We propose to process SUM-like queries by computing an upper-bound estimate of $l_{\max}(\mathrm{q}|Q_k)$ based on n_{q}. The basic idea of such computation is due to the following two properties of SUM-like functions which we shall demonstrate in Section 6: 1) the value of $H(q|Q_k, G_k) - H(q|Q_k, x, G_k)$ decreases when n_{q} increases, and 2) a lower bound on n_{q} can be efficiently derived from the number of sensitive cells covered by q and by the query history Q_k. As such, our methodology for computing $l_{\max}(\mathrm{q}|Q_k)$ is to first derive a lower bound on n_{q}, and then compute $l_{\max}(\mathrm{q}|Q_k)$ based on the derived lower bound.

For example, consider a 2-dimensional data cube with each cell chosen uniformly at random from 0 and 1. Suppose that $Q_k \cup \{\mathrm{q}\}$ is composed of skeleton queries on the SUM of a row or a column of the data cube. As we shall prove in Section 6, there is

$$\frac{l_{\max}(\mathrm{q}|Q_k)}{1 - l_{\max}(Q_k)} \leq \frac{1}{2} \log \left(\frac{n_{\mathrm{q}}}{n_{\mathrm{q}} - 1} \right). \tag{12}$$

Consider the case where $Q_k = \varnothing$. When $n_{\mathrm{q}} = 2$, we have $l_{\max}(\mathrm{q}|Q_k) \leq 1/2$. Nonetheless, when n_{q} increases to 100, the value of $l_{\max}(\mathrm{q}|Q_k)$ can be reduced to less than $1/100$. As we can see, when n_{q} is large, the value of $l_{\max}(\mathrm{q}|Q_k)$ derived from n_{q} can serve as an effective upper-bound estimate for processing queries with SUM-like functions.

4.2.3 Discussions of MIN-like and SUM-like functions

MIN-like and SUM-like functions cover most of the common aggregate functions: In this paper, we consider the following lists of MIN-like and SUM-like functions:

- MIN-like functions: MIN and MAX
- SUM-like functions: SUM, AVG, COUNT, MEDIAN, and STANDARD DEVIATION.

We would like to note that our processing of MIN-like functions can also be extended to other similar functions, such as RANGE (i.e., MAX-MIN), as long as the function has entropy much smaller than that of the covered data points. Nonetheless, MIN-like and SUM-like functions do not constitute a comprehensive classification of all aggregate functions. For example, consider an aggregate function that returns, as a result, a 2-tuple $q = \langle q_1, q_2 \rangle$ where q_1 and q_2 are the MIN and SUM of all covered cells, respectively. This function does not satisfy the property of MIN-like functions because $H(q) \geq H(q_2) \geq H(x)$. It is not SUM-like either because as we have shown above, the q_1 component with MIN function does not satisfy the property of SUM-like functions. Thus, this aggregate function is neither MIN-like nor SUM-like[4].

Note that our definition of MIN-like and SUM-like functions is independent of the classification of aggregation functions into distributive, algebraic, and holistic ones [20]. For example, a SUM-like function can be distributive (e.g., SUM), algebraic (e.g., AVG), or holistic (e.g., MEDIAN).

5 Inference Control Algorithm

In this section, we present the detailed algorithm of our information-theoretic inference control approach. In order to help readers better understand the methodology of our proposed approach, we first show the algorithm on a simple case of 2-dimensional data cube in which each cell follows a uniform distribution on $\{0, 1\}$ and has a constant $l(x)$. After that, we present a generic algorithm for n-dimensional data cubes with cells of arbitrary distribution and heterogeneous $l(x)$. At the end of this section, we shall show that both algorithms satisfy the simulatable auditing model [23] and disclose no private information through query rejections.

5.1 Algorithm A: A Simple 2-Dimensional Case

We first consider a simple case with a 2-dimensional $d_1 \times d_2$ data cube, in which each sensitive cell x is uniformly distributed on $\{0, 1\}$, and has a constant owner-specified threshold $l(x) = l \in [0, 1]$. Note that the assumptions of uniform distribution and constant $l(x)$ (for all cells) are certainly not realistic models of real-world data. Nonetheless, we would like to remark that the introduction of Algorithm A is not intended to promote the practical usage of it. Instead, we use it as a simple demonstration of the power of our information-theoretic approach, and a foundation for the introduction of Algorithm B, which is a generic and practical algorithm for n-dimensional data cubes with arbitrary distribution.

4. Nevertheless, this function can be supported by our approach if we consider it as two separate queries on MIN and SUM.

Before presenting the inference control algorithm, we first introduce some basic notions: Given user C_k and received query q, let $r_k(\mathrm{q})$ be the number of cells in q that belong to G_k (i.e., known by C_k as pre-knowledge). Recall that $|\mathrm{q}|$ is the number of all cells in q. Let Q_k^{S} be the set of SUM-like queries in the query history Q_k, and $|Q_k^{\mathrm{S}}|$ be the number of queries in Q_k^{S}. Let σ_k be the number of pre-known cells covered by at least one query in Q_k^{S}. That is,

$$\sigma_k = |\{x | x \in G_k, \exists q \in Q_k^{\mathrm{S}} \text{ such that } x \in q\}|. \quad (13)$$

Let μ_k be the minimum number of sensitive cells covered by a SUM-like query in the query history:

$$\mu_k = \min_{q: q \in Q_k^{\mathrm{S}}} (|\mathrm{q}| - r_k(\mathrm{q})). \quad (14)$$

Algorithm A shows our inference control algorithm for the simple case. In the algorithm, we use l_{p} to denote the current upper-bound estimate on $l_{\max}(Q_k)$. When $Q_k = \varnothing$, the initial values of the parameters are $\mu_k = \infty$, $\sigma_k = 0$, and $l_{\mathrm{p}} = 0$. With the algorithm, when a new query q is received, the data warehouse server computes an upper-bound estimate on $l_{\max}(\mathrm{q}|Q_k)$ as l_0, and answers the query if and only if $l_{\mathrm{p}} + l_0$ is less than the owner-specified threshold l.

Algorithm A Inference Control for a $d_1 \times d_2$ data cube with $\{0, 1\}$-Uniform Distribution

1: {When a query q is received.}
2: **if** function of q is MIN-like **then**
3: $l_0 \leftarrow |\mathrm{q}| - \log(2^{|\mathrm{q}|} - 1) + \frac{\log(2^{|\mathrm{q}|}-1)}{2^{|\mathrm{q}|}}$.
4: **else if** function of q is SUM-like **and** $\mu_k > 1$ **then**
5: $\mu \leftarrow \min(\mu_k, d_1 + d_2 - |Q_k^{\mathrm{S}}|, d_1/2, d_2/2)$.
6: $n_0 \leftarrow |\mathrm{q}| - r_k(\mathrm{q}) - \max(0, \lfloor \frac{\sigma_k - (\mu-1)|\mathrm{q}|}{d_1+d_2-|\mathrm{q}|+1-2\mu} \rfloor)$.
7: $l_0 \leftarrow \frac{1-l_{\mathrm{p}}}{2} \log \frac{n_0}{n_0-1}$.
8: **end if**
9: **if** $l_{\mathrm{p}} + l_0 \geq l$ **then**
10: **return** $\varnothing$. {Reject query q}
11: **else**
12: **if** function of q is SUM-like **then**
13: $\mu_k \leftarrow \min(\mu_k, (|\mathrm{q}| - r_k(\mathrm{q}))$.
14: $\sigma_k \leftarrow \sigma_k + r_k(\mathrm{q})$.
15: $|Q_k^{\mathrm{S}}| \leftarrow |Q_k^{\mathrm{S}}| + 1$.
16: **end if**
17: $l_{\mathrm{p}} \leftarrow l_{\mathrm{p}} + l_0$.
18: **return** q. {Answer query q correctly}
19: **end if**

In particular, if the received query q has MIN-like aggregate function, l_0 is computed in Step 3 as

$$l_0 = \frac{H(q)}{H(x)} = |\mathrm{q}| - \log(2^{|\mathrm{q}|} - 1) + \frac{\log(2^{|\mathrm{q}|} - 1)}{2^{|\mathrm{q}|}}. \quad (15)$$

Since $l_{\max}(\mathrm{q}|Q_k) \leq H(q)/H(x)$ for MIN-like functions (due to (11)), MIN-like query answers issued by Algorithm A will not violate the privacy requirements specified by the data owners.

If the received query q has SUM-like aggregate function, the data warehouse server first computes in Steps 5 and 6 a lower-bound estimate of n_{q} as n_0 based on μ_k, σ_k, and $|Q_k^{\mathrm{S}}|$. Recall that n_{q} is the number of cells in the maximum self-supportive subset of q. Then, the data warehouse server computes l_0 in Step 7 as

$$l_0 = \frac{1 - l_{\mathrm{p}}}{2} \log \frac{n_0}{n_0 - 1}, \quad (16)$$

we shall justify the computation of l_0 in Section 6 by proving that $l_{\max}(\mathrm{q}|Q_k) \leq l_0$ for SUM-like functions.

If $l_p + l_0 < l$ (i.e., query answer q can be issued to user C_k), then the data warehouse server updates the values of μ_k, σ_k, $|Q_k^{\mathrm{S}}|$, and l_{p} in Steps 12 to 17. As we can see, only l_{p} needs to be updated for MIN-like queries while all four parameters are updated for SUM-like ones.

TABLE 3
Execution of Algorithm A on Table 2

Case 1:

After	Value of n_0 and l_0	Response	Value of μ_k and σ_k
q_1	$n_0 = 2, l_0 = 1/2$	$q_1 = 30$	$\mu_k = 2, \sigma_k = 2$
q_5	$n_0 = 2, l_0 = 1/4$	$q_5 = 25$	$\mu_k = 2, \sigma_k = 4$
q_6	$n_0 = 2, l_0 = 1/8$	$q_6 = 47$	$\mu_k = 2, \sigma_k = 6$
q_3	$n_0 = 1, l_0 = \infty$	Reject	$\mu_k = 2, \sigma_k = 6$

Case 2:

After	Value of n_0 and l_0	Response	Value of μ_k and σ_k
q_1	$n_0 = 2, l_0 = 1/2$	$q_1 = 30$	$\mu_k = 2, \sigma_k = 2$
q_3	$n_0 = 3, l_0 \approx 0.15$	$q_3 = 58$	$\mu_k = 2, \sigma_k = 3$
q_5	$n_0 = 2, l_0 \approx 0.08$	$q_5 = 25$	$\mu_k = 2, \sigma_k = 5$
q_6	$n_0 = 1, l_0 = \infty$	Reject	$\mu_k = 2, \sigma_k = 5$

Table 3 demonstrates the execution of Algorithm A for the two-dimensional data cube defined in Table 2. Since the processing of MIN-like queries is relatively straightforward, we study two sequences of SUM-like queries, Case 1: $\langle \mathrm{q}_1, \mathrm{q}_5, \mathrm{q}_6, \mathrm{q}_3 \rangle$ and Case 2: $\langle \mathrm{q}_1, \mathrm{q}_3, \mathrm{q}_5, \mathrm{q}_6 \rangle$, respectively, as examples. For each sequence, Table 3 shows the values of n_0 and l_0 after each query is received and the answer/reject decision made when the owner-specified threshold is $l = 1$ (i.e., only exact disclosure is of concern), as well as the values of μ_k and σ_k after the decision. As we can see from Table 3, the query answer/reject decisions made by Algorithm A conforms to our analysis in Section 2 which concludes that both sequences will result in the exact disclosure of DVD/June, and thus should be rejected at the last query when only exact disclosure is of concern.

5.2 Algorithm B: A Generic n-d Algorithm

We now present Algorithm B as a generalization of Algorithm A to n-dimensional data cubes. Compared with Algorithm A, Algorithm B removes the assumption of uniform distribution and constant $l(x)$ on all cells. Instead, Algorithm B supports cells of arbitrary prior distribution and heterogeneous owner-specified thresholds $l(x)$. Most of the variables used in Algorithm B are the same as those in Algorithm A. To accommodate the

arbitrary thresholds $l(x)$, we introduce two additional parameters $l(\mathrm{q})$ and l_k. $l(\mathrm{q})$ is the minimum value of $l(x)$ for all cells x included in a query q,

$$l(\mathrm{q}) = \min_{x:x\in\mathrm{q}} l(x), \tag{17}$$

while l_k is the minimum value of $l(\mathrm{q})$ for all $\mathrm{q} \in Q_k$.

$$l_k = \min_{q:q\in Q_k} l(\mathrm{q}). \tag{18}$$

To accommodate the arbitrary distribution of x, we introduce three auxiliary inputs: $H(x)$, $f_{\max}(\mathrm{q}, c)$, and $f_{\min}(\mathrm{q}, c)$, all of which can be derived from the prior distribution of cells, which is public information available to both the data warehouse server and the users due to our system settings presented in Section 3. In particular, $H(x)$ is the minimum entropy of a sensitive cell x while $f_{\max}(\mathrm{q}, c)$ and $f_{\min}(\mathrm{q}, c)$ are, respectively, the maximum and minimum entropy of a query answer which has the same aggregate function as q and is composed of c cells in q. For example, in the simple case addressed by Algorithm A, we have

$$f_{\max}(\mathrm{q}, c) = f_{\min}(\mathrm{q}, c) = c - \log(2^c - 1) + \frac{\log(2^c - 1)}{2^c} \tag{19}$$

for queries q with MIN-like aggregation functions, and

$$f_{\max}(\mathrm{q}, c) = f_{\min}(\mathrm{q}, c) \approx \frac{1}{2}\log\left(\frac{\pi e c}{2}\right) \tag{20}$$

for queries q with SUM function. It is easy to verify that for the simple case, Algorithm B can be reduced to Algorithm A if we substitute $l(\mathrm{q})$ and l_k with l, and substitute $f_{\max}(\mathrm{q}, c)$ and $f_{\min}(\mathrm{q}, c)$ with the corresponding values in (19) and (20). Note that in the general case addressed by Algorithm B, the values of $f_{\max}$ and $f_{\min}$ might be different (i.e., $f_{\max}(\mathrm{q}, c) > f_{\min}(\mathrm{q}, c)$) when cells covered by q follow different distributions or are correlated with each other (i.e., $I(x_1, x_2) > 0$ for $x_1, x_2 \in \mathrm{q}$).

The execution process of Algorithm B is similar to that of Algorithm A, with an exception that instead of maintaining one copy of parameters μ_k, σ_k, and $|Q_k^{\mathrm{S}}|$ for each user, Algorithm B maintains a different copy of μ_k and σ_k for each of the previously answered SUM-like queries. In particular, for each subcube Θ corresponding to a SUM-like query answer $q_0 \in Q_k^{\mathrm{S}}$, Algorithm B maintains $\langle \Theta, \mu_k(\Theta), \sigma_k(\Theta)\rangle$ in the subcube-history set Θ_k^{S}, where $\mu_k(\Theta)$ is the number of sensitive cells in Θ and $\sigma_k(\Theta)$ is the number of pre-known cells in Θ that has not been covered by any query answer issued prior to q_0. With the subcube-history set Θ_k^{S}, while processing an h-dimensional SUM-like query q, the data warehouse server computes l_0 as the maximum value for all $(h+1)$-dimensional (skeleton) subcubes Θ_i (defined in Step 6) that contain q as a subset. In the computation for each Θ_i (Steps 7 to 11), we take into consideration the values of $\mu_k(\Theta')$ and $\sigma_k(\Theta')$ for all Θ' in the subcube-history Θ_k^{S} that satisfies $\Theta' \subseteq \Theta_i$. As we can see, when $n = 2$ and $h = 1$ (i.e., the two-dimensional case addressed by Algorithm A), there is only one $(h+1)$-dimensional subcube which is the data cube itself. Thus, $|\Gamma_i|$, $\min\{\mu_k(\Theta)|\Theta \in \Gamma_i\}$ and $\sum\{\sigma_k(\Theta)|\Theta \in \Gamma_i\}$ in Algorithm B will be reduced to Q_k^{S}, μ_k, and σ_k in Algorithm A, respectively.

Algorithm B for n-d Arbitrary Distribution

Require: h-dimensional query q on subcube $(a_1, \ldots, a_{n-h}, \mathrm{ALL}, \ldots, \mathrm{ALL})$, $l_0 = 0$.

1: {When a query q is received.}
2: **if** function of q is MIN-like **then**
3: $l_0 \leftarrow f_{\max}(\mathrm{q}, |\mathrm{q}|)/H(x)$.
4: **else if** function of q is SUM-like **then**
5: **for** $i \leftarrow 1$ to $n - h$ **do**
6: $\Theta_i \leftarrow (a_1, \ldots, a_{i-1}, \mathrm{ALL}, a_{i+1}, \ldots, a_{n-h}, \mathrm{ALL}, \ldots, \mathrm{ALL})$.
7: Find $\Gamma_i \leftarrow \{\Theta'|\Theta' \in \Theta_k^{\mathrm{S}}, \Theta' \subseteq \Theta_i\}$.
8: $\mu \leftarrow \min(\min\{\mu_k(\Theta')|\Theta' \in \Gamma_i\}, d_i + d_{n-h+1} + d_{n-h+2} + \cdots + d_n - |\Gamma_i|, d_i/2, d_{n-h+1}/2, \ldots, d_n/2)$.
9: $t \leftarrow$ the maximum integer that satisfies $\sum\{\sigma_k(\Theta')|\Theta' \in \Gamma_i\} \geq t(d_i - \mu) + t^{\frac{h-1}{h}}(\mu - 1)(d_1 + d_2 + \cdots + d_h - ht^{\frac{1}{h}})$, assuming $0^0 = 1$.
10: $n_0 \leftarrow |\mathrm{q}| - r_k(\mathrm{q}) - t$.
11: $l_0 \leftarrow \max(l_0, (1 - l_{\mathrm{p}}) \cdot (f_{\max}(\mathrm{q}, n_0) - f_{\min}(\mathrm{q}, n_0 - 1))/H(x))$.
12: **end for**
13: **end if**
14: **if** $l_{\mathrm{p}} + l_0 \geq \min(l_k, l(\mathrm{q}))$ **then**
15: **return** $\varnothing$. {Reject query q}
16: **else**
17: **if** function of q is SUM-like **then**
18: $\mu_k(\Theta) \leftarrow \mu(\Theta)$.
19: $\Gamma \leftarrow \{\Theta'|\Theta' \in \Theta_k^{\mathrm{S}}, \Theta' \subseteq \Theta\}$.
20: $\sigma_k(\Theta) \leftarrow \sigma(\Theta) - \sum_{\Theta'\in\Gamma} \sigma_k(\Theta')$.
21: $\Theta_k^{\mathrm{S}} \leftarrow \Theta_k^{\mathrm{S}} \cup \langle \Theta, \mu_k(\Theta), \sigma_k(\Theta)\rangle$.
22: **end if**
23: $l_k \leftarrow \min(l_k, l(\mathrm{q}))$.
24: $l_{\mathrm{p}} \leftarrow l_{\mathrm{p}} + l_0$.
25: **return** q. {Answer query q correctly}
26: **end if**

5.3 Simulatability of Algorithms A and B

We now show that both Algorithms A and B satisfy the simulatable auditing model [23]. Note that either algorithm makes the answer/reject decision for a received query q based on q, the query history Q_k, the pre-known set G_k, the prior distribution of the cells, and nothing else. Neither algorithm uses the values of sensitive cells or the answer to q as input. As such, all inputs to Algorithms A and B are public information available to the users. Thus, the answer/reject decisions made by the data warehouse server are fully simulatable by the users. Due to the definition of simulatable auditing [23], the auditor (i.e., the data warehouse server) is simulatable with Algorithms A and B.

Note that due to the simulatable auditing property, the data warehouse server will not disclose any additional private information through query rejections. Thus, the rejected queries need not be taken into account by the inference control algorithm. This is reflected in the design of Algorithms A and B, as the rejected queries are not recorded by the data warehouse server.

5.4 Efficiency of Algorithms A and B

Algorithm A is efficient in terms of both time and space. The time complexity of answering each query is $O(1)$. For each user C_k, the data warehouse server only needs to maintain four parameters:

- μ_k the minimum number of sensitive cells covered by a SUM-like query issued by C_k,
- σ_k, the cumulative sum of the number of pre-known cells covered by SUM-like queries issued by C_k,
- $|Q_k^{\mathrm{S}}|$, the number of SUM-like queries issued by C_k,
- l_{p}, the current cumulative sum of the privacy disclosure level.

The space required is $O(1)$ for each user.

For Algorithm B, the computation of $l_{\max}(q)(\mathrm{q}|Q_k)$ remains the same for MIN-like queries, having time complexity of $O(1)$. For SUM-like queries, in order to further optimize the query availability level for high-dimensional data cubes, we propose to store more information. In particular, for each cuboid covered by a user-issued SUM-like query, two parameters, the number of sensitive cells μ and the number of pre-known cells σ, should be recorded. If the inference control component is implemented independently of the OLAP system, the worst-case time complexity for processing an h-dimensional SUM-like query is $O((n-h)\cdot|Q_k^{\mathrm{S}}|)$ (i.e., when the search for Γ_i in Step 7 is achieved by a linear search on Θ_k^{S}). Nonetheless, note that the storage of μ and σ as well as the processing of SUM-like queries can be readily integrated with the query processing in OLAP systems. In particular, the values of μ and σ can be stored along with the pre-materialized aggregates of a cuboid. Then, the aggregation of μ and σ in Steps 8-9 of Algorithm B can be computed while processing the OLAP query by retrieving and aggregating μ and σ along with other pre-materialized values. With this method, the inference control component is no longer transparent to the OLAP system. Nevertheless, the overhead of inference control can also be significantly reduced.

5.5 Processing of Non-Skeleton Queries

For the ease of understanding, our previous examples and discussions have been focused on skeleton queries. Although skeleton queries are the main focus of an OLAP system, a user may also issue non-skeleton queries, such as range queries, each of which covers more than one value (but fewer than all values) of an attribute. Since the cells covered by a non-skeleton query can always be separated into multiple non-overlapping skeleton queries, an easy method to process a non-skeleton query with distributive aggregate function (e.g., MIN, MAX, and SUM) is to transform the non-skeleton query into a sequence of skeleton ones, and to answer the non-skeleton query iff all (transformed) skeleton ones can be answered. Since the answer to the non-skeleton query can always be derived from the set of answers to the skeleton queries, the safety of private information is guaranteed. Nonetheless, this method may yield low query availability, as the query availability level of the non-skeleton query will also be computed as the sum of that of the skeleton queries.

Fortunately, with our approach, a non-skeleton query can be processed in the same way as *one* skeleton query. The reason can be explained as follows. We consider Algorithm A as an example. Recall from the proof of Theorem 6.1 that SUM auditing in Algorithm A is based on the fact that if t_0 sensitive cells in a skeleton query q cannot be included in the maximum self-supportive subset of q, then the answered queries must cover enough pre-known cells to separate a $t_0 \times (\mu - 1)$ subcube from all other sensitive cells in both dimensions. A critical observation here is that due to Definition 6, if Ω is a self-supportive subset of a skeleton query q, then Ω is also a self-supportive subset of a non-skeleton query q' which satisfies $q \subseteq q'$. Thus, for any given query q', if t_0 sensitive cells in q' cannot be included in the maximum self-supportive subset of q', then the answered queries must cover enough pre-known cells to separate a $t_0 \times (\mu - 1)$ subcube from all other sensitive cells in both dimensions.

Since the computation of $l_{\max}(\mathrm{q}'|Q_k)$ depends solely on the number of sensitive cells in the maximum self-supportive subset of q', and does not depend on whether q' is a skeleton query, we can follow the exact same steps in Algorithm A to compute the level of privacy disclosure for a non-skeleton query. In analogy, Algorithm B can also be readily applied to process non-skeleton queries for n-dimensional data cubes with arbitrary underlying distribution: The only note of caution is that in Steps 6-7 of Algorithm B, Θ_i may be a union of multiple $(n-h-1)$-dimensional cuboids if q is a non-skeleton query, and Γ_i should consist of all cuboids in Θ_k^{S} which belong to any of the cuboids in Θ_i.

6 Theoretical Analysis

In this section, we first prove that both Algorithm A and B achieve Objective O_1 of the inference control problem by rejecting all unsafe queries. Then, we analyze the achievability for Objective O_2 in terms of the query availability level $l_{\mathrm{a}}(C_k)$ for the two algorithms respectively. Due to space limitations, please refer to [41] for the proofs of all theorems in this section.

6.1 Proof of Security

Theorem 6.1. *When Algorithm A is used for the simple 2-dimensional case, for all cells x and all users C_k , there is*

$$l_{\mathrm{p}}(x; Q_k) < l(x). \tag{21}$$

For the n-dimensional case, we have the following theorem illustrating the security of Algorithm B:

Theorem 6.2. *When Algorithm B is used, $\forall x$ and C_k,*

$$l_{\mathrm{p}}(x; Q_k) < l(x). \tag{22}$$

As we can see, since Theorems 6.1 and 6.2 show that our algorithms maintain $l_{\mathrm{p}}(x; Q_k) < l(x)$ for all sensitive cells $x \notin G_k$, and Section 5.3 shows that our algorithms disclose no private information through query rejections, both of Algorithms A and B can prevent the data owner's privacy requirements from being violated by the users, thus achieving Objective O_1 of the problem statement.

6.2 Query Availability Analysis

In this section, we analyze the performance of Algorithms A and B on Objective O_2 in terms of the query availability level. Since our algorithms deal with MIN-like and SUM-like functions separately, we consider the following three cases respectively: 1) all queries are MIN-like, 2) all queries are SUM-like, and 3) a combination of MIN-like and SUM-like queries. For the sake of simplicity, we first present the results for Algorithm A on the simple 2-d case with a $d \times d$ data cube, and then generalize the results to Algorithm B with arbitrary distribution of x and arbitrary privacy requirements $l(x)$.

Theorem 6.3. (MIN-Like Queries, Algorithm A) *For Algorithm A, when all m received queries are MIN-like, the expected number of queries answered by the data warehouse server is at least*

$$m_{\mathrm{M}} \approx \min\left(\frac{l \cdot 2^d}{d}, m\right). \tag{23}$$

Due to the theorem, if all queries are MIN-like, the number of answered queries is determined by the size d of the data cube and the owner-specified threshold l. The larger l or d is, the more queries will be answered. In particular, with a relatively large d (e.g., $d = 10$), all $2d$ MIN-like skeleton queries (i.e., on each row/column) can be answered even when l is small (e.g., $l_{\mathrm{p}} = \min(2d, m)$ when $l \geq 0.196$). This indicates an extremely high level of query availability Algorithm A provides to MIN-like queries.

For analysis on SUM-like queries, we introduce a new parameter μ_0, which is the minimum number of sensitive cells in each row/column of the data cube. Recall that G_k is the set of cells pre-known by user C_k. We have the following theorem regarding the query achievability level of Algorithm A for SUM-like queries.

Theorem 6.4. (SUM-Like Queries, Algorithm A). *For Algorithm A, when all m received queries are SUM-like skeleton queries, the expected number of queries answered by the data warehouse server is at least*

$$m_{\mathrm{S}} \approx \min\left(\ln(4) \cdot l \cdot \frac{d^2 - d - |G_k|}{d}, \frac{(\mu_0 - 1) \cdot d^2}{|G_k|}, m\right).$$

when d is sufficiently large.

As we can see from the theorem, when all queries are SUM-like, the query availability level is determined by not only the data cube size d and the owner-specified threshold l, but also the minimum number of sensitive cells per row/column μ_0 and the total number of pre-known cells $|G_k|$. Similar to MIN-like queries, a larger l or d can increase the number of answered queries. Besides, the larger μ_0 or the smaller $|G_k|$ is, the more queries will be answered. Theorem 6.4 also indicates that Algorithm A is likely to achieve a reasonable query availability level for SUM-like queries in practice. Note that $\mu_0 \cdot d \geq |G_k|$. Thus, the second input to the $\min$ function is likely to be close to d. As a result, when l is small and m is large, the value of m_{S} is determined by the first input, $ln(4) \cdot l \cdot (d - |G_k|/d - 1)$. One can see that even when 25% of all cells are known by a user through external knowledge, and l is as low as 0.5, Algorithm A still guarantees that at least $ln(4) \cdot 0.5 \cdot 0.75 \cdot d/(2d) = 26\%$ of all possible skeleton queries over the data cube.

We now consider mixed MIN-like and SUM-like queries. In this case, it is possible that after one query (e.g., SUM-like) is rejected, many other queries (e.g., MIN-like) can still be answered. Indeed, it is possible for all MIN-like queries to be answered after a substantial amount of SUM-like queries have been rejected due to the differences between the values of $l_{\max}(q|Q_k)$ for MIN-like and SUM-like functions. Therefore, instead of analyzing the total number of queries answered by the data warehouse server, we derive a lower bound on the expected number of answered queries when *the first query rejection occurs*. Suppose that β is the ratio between the number of received MIN-like and SUM-like queries.

$$\beta = \frac{|\{\text{MIN-like queries}\}|}{|\{\text{SUM-like queries}\}|}. \tag{24}$$

Theorem 6.5. (Mixed Queries, Algorithm A). *For Algorithm A, when the received m queries are mixed MIN-like and SUM-like skeleton queries with ratio of β, the expected number of queries answered by the data warehouse server before the first query rejection occurs is at least*

$$m_{\mathrm{MS}} \approx \min\left(\frac{l(\beta+1)2^d(d^2 - |G_k| - d)}{\beta d^3 - \beta d|G_k| - \beta d + d2^{d-1}\log(e)}, (1+\beta)m_{\mathrm{S}}, m\right) \tag{25}$$

$$\approx \min((1+\beta)m_{\mathrm{S}}, m) \tag{26}$$

when d is sufficiently large.

As we can see, when d is sufficiently large, the number of answered queries is predominantly determined by the SUM-like queries, rather than the MIN-like ones. This conforms to the intuition that answers to MIN-like queries lead to rather small increases in l_{p}.

We now extend the results to the n-dimensional cases addressed by Algorithm B. In particular, we consider a $d \times \cdots \times d$ data cube. Let μ_h be the minimum number of sensitive cells covered by an h-dimensional skeleton query on the data cube. Let $f_{\mathrm{S}}(c)$ be the maximum value of $f_{\max}(\mathrm{q}, c) - f_{\min}(\mathrm{q}, c-1)$ for all SUM-like queries q. We have the following theorem.

Theorem 6.6. (Algorithm B). *For Algorithm B, when the received m queries are h-dimensional MIN-like, SUM-like*

skeleton, and mixed skeleton queries, the corresponding lower bounds on the expected number of queries answered by the data warehouse server are

$$m_{\mathrm{M}}(h) \approx \min\left(\frac{\min_x l(x) \cdot 2^{(d^h)}}{d^h}, m\right). \quad (27)$$

$$m_{\mathrm{S}}(h) \approx \min\left(\frac{\min_x l(x) \cdot d^{n-h-1}}{f_{\mathrm{S}}\left(d^h - \frac{|G_k|}{d^{n-h}}\right)}, m, \frac{(\mu_h(hd-h-1)-(hd-h-d))d^{2n-2h-1}}{|G_k|}\right). \quad (28)$$

$$m_{\mathrm{MS}}(h) \approx \min((1+\beta)m_{\mathrm{S}}, m). \quad (29)$$

when d is sufficiently large.

As we can see, Theorems 6.3, 6.4, and 6.5 are special cases of Theorem 6.6 for the simple two-dimensional case. In the n-dimensional case addressed by Algorithm B, the number of answered MIN-like queries increases almost exponentially with the data cube size d or the query dimensionality h (until all MIN-like queries can be answered). For SUM-like queries, however, the smaller h is, the more queries can be answered. Nonetheless, if we consider the percentage of h-dimensional queries answered by the data warehouse server, then the percentage may be increasing with h because generally speaking, $f_{\mathrm{S}}(d^h - |G_k|/d^{n-h})$ is monotonically decreasing with h. For mixed queries, as in the simple case, the number of answered queries is mostly determined by the SUM-like queries, rather than the MIN-like ones.

7 Simulation Results

In this section, we present simulation results on the performance of Algorithms A and B, respectively, and compare the performance of our approach with previous ones proposed for inference control in OLAP. As shown in [40], the previous approaches in [36], [37] achieve their best performance in 2-dimensional data cubes. Thus, we compare the performance of our approach with the previous ones in the 2-dimensional case addressed by Algorithm A.

In particular, for Algorithm A, we conduct the simulation on a $1,000 \times 1,000$ data cube, each cell of which is chosen uniformly at random from $\{0, 1\}$. We consider various numbers of pre-known cells ranging from 0% to 100% of all cells, and assume that the pre-known cells are randomly distributed in the data cube.

We compare the performance of Algorithm A with two previous approaches for inference control in OLAP: the SUM-only approach in [36], [37] and the (MIN, MAX, SUM)-approach in [40]. Since both previous approaches can only eliminate exact disclosure, we set the owner-specified threshold in Algorithm A to be $l = 1$. Note that the original algorithms presented in [36], [37] set an upper-bound threshold on the number of pre-known cells, and answers (all, otherwise nothing) queries if and only if the number of pre-known cells in the data cube is lower than the threshold. In order to make a fair comparison, we make the previous approaches more flexible by allowing the data warehouse server to answer all queries until the number of pre-known cells *covered* by the answered queries exceeds the threshold. This is due to the fact that a pre-known cell not covered by any of the answered queries cannot increase the amount of private information disclosed by answered queries.

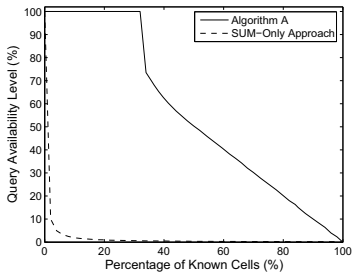

Fig. 1. Comparison with Previous Approaches in [36], [37]

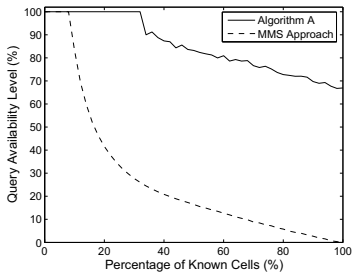

Fig. 2. Comparison with Previous Approach in [40]

Figure 1 shows the comparison results between our proposed approach and the approaches in [36], [37]. Since the previous approaches only support SUM queries, we issue all $2,000$ 1-dimensional SUM skeleton queries of the data cube (i.e., SUM of each row and column) in random order. In the simulation, we investigate the relationship between the query availability level and the percentage of pre-known cells, which ranges from 0% to 100%. As we can see from the figure, while all approaches answer more queries when fewer cells are pre-known by the users, Algorithm A achieves a significantly higher level of query availability than the previous approaches.

Figure 2 shows the comparison results between our proposed approach and the approach in [40] (denoted by MMS approach). Since the previous approach supports MIN, MAX, and SUM queries, we issue all $6,000$ 1-dimensional SUM, MIN, and MAX skeleton queries of the data cube in random order. Again, we investigate the relationship between the query availability level and the percentage of pre-known cells which ranges from 0% to 100%. As we can see from the results, while both approaches can answer almost all queries when the percentage of pre-known cells is low, Algorithm A achieves a significantly higher level of query availability

when more cells in the data cube are pre-known by the users. For example, when 30% of the cells are pre-known, our proposed approach can answer 100% of all queries, while the previous approach in [40] can answer only 27.73% of them. Note that Algorithm A can answer a substantial amount of MIN and MAX queries even if nearly all cells in the data cube are pre-known. This is because that even if a user knows all cells except one sensitive cell that are included in a MIN/MAX query, it may still not be able to derive any private information about the sensitive cell from the query answer. For example, as long as one pre-known cell included in a MAX query has value of 1, the user can derive from its pre-knowledge that the query answer is always 1, and thus cannot infer any additional private information after receiving the query answer.

As we can see from the performance comparison, our proposed approach not only offers better privacy protection by controlling both partial and exact disclosure of private data, but also provides higher query availability than the previous approaches given the same level of privacy protection.

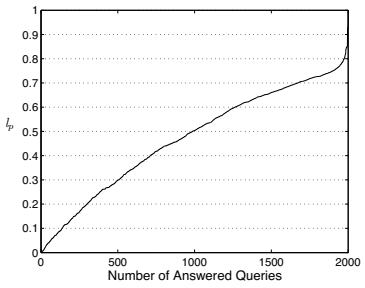

Fig. 3. Change of l_{p} With the Number of Answered Queries in Algorithm A

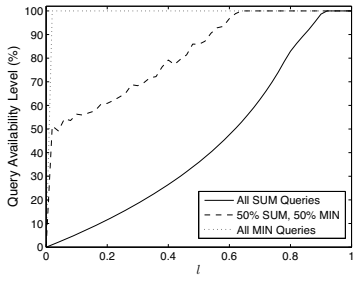

Fig. 4. Query Availability Level of Algorithm A for MIN-like and SUM-like Queries

To evaluate the performance of our approach with different user-specified thresholds $l \in [0, 1]$, in Figure 3, we demonstrate the change of l_{p} with the number of issued query answers (i.e., $|Q_k|$) in the execution of Algorithm A. In particular, based on the above 2-dimensional settings, we set the percentage of pre-known cells to be 50%, and issue in random order $2,000$ 1-dimensional skeleton queries which include 50% MIN-like and 50% SUM-like queries. Note that Figure 3 also demonstrates how the degree of uncertainty on a sensitive cell (i.e., $1 - l_{\mathrm{p}}$) decreases as aggregate values become available

As we can see, the value of l_{p} increases fairly slowly (almost linearly) when $|Q_k|$ is low, but grows much faster when more than $1,900$ query answers have been issued. This conforms to the intuition that the more query answers a (malicious) user has received, the more private information it can infer from a new query answer. Another observation we can make from Figure 3 is that the query availability level of our approach is not very sensitive to the user-specified threshold l when l is large. For example, our approach can answer $1,999$ queries when $l = 0.95$. When l is reduced to 0.85 and 0.75, our approach can still answer $1,993$ and $1,902$ queries, respectively.

We also evaluate the performance of Algorithm A in terms of the query availability level 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 conduct the simulation on the above 2-dimensional settings when the own-specified threshold l varies from 0 to 1, and set 25% of the cells (randomly chosen) to be pre-known. In the simulation, we issue $2,000$ 1-dimensional skeleton queries in random order. In the case where there are half SUM-like and half MIN-like queries, the aggregation function of each query is chosen uniformly at random from SUM and MIN.

The results are shown in Figure 4. As we can see, Algorithm A can answer more MIN-like operations than SUM-like ones, especially when l is small. For example, when $l = 0.4$, Algorithm A can answer 100% of all MIN-like queries, 79.2% of queries that are half MIN-like and half SUM-like, and 26.5% of all SUM-like queries. This is consistent with our theoretical results in Section 6. Figure 4 also shows that the higher owner-specified threshold l is, the more queries can be answered by Algorithm A. In particular, when $l = 1$ (i.e., only exact disclosure is of concern), Algorithm A can answer almost all queries regardless of their aggregation functions.

We now evaluate the performance of Algorithm B in the n-dimensional case with arbitrary distribution of cells. In particular, we consider a 4-dimensional $100 \times 100 \times 100 \times 100$ data cube, each cell of which is randomly generated from a normal distribution with mean of 0 and variance of 1. We assume that each cell is described with 2-bits accuracy (i.e., 2 bits to the right of the decimal point). As such, the entropy of each cell is $H(x) = 2 + \log(\sqrt{2\pi e}) \approx 4.0$ bits. We set $\min_x l(x) = 0.8$, and conduct the experiment when the percentage of pre-known cells varies from 1% to 90%. In the experiment, we issue MIN and SUM queries randomly chosen from all 4×10^6 1-d, $60,000$ 2-d, and 400 3-d subcubes corresponding to skeleton queries on the data cube.

We find that with Algorithm B, the query availability level for MIN-like queries is always 100%. The query availability level for SUM queries is shown in Figure 5.

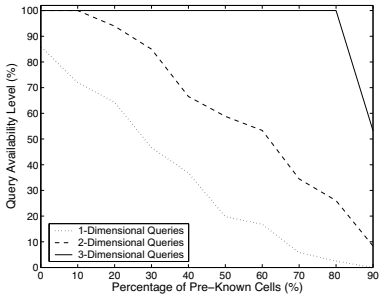

Fig. 5. Performance of Algorithm B on Answering SUM queries in n-Dimensional Cases

As we can see from the figure, Algorithm B can answer most of the queries when the percentage of pre-known cells is low. For example, when 10% cells are pre-known by the users, our approach can answer 100% 2-dimensional and 3-dimensional queries, as well as 71.9% 1-dimensional queries. Note that Algorithm B answers a higher percentage of high-dimensional queries, partially due to the higher number of possible low-dimensional queries compared with the higher-dimensional ones.

8 Extensions and Discussions

In this section, we present the extension of our paper to: 1) defend against malicious users that collaborate with each other, and 2) process a batch of queries submitted simultaneously.

8.1 Defense Against Colluding Users

A traditional challenge for inference control (e.g., in statistical databases) is the defense against colluding users. Since the answer/reject decision for a query is made based on a user's query history Q_k and pre-known cells G_k, privacy breaches may occur when multiple users collude with each other to share their received query answers and pre-known cells, in order infer additional private data from the shared information. A possible way to defend against colluding users is to conservatively assume that every user learns all other users' query histories ($\bigcup_k Q_k$) and pre-known cells ($\bigcup_k G_k$), and to ensure $l_{\mathrm{p}}(x; \bigcup_k Q_k | \bigcup_k G_k) < l(x)$ for all cells x . Nonetheless, this strategy will significantly reduce the number of answered queries. For example, consider the case where the first query q issued by a user has disclosure level $l_{\max}(q) = l(x) - \epsilon$, where $l(x)$ is the owner-specified threshold on privacy disclosure, and $\epsilon > 0$ is arbitrarily close to 0. This query will be answered because it does not violate $l(x)$. Nonetheless, no future query issued by any user can be answered because $l_{\max}(q)$ takes almost all the amount of disclosure tolerable by the data owners.

In order to defend against colluding users while maintaining a reasonable level of query availability, we propose two methods, to be applied when the defender knows and does not know a (small) maximum possible number of colluding users, respectively. The first method applies when the defender knows there are up to h colluding users. The basic idea is to maintain global upper bounds for μ_k, σ_k, and $|Q_{\mathrm{S}}^k|$, and enforce an upper-bound limit of $l(x)/h$ for all users. Consider Algorithm A as an example (Algorithm B can be revised in analogy). We replace μ_k, σ_k, and $|Q_{\mathrm{S}}^k|$ in Algorithm A by $\mu = \min_k \mu_k$, the minimum number of sensitive cells covered by a SUM-like query issued by any user, $\sigma = h \cdot \max_k \sigma_k$, the maximum number of pre-known cells covered by SUM-like queries issued by h users, and $|Q^{\mathrm{S}}| = h \cdot \max_k |Q_k^{\mathrm{S}}|$, the maximum number of SUM-like queries issued by h users, respectively. Then, we ensure that the cumulative sum l_{p} in Algorithm A is smaller than l/h. One can see that no h colluding users can violate the owner-specified threshold l. Meanwhile, the time and space complexity of Algorithm A remains $O(1)$, because the values of $\min_k \mu_k$, $\max_k \sigma_k$, and $\max_k |Q_k^{\mathrm{S}}|$ can be updated after each query is answered. Figure 6 shows the change of l_{p} with the number of answered queries per user when h ranges from 1 to 7, using the same experimental settings as Figure 4 (i.e., each user knows 50% of all cells as pre-knowledge; the query workload consists of $2,000$ 1-dimensional skeleton queries which include 50% MIN-like and 50% SUM-like queries). One can see that although the number of queries answerable to each user becomes much smaller (which is expected because the users may share their query answers), the total number of queries answered to all users remains fairly high (e.g., $1,575$ queries when $l_{\mathrm{p}} = 1$ and $h = 3$).

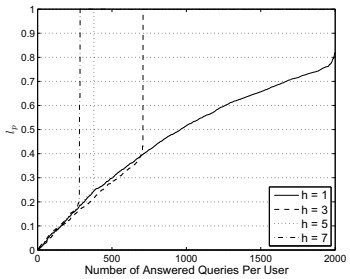

Fig. 6. Defense Against Colluding Users when h is Known

If the maximum number of colluding users is unknown to the defender, or if the value of h is too large for the above method to answer a reasonable number of queries, then we propose a second method, which is essentially a static version of our inference control approach. The basic idea of the static approach is to determine a *safe set* of queries Q_0 *offline* before any query is actually received, such that no privacy requirement is violated even if a user receives the answers to all queries in Q_0 and has $\bigcup_k G_k$ as pre-knowledge (i.e., $l_{\mathrm{p}}(x; Q_0 | \bigcup_k G_k) < l(x)$ for all cells x). Note that if a query set is safe, then each of its subsets is also safe. At runtime, when the data warehouse server receives a query, it answers the query if and only if the query is in the predetermined safe set.

Depending on the requirements of OLAP applications (e.g., whether MIN-like and SUM-like queries are equally important), there are many possible ways to construct the safe set Q_0. Generally speaking, the safe set should include as many queries as possible. In order to do so, we must exclude certain "privacy-divulging" queries (e.g., a query which consists of mostly pre-known cells) which lead to a substantial increase in the estimated privacy disclosure level $l_{\max}(Q_k)$. For example, in the simple 2-dimensional case addressed by Algorithm A, by excluding queries with the number of pre-known cells greater than $\min(d_1/2, d_2/2)$, we can include all other queries in the safe set when $l = 1$ (this follows from Step 3 in the proof of Theorem 6.1).

The insight behind the construction of safe set is that a few privacy-divulging queries (e.g., the above-discussed query with $l_{\max}(q) = l(x) - \epsilon$) cause the most increases on $l_{\max}(Q_k)$. While dealing with individual users (i.e., without collusion), it may not be effective to preemptively reject such queries due to the following three reasons: 1) The users may only issue a small number of queries, so that the privacy disclosure level will not exceed the threshold anyway, 2) The privacy-divulging queries may be important for the user's legitimate intent, and 3) Algorithms A and B are fully simulatable by users. Thus, if the privacy-divulging query is not important to the user, it may not send it to the data warehouse server due to the predicated decisions of inference control.

Nonetheless, when we aim to defend against colluding users, answering one of such privacy-divulging queries will increase the disclosure level for *all* users, therefore significantly reducing the total number of answered queries. Since a user has no control on the queries issued by the other users, we need the static approach to serve as a global control on the amount of private information disclosed by each query. In particular, the static approach preemptively rejects privacy-divulging queries by excluding them from the pre-determined safe set. This allows the data warehouse server to answer a large portion of queries even with rigid privacy requirements and a high percentage of known cells (by all users).

8.2 Answering a Batch of Queries

In the previous part of the paper, we assume that each user submits one query at a time to the data warehouse server. Thus, the data warehouse server must decide whether to answer the query or to reject it before receiving the next query. We now consider the extension of our results to cases where a user can submit multiple queries simultaneously to the data warehouse server. Apparently, our algorithms can be used in these cases without any change if the data warehouse server processes the submitted queries in a serialized manner. Nonetheless, we shall show that it is also possible to further optimize the performance of our approach in these cases because the data warehouse server learns more information about the upcoming queries.

Recall that our inference control approach derives an upper-bound estimate of privacy disclosure $l_{\max}(q|Q_q)$ for each received query q, and uses the estimate to update the system disclosure level $l_{\max}(Q_k)$. As we mentioned in Section 8.1, a few privacy-divulging queries cause the most increases on $l_{\max}(Q_k)$. Thus, when the data warehouse server receives a batch of queries that cannot all be answered due to privacy requirements, the data warehouse server may identify and reject the privacy-divulging queries in order to answer more of the other queries in the batch. For example, the data warehouse server may choose to reject queries with the highest $l_{\max}(q|Q_q)$. For SUM-like queries, the data warehouse server may also choose to reject queries with a high number of pre-known cells $r_k(\mathrm{q})$ in order to maintain a low value of μ, thereby reducing the value of $l_{\max}(q|Q_q)$ for future queries. Note that the value of $l_{\max}(Q_k)$ computed by our approach also depends on the order of queries in Q_k. Thus, the data warehouse server may also adjust the query order in the received batch in order to further increase the number of answered queries. we shall investigate the optimal ordering of received queries in our future work.

9 Related Work

As we mentioned in Section 1, there are two kinds of approaches that have been proposed for privacy protection in OLAP, namely inference control and input/output perturbation. In this section, we review related work in these two categories.

Many inference control algorithms have been proposed in the literature for not only OLAP [26], [34], [35], [37] but also statistical databases [1], [9], [10], [11], [16], [22] and the general query auditing problem [13], [23], [24], [29], [30]. Both online and offline versions of the problem have been investigated. While the online version determines the safety of a received query based on the query history [9], [13], [30], the offline version aims to determine whether or not a given sequence of queries $\mathrm{q}_1, \ldots, \mathrm{q}_m$ is safe [9], and if not, how to find the maximum safe subset of queries [11]. The ideal objective of inference control is to eliminate the inference of individual data points from query answers while minimizing the number of rejected queries. Nonetheless, a special case of the optimal inference control problem (with mixed SUM and MAX queries) in statistical databases have been proven to be NP-hard [11]. The optimal boolean query auditing problem has also been proven to be co-NP-complete [24]. As such, the majority of existing work on inference control makes a tradeoff between efficiency and query availability. Due to efficiency concerns, many existing inference control algorithms for OLAP make query answer/reject decisions based on the cardinality of the data cube (i.e., the number of pre-known and sensitive cells) [36], [37], [40]. Most only address the exact disclosure of sensitive cells [26], [34], [35], [37] and only support SUM queries [26], [35], [36], [37].

Input/output perturbation approaches aim to protect privacy by adding random noise to either the input data to the data warehouse [4], [32] or the (output) query answers [16]. There are variations of this approach which add noise to the input data with linear or nonlinear transformations [6], cluster the input data points into a number of mutually exclusive groups and replace the individual values with group-wise average values (i.e., microaggregation) [15], or generalize query answers into interval values [18]. Perturbation-based approaches have also been studied in a broader scope, especially for privacy-preserving data mining (e.g., [3]). For privacy-preserving OLAP, when the input data are perturbed, algorithms have been proposed to reconstruct the (original) outputs of aggregate functions from the subcubes of perturbed data [4], [32]. Although input/output perturbation allows the data warehouse server to answer all received queries, a tradeoff has to be made between the accuracy of query answers and the protection (i.e., defense against partial disclosure) of private information. Again, most existing approaches only support limited aggregated functions, such as COUNT-only [4] and SUM-only [32].

In another related direction, the privacy models of inference control and query auditing have been studied. The simulatable auditing model was proposed [23] to capture the leakage of private information through query denials. A probabilistic measure was also introduced [23] to capture the partial disclosure of private data. As we have shown in Section 5.3, our proposed approach follows the simulatable auditing model while addressing the partial disclosure of private information.

10 Final Remarks

In this paper, we address the protection of private data in OLAP systems. Most existing inference control approaches only address exact disclosure, and eliminate from consideration an important class of privacy breaches where partial information about a private data point is disclosed. We propose an information-theoretic inference control approach that protects private information against both exact and partial disclosure. In particular, our approach guarantees that the level of privacy disclosure cannot exceed thresholds specified by data owners. Compared with previous approaches, our approach provides more effective privacy protection while maintaining a higher level of query availability. We conclude with future directions.

- General Query Auditing Problems: In this paper, we are focusing on the processing of multi-dimensional OLAP queries. A future direction is to extend the information-theoretic framework to more general query auditing problem which allows arbitrary queries on the private data. It would also be interesting to investigate ways to improve the query availability level when the private data are frequently updated.
- Real-world Privacy Measure: The privacy versus data utility tradeoff in privacy-preserving data processing requires accurate measurement of privacy protection. Our privacy measure in the paper enables the information-theoretic formulation but cannot capture certain extreme-case privacy breaches, which we have to address by integrating our approach with the input/output perturbation method. A comprehensive study on real-world privacy measurement would be a significant step toward improving the performance of privacy-preserving data processing techniques.
- Integration of Data Collection, Inference Control, and Information Sharing: Existing research on privacy-preserving data processing has addressed three system settings separately: 1) data collection from distributed sources for data mining (e.g., [3]), 2) inference control on data warehouse (e.g., this paper), and 3) data sharing across private databases (e.g., [2]). Many systems need a seamless integration of all three scenarios, yet there is limited research that has addressed this need. We proposed an integrated architecture for the three scenarios [39]. As we demonstrate in the technical report [41], our inference control approach can be effectively integrated with input/output perturbation, which is also a popular choice for data collection. More extensive research is needed to pave the way for effective and efficient integration of the three scenarios.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments. This work was supported in part by the National Science Foundation under grants 0324988, 0329181, 0721571, 0808419, 0845644, 0852673, 0852674, and 0915834. Any opinion, findings, conclusion, and/or recommendation in this material, either expressed or implied, are those of the authors and do not necessarily reflect the views of the sponsor listed above. The authors would like to thank Ms. Larisa Archer for her editorial help with the paper.

References

[1] N. R. Adam and J. C. Worthmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515–556, 1989.

[2] R. Agrawal, A. Evfimievski, and R. Srikant, "Information sharing across private databases," in *Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data*, 2003, pp. 86–97.

[3] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 19th ACM SIGMOD International Conference on Management of Data*, 2000, pp. 439–450.

[4] R. Agrawal, R. Srikant, and D. Thomas, "Privacy preserving OLAP," in *Proceedings of the 25th ACM SIGMOD International Conference on Management of Data*, 2005, pp. 251–262.

[5] L. L. Beck, "A security machanism for statistical database," *ACM Transactions on Database Systems*, vol. 5, no. 3, pp. 316–3338, 1980.

[6] R. Brand, "Microdata protection through noise addition," vol. 2316, pp. 97–116, 2002.

[7] L. Brankovic, P. Norak, M. Miller, and G. Wrightson, "Usability of compromise-free statistical databases," in *Proceedings of the 9th International Conference on Scientific and Statistical Database Management*, 1997, p. 144154.
[8] B. Chen, K. LeFevre, and R. Ramakrishnan, "Privacy skyline: Privacy with multidimensional adversarial knowledge," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, 2007, pp. 770–781.
[9] F. Chin, "Security problems on inference control for sum, max, and min queries," *Journal of the ACM*, vol. 33, no. 3, pp. 451–464, 1986.
[10] F. Chin and G. Ozsoyoglu, "Auditing for secure statistical databases," in *Proceedings of the ACM '81 Conference*, 1981, pp. 53–59.
[11] ——, "Auditing and inference control in statistical databases," *IEEE Transactions on Software Engineering*, vol. SE-8, no. 6, pp. 574–582, 1982.
[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.
[13] D. Dobkin, A. K. Jones, and R. J. Lipton, "Secure databases: protection against user influence," *ACM Transactions on Database Systems*, vol. 4, no. 1, pp. 97–106, 1979.
[14] J. Domingo-Ferrer, *Inference Control in Statistical Databases*. New York: Springer, 2002.
[15] J. Domingo-Ferrer and J. M. Mateo-Sanz, "Practical data-oriented microaggregation for statistical disclosure control," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 189–201, 2002.
[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the 3rd Theory of Cryptography Conference*, 2006, pp. 265–284.
[17] A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2003, pp. 211–222.
[18] R. Gopal, R. Garfinkel, and P. Goes, "Confidentiality via camouflage: the CVC approach to disclosure limitation when answering queries to databases," *Operations Research*, vol. 50, pp. 501–516, 2002.
[19] P. P. Griffiths and B. W. Wade, "An authorization mechanism for a relational database system," *ACM Transactions on Database Systems*, vol. 1, no. 3, pp. 242–255, 1976.
[20] J. Han and M. Kamber, *Data Mining Concepts and Techniques*, 2nd ed. Morgan Kaufmann, 2006.
[21] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, "Flexible support for multiple access control policies," *ACM Transactions on Database Systems*, vol. 26, no. 2, pp. 214–260, 2001.
[22] J. B. Kam and J. D. Ullman, "A model of statistical databases and their security," *ACM Transactions on Database Systems*, vol. 2, no. 1, pp. 1–10, 1977.
[23] K. Kenthapadi, N. Mishra, and K. Nissim, "Simulatable auditing," in *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2005, pp. 118–127.
[24] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "Auditing boolean attributes," in *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2000, pp. 86–91.
[25] K. Lefevre, D. J. Dewitt, and R. Ramakrishnan, "Incognito: efficient full-domain k-anonymity," in *Proceedings of the 25th ACM SIGMOD International Conference on Management of Data*, 2005, pp. 49–60.
[26] 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.
[27] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern, "Worst-case background knowledge for privacy-preserving data publishing," in *Proceedings of the IEEE 23rd International Conference on Data Engineering*, 2007, pp. 126–135.
[28] G. Miklau and D. Suciu, "A formal analysis of information disclosure in data exchange," *Journal of Computer and System Sciences*, vol. 73, no. 3, pp. 507–534, 2007.
[29] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards robustness in query auditing," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 151–162.
[30] S. P. Reiss, "Security in databases: A combinatorial study," *Journal of the ACM*, vol. 26, no. 1, pp. 45–57, 1979.
[31] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
[32] Y. Sung, Y. Liu, H. Xiong, and A. Ng, "Privacy preservation for data cubes," *Knowledge and Information Systems*, vol. 9, no. 1, pp. 38–61, 2006.
[33] United States Department of health and human services, Office for civil rights, "Summary of the HIPAA privacy rule," 2003.
[34] 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.
[35] 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.
[36] 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.
[37] ——, "Cardinality-based inference control in data cubes," *Journal of Computer Security*, vol. 12, no. 5, pp. 655 – 692, 2004.
[38] ——, "OLAP means on-line anti-privacy," Center for Secure Information Systems, Tech. Rep. CSIS-TR-03-06, 2003.
[39] N. Zhang and W. Zhao, "Privacy-preserving data mining systems," *IEEE Computer*, vol. 40, no. 4, pp. 52–58, 2007.
[40] 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.
[41] N. Zhang and W. Zhao, "An information-theoretic approach for privacy protection in OLAP systems," TR-GWU-CS-09-003, Department of Computer Science, George Washington University, Tech. Rep., 2009.

Dr. Nan Zhang is an Assistant Professor of Computer Science at the George Washington University. He received the B.S. degree from Peking University in 2001 and the Ph.D. degree from Texas A&M University in 2006, both in computer science. His current research interests include security and privacy issues in databases and computer networks, in particular privacy and anonymity in data collection, publishing, sharing, and wireless network security and privacy.

Dr. Wei Zhao is currently the Rector of the University of Macau. Before joining the University of Macau, he served as the Dean of the School of Science at Rensselaer Polytechnic Institute. Between 2005 and 2006, he served as the director for the Division of Computer and Network Systems in the US National Science Foundation when he was on leave from Texas A&M University, where he served as Senior Associate Vice President for Research and Professor of Computer Science. Dr. Zhao completed his undergraduate program in physics at Shaanxi Normal University, Xian, China, in 1977. He received the MS and PhD degrees in Computer and Information Sciences at the University of Massachusetts at Amherst in 1983 and 1986, respectively. Since then, he has served as a faculty member at Amherst College, the University of Adelaide, and Texas A&M University. As an elected IEEE fellow, Wei Zhao has made significant contributions in distributed computing, real-time systems, computer networks, and cyber space security.