Auditing Interval-Based Inference*

Yingjiu Li, Lingyu Wang, X. Sean Wang, and Sushil Jajodia

Center for Secure Information Systems, George Mason University
Fairfax VA 22030-4444, USA
{yli2,lwang3,xywang,jajodia}@gmu.edu

Abstract. In this paper we study the feasibility of auditing *interval-based inference*. Sensitive information about individuals is said to be compromised if an accurate enough interval, called *inference interval*, is obtained into which the value of the sensitive information must fall. Compared with auditing exact inference that is traditionally studied, auditing interval-based inference is more complicated. Existing auditing methods such as audit expert do not apply to this case. Our result shows that it is intractable to audit interval-based inference for bounded integer values; while for bounded real values, the auditing problem is polynomial yet involves complicated computation of mathematical programming. To further examine the practicability of auditing interval-based inference, we classify various auditing methods into three categories: exact auditing, optimistic auditing, and pessimistic auditing. We analyze the trade-offs that can be achieved by these methods among various auditing objectives: inference security, database usability, and auditing complexity.

1 Introduction

Conflicts exist between individual rights to privacy and society's needs to know and process information [24]. In many applications, from census data through statistical databases to data mining and data warehousing, only aggregated information is allowed to be released while sensitive (private) information about individuals must be protected. However, *inference problem* exists because sensitive information could be inferred from the results of aggregation queries.

Depending on what is exactly inferred from the queries about aggregated information, various types of inference problem have been identified and studied. See surveys [10,1,14,12]. Most existing works deal with either the inference of exact values, referred to as *exact inference* (or exact disclosure), or statistical estimators, referred to as *statistical inference* (or partial disclosure).

Interval-Based Inference We study another type of inference. Consider a relation with attributes (*model*, *sale*) where attribute *model* is public and attribute *sale* is sensitive. Assume that a user issues two sum queries and that the correct answers are given as following: (1) The summed sales of model A

* This work was partially supported by the National Science Foundation under grant CCR-0113515.

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 553–568, 2002.

and model C are 200; (2) The summed sales of model A and model B are 4200. Clearly, from those two queries, the user cannot infer any exact value of *sale*. However, due to the fact that attribute *sale* is positive, first the user is able to determine from query (1) that both the sales of model A and the sales of model C are between zero and 200, and then from query (2) that the sales of model B are between 4000 and 4200. The length of interval $[4000, 4200]$, which is the maximum error of user's estimation of the sales of model B, is less than 5% of actual sales of that model. This certainly can be regarded as a disclosure of sensitive information about model B.

The above example indicates that a database attacker is able to infer an accurate enough interval when he or she may not infer an exact value of the sensitive attribute. We call this type of inference *interval-based inference* and the interval *inference interval*.

Interval-based inference poses a different threat to individual privacy in comparison with exact inference and statistical inference. On one hand, exact inference can be regarded as a special case of interval-based inference where the length of inference interval is equal to zero. This means that exact inference certainly leads to interval-based inference while interval-based inference may still occur in the absence of exact inference. On the other hand, the statistical inference is studied in the context that random perturbation, i.e., noise, is added to sensitive individual data. If the variance of the noise that is added is no less than a predetermined threshold, then the perturbed database is considered "safe" in terms of statistical inference. Because the perturbation is probabilistic, it is possible – no matter how large the variance of the noise is – that some perturbed values are close enough to the unperturbed ones, leading to interval-based inference.

Inference Control Techniques For controlling exact inference, many restriction based techniques have been studied (in the statistical database literature) which include restricting the size of a *query set* (i.e., the entities that satisfy a single query) [17,13], controlling the overlap of query sets [15], suppressing sensitive data cells in a released table of statistics (i.e., query results) [9], partitioning data into mutually exclusive chunks and restricting each query set to some undivided data chunks [6,7], and (closer to our concerns in this paper) auditing all queries in order to determine whether inference is possible [8,5,20,23]. For controlling statistical inference, some perturbation based techniques have been studied which include adding noise to source data [27,28], changing output results [2], altering database structure [25], or sampling data to answer queries[11].

Auditing We study the auditing approach in this paper. By auditing, all queries made by each user are logged and checked for possible inference before the results of new queries are released. For auditing exact inference of arbitrary queries with real-valued data, one of the best results was given by Chin and Özsoyoglu [8] in a system called *audit expert*. Audit expert uses a binary matrix to efficiently describe the knowledge about the sensitive attributes (e.g., *sale*), where each row of the matrix represents a query; each column, a database record; and each element, whether the record is involved in the query. Audit expert transforms

the matrix by elementary row operations to a standard form and concludes that exact inference exists iff at least one row contains all zeros except one column. It has been shown that it takes audit expert no more than $O(n^2)$ time to process a new query where n is the number of individuals or database records, and no more than $O(mn^2)$ time to process a set of m queries. Recall that audit expert deals with exact inference; one may ask natural questions about interval-based inference: *Is it possible to adapt audit expert to audit interval-based inference?* If not: *Is it tractable to audit interval-based inference?* Answering these questions will shed new lights on auditing and inference control studies.

As pointed out in [20], most of works in this area, including audit expert, assume that the confidential data are real-valued and essentially unbounded. In real world applications, however, data may have maximum or minimum values that are fixed a priori and attainable. In such a case, traditional auditing methods such as audit expert are inadequate, especially for protecting databases against interval-based inference. One reason is that such methods do not take the boundary information into account, which has a significant impact on the inference. To illustrate this, consider a single sum query with each response represented by $\sum_{i=1}^{n} a_i x_i = b$, where a_i is either one or zero, and each real variable x_i is bounded: $l_i \leq x_i \leq r_i$, $1 \leq i \leq n$, then the k-th variable x_k must fall into the following interval I_k:

$$I_k = \begin{cases} [max(l_k, b - \sum_{i \neq k}^{n} a_i r_i), min(r_k, b - \sum_{i \neq k}^{n} a_i l_i)] & \text{, if } a_k = 1 \\ [l_k, r_k] & \text{, if } a_k = 0 \end{cases}$$

For example, given $x_1 + x_2 = 5$ and $1 \leq x_1, x_2 \leq 3$, we have $2 \leq x_1, x_2 \leq 3$ and $I_1 = I_2 = [2, 3]$. If the length of the interval I_k is less than a predetermined value (e.g., 5% of the actual value of x_k), then variable x_k can be considered as compromised in terms of interval-based inference. This shows that even a single sum query of multiple variables may be vulnerable to interval-based inference, while audit expert will indicate "safe" in this case.

Another reason is the impact of data types. Recall that audit expert deals with real-valued data. In fact, discrete-valued data (Boolean data or integer data) make the auditing problem more difficult. For example, regarding exact inference, Kleinberg et al. [20] recently proved that auditing Boolean attributes is a NP-hard problem. Hence, as for the problem of auditing interval-based inference, an interesting question is: *What is the complexity for auditing interval-based inference with different types of data?*

Our study answers the above questions. In section 2, we define interval-based inference and formulate the problem of auditing interval-based inference. In section 3, we investigate the particular challenges presented by interval-based inference, boundary information, and different data types. Our result shows that it is intractable to audit the interval-based inference for bounded integer values; whereas for bounded real values, the auditing problem is polynomial yet involves complicated computation of mathematical programming. In section 4, we further examine the practicability of auditing interval-based inference. First we classify various auditing systems into three categories: exact auditing system, optimistic

auditing system, and pessimistic auditing system. Then we analyze the trade-offs that can be achieved by these auditing systems among different auditing objectives: inference security, database usability, and auditing complexity. Finally we point out some promising future directions based on our discussions.

2 Problem Formulation

In this section, we formulate the auditing problem. Consider an attribute with n individual values $x_1^*, \ldots, x_n^*$ stored in a database. Denote $x^* = (x_1^* \ldots x_n^*)^T$. A *sum query* (or *query* for short) is defined to be a subset of the individual values in vector x^* and the *query response* the sum of the values in the specified subset.

From the point of view of database users, each individual value x_i^* is secret and thus a variable (to be inferred), denoted by x_i. Let $x = (x_1 \ldots x_n)^T$. To model that database users or attackers may have a priori knowledge about variable bounds, we assume that $l_i \leq x_i \leq r_i$, where l_i, r_i are the lower bound and upper bound of x_i, respectively (l_i and/or r_i can be infinite). We use $l \leq x \leq r$ to denote the boundary information $\{l_i \leq x_i \leq r_i | 1 \leq i \leq n\}$.

By above notation, *each query with its response w.r.t.* x^* can be expressed by a linear equation $a_i^T x = b_i$, where a_i is an n-dimensional vector whose component is either one or zero, and b_i is a scalar that equals to $a_i^T x^*$. Similarly, *a set of* m *queries with responses w.r.t.* x^* can be expressed by a system of linear equations $Ax = b$, where A is an $m \times n$ matrix whose element is either one or zero, and b is an m-dimensional vector that equals to Ax^*.

Given a set of queries with responses w.r.t. x^*, $Ax = b$, and the boundary information $l \leq x \leq r$, exact inference and interval-based inference can be defined as follows.

Definition 2.1. (Exact Inference of x_k) In all the solutions for x that satisfy $Ax = b$ and $l \leq x \leq r$, variable x_k has the same value x_k^*.

Definition 2.2. (Interval-Based Inference of x_k) There exists an interval $I_k = [x_k^{min}, x_k^{max}]$ such that (i) the length of the interval I_k, $|I_k| = x_k^{max} - x_k^{min}$, satisfies $|I_k| < \epsilon_k$; (ii) in all the solutions for x that satisfy both $Ax = b$ and $l \leq x \leq r$, variable x_k has values in the interval I_k; (iii) x_k^{min}, x_k^{max} and x_k^* are three of these values (in the interval I_k), where ϵ_k is a predefined threshold called *tolerance level*.

In the above definition, tolerance level ϵ_k (usually $\epsilon_k \ll r_k - l_k$, $k = 1, \ldots, n$) is used to indicate whether variable x_k is compromised in terms of interval-based inference. For example, we may set $\epsilon_k = 5\% \times x_k^*$. Denote $\epsilon = (\epsilon_1 \ldots \epsilon_n)^T$. Interval $I_k = [x_k^{min}, x_k^{max}]$ is called the *k-th inference interval* (or *inference interval* for short). The endpoints x_k^{min} and x_k^{max} of I_k can be derived as the optimal objective values of the following mathematical programming problems (MPs) $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$, respectively:

$\mathcal{P}^{min}[k]$:	minimize	x_k	$\mathcal{P}^{max}[k]$:	maximize	x_k
	subject to	$Ax = b$		subject to	$Ax = b$
		$l \leq x \leq r$			$l \leq x \leq r$

Due to the fact that the feasible set $P = \{x|Ax = b, l \leq x \leq r\}$ of the above MPs is a nonempty bounded polyhedron (note x^* must be in P), the values x_k^{min} and x_k^{max} must exist uniquely.

In the case of $\epsilon_k = 0$, variable x_k has the same value x_k^* in all the solutions that satisfy $Ax = b$ and $l \leq x \leq r$. Therefore, interval-based inference is more general than exact inference.

Problem of Auditing Interval-Based Inference Given a set of queries with responses w.r.t. x^*, $Ax = b$, boundary information $l \leq x \leq r$, and tolerance level ϵ, show there is no interval-based inference for any variable x_k (that is, $|I_k| \geq \epsilon_k$). Specifically, we study the following two problems:

1. *Auditing integer variables*, where x, l, r, b are restricted to be integer vectors. In this case, the corresponding MPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$ are integer programmings (IPs).
2. *Auditing real variables*, where x, l, r, b are real vectors. In this case, the MPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$ are linear programmings (LPs).

3 Auditing Interval-Based Inference

We study the complexity of auditing interval-based inference. The complexity results are given with respect to the number of queries (m) and the number of database records (n). By default, we consider arbitrary queries ($Ax = b$) w.r.t. a single sensitive attribute[1] (x^*). For auditing integer variables, we also consider one-dimensional range queries.

Proposition 3.1. *Auditing integer variables is NP-hard.*

Proof: Our proof is based on the result that *auditing exact inference of Boolean variables*[2] is NP-hard [20]. Auditing integer variables is NP-hard due to restriction to the problem of auditing exact inference of Boolean variables by allowing only instances in which $l = (0, \ldots, 0)^T$, $r = (1, \ldots, 1)^T$, and $\epsilon_k = 0$ $(k = 1, \ldots, n)$. □

Proposition 3.1 indicates that it is intractable for auditing integer variables of arbitrary queries. This also implies that solving each IP $\mathcal{P}^{min}[k]$ or $\mathcal{P}^{max}[k]$ is NP-hard (Note that the NP-hardness of $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ cannot be derived directly from the fact that general IP is NP-hard since $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$

[1] The setting may involve multiple nonsensitive attributes since they are known to database users. It can also be extended to the queries that even involve multiple sensitive attributes if the auditing system treats them independently; that is, each time a single sensitive attribute is audited while the others are treated as "nonsensitive." We do not consider inferring information using other database features such as schema constraints and functional dependencies. The reader is referred to [4,3] for the study in this aspect.

[2] Auditing exact inference of Boolean variables can be described as follows: Given n 0-1 variables $\{x_1, \ldots, x_n\}$ and a set of sum queries $Ax = b$, show there is no variable x_k $(1 \leq k \leq n)$ such that in all 0-1 solutions of $Ax = b$, variable x_k has the same value.

are special IP problems) in that (i) all $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(k = 1, \ldots, n)$ are equivalent problems; (ii) if anyone of them can be solved in polynomial time, then all can be solved in polynomial time; therefore, all inference intervals I_k $(k = 1, \ldots, n)$ can be computed in polynomial time, which is a contradiction to proposition 3.1.

Although proposition 3.1 indicates that it is theoretically difficult (i.e., NP-hard) to audit integer variables, it does not necessarily mean that the problem is practically unsolvable. People have developed various methods such as branch-and-bound, cutting planes, and heuristics (see, e.g., [19]) to solve IP problems in practice. Such methods can be used in auditing. On the other hand, proposition 3.1 holds in the case of arbitrary queries, the following result shows that it is polynomial in the case of *one-dimensional range queries*. A set of one-dimensional range queries with responses w.r.t. x^* can be defined as $Ax = b$, where the matrix A holds the *consecutive ones property* (i.e., the ones in each row must be consecutive) and $b = Ax^*$.

Proposition 3.2. *For one-dimensional range queries, auditing integer variables is polynomial.*

Proof: We first prove that every vertex $\hat{x}$ of polyhedron $P = \{x | Ax = b, l \leq x \leq r\}$ is an integer vector. By the definition of vertex, there are n linearly independent constraints[3] that are active[4] at $\hat{x}$ (certainly all equality constraints $Ax = b$ must be active). Let $A'x = b'$ denote the active constraints. We know that vertex $\hat{x}$ satisfies the constraints $\hat{x} = (A')^{-1}b'$ and $l \leq \hat{x} \leq r$ and that $n \times n$ matrix A' has the consecutive ones property. Since any nonsingular matrix with the consecutive ones property has determinants $+1$ or -1 [26], the matrix inverse A^{-1} is an integer matrix. Therefore, vertex $\hat{x}$ is an integer vector.

Next we show that the optimal objective value of each LP $\mathcal{P}^{min}[k]$ (actually $\mathcal{P}^{min}[k]$ is IP but we consider its LP relaxation here) is an integer. We know that if a LP has nonempty feasible set and finite optimal objective, then there exists a vertex in its feasible set which is its optimal solution. Let x^{min} denote this vertex of $\mathcal{P}^{min}[k]$. The vertex x^{min} is an integer vector since every vertex of polyhedron P is an integer vector. Therefore, the optimal objective of $\mathcal{P}^{min}[k]$ must be an integer. The same conclusion also holds for LP $\mathcal{P}^{max}[k]$. Hence, inference interval I_k can be computed by any polynomial-time algorithm for LPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ (note that $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ are actually IPs). □

Regarding to the problem of auditing real variables, we have the following result.

Proposition 3.3. *Auditing real variables is polynomial.*

Proof: The proof is straightforward due to the fact that LP problems can be solved in polynomial time. □

[3] A set of linear equality or inequality constraints is said to be *linearly independent* if the corresponding vectors a_i are linearly independent.

[4] Given a linear equality or inequality constraint $a_i^T x \geq b_i$ or $a_i^T x \leq b_i$ or $a_i^T x = b_i$, the constraint is said to be *active* at $\hat{x}$ if $\hat{x}$ satisfies $a_i^T \hat{x} = b_i$.

One of the most efficient algorithms for solving those LPs is Karmarkar's algorithm[19], whose time complexity is $O(mn^{4.5})$ (strictly speaking, the complexity is $O(n^{3.5}L)$ where L is the number of bits required to store each LP $\mathcal{P}^{min}[k]$ or $\mathcal{P}^{max}[k]$ in computer). Therefore, if we solve those LPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ by the Karmarkar's algorithm, the complexity of auditing real variables (or equivalently, auditing integer variables in the case of one-dimensional queries) is $O(mn^{5.5})$ (note that we need to solve $2n$ LPs), which is worse than the complexity result $O(mn^2)$ of audit expert[8] in the case of auditing exact inference.

Because the feasible set $P = \{x|Ax = b, l \leq x \leq r\}$ of LPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ is a convex set, for any $x_k \in I_k$ there exists a solution $x' \in \{x|Ax = b, l \leq x \leq r\}$ such that $x'_k = x_k$. This means that a database attacker cannot obtain any strict subset of I_k without extra knowledge.

4 Auditing Systems with Different Auditing Policies

In the previous section, we studied the problem of auditing interval-based inference. Specifically, we proved the following results: (i) auditing integer variables is NP-hard; (ii) auditing integer variables in the case of one-dimensional queries or auditing real variables is polynomial (w.r.t. the number of queries and the number of database records). In this section, we study various auditing systems that enforce different auditing policies. An *auditing policy* determines under which condition a set of queries are "safe" or "unsafe", and an *auditing system* checks user queries and enforces a particular auditing policy. We analyze the trade-offs among inference security, database usability, and auditing complexity in various auditing systems.

4.1 Auditing System State

We first define *auditing system state* for the purpose of describing auditing systems. The auditing system state consists of three components: *inference security*, *database usability*, and *auditing complexity*, each of which is defined as a lattice.

Definition 4.1. (Inference Security) *Inference security is defined as a lattice* $\langle S, \subseteq_s \rangle$*, where* S *is the power set of* n *variables* $x_1, \ldots, x_n$*, and* $\subseteq_s$ *is subset relationship* $\subseteq$ *on* S*.*

For each pair $s_1, s_2 \in S$, the least upper bound is $s_1 \cup s_2$ and the greatest lower bound is $s_1 \cap s_2$. Each element $s \in S$ describes an auditing system which *guarantees no interval-based inference (in any set of queries) for any variable in* s. Given $s_1, s_2 \in S$, if $s_1 \subseteq_s s_2$, then the auditing system described by s_2 is safer than the one described by s_1. Particularly, $s = \{x_1, \ldots, x_n\}$ indicates no interval-based inference, and $s = \emptyset$ gives no guarantee on interval-based inference.

Definition 4.2. (Database Usability) *Database usability is defined as a lattice* $\langle U, \subseteq_u \rangle$*, where* U *is the power set of the power set of all the queries, and* $\subseteq_u$ *is subset relationship* $\subseteq$ *on* U*.*

Each element $u \in U$ describes an auditing system in which the sets of answerable queries are given by u. For example, if $u = \{\{\{x_2, x_3\}\}, \{\{x_1, x_2\}, \{x_1, x_3\}\}\}$, it means that the auditing system can answer the query $\{x_2, x_3\}$, or any subset of the queries $\{\{x_1, x_2\}, \{x_1, x_3\}\}$. Given $u_1, u_2 \in U$, if $u_1 \subseteq_u u_2$, then the auditing system described by u_2 is more accessible than the one described by u_1. We say that an auditing system described by $u \in U$ provides (i) *appropriate restriction* (on database usability) if $u = u_{approp}$ and u_{approp} contains all the sets of queries except exactly the sets of queries which lead to interval-based inferences; (ii) *strong restriction* if $u = u_{strong}$ and $u_{strong} \subset u_{approp}$; (iii) *weak restriction* if $u = u_{weak}$ and $u_{weak} \supset u_{approp}$; and (iv) *inappropriate restriction* if the system provides neither appropriate nor strong nor weak restriction. An auditing system that provides appropriate or strong restriction is free of interval-based inference, while the one that provides weak or inappropriate restriction gives no such guarantee.

Definition 4.3. (Auditing Complexity) *Given an arbitrary set of* m *queries on* n *variables, auditing complexity is defined as a lattice* $\langle C, \subseteq_c \rangle$*, where* $C = \{NP\text{-}hard^5\} \cup \{m^i n^j : i, j \geq 0\}$*, and* $\subseteq_c$ *is a binary relationship on* C*: (i) for each* $c = m^i n^j$ *in* C *we have* $NP\text{-}hard \subseteq_c c$*; (ii) for each pair* $c_1 = m^{i_1} n^{j_1}$ *and* $c_2 = m^{i_2} n^{j_2}$ *in* C *we have* $c_1 \subseteq_c c_2$ *iff* $i_1 \geq i_2$ *and* $j_1 \geq j_2$*.*

Auditing complexity $\langle C, \subseteq_c \rangle$ is a lattice with infinite number of elements. The lattice is used to classify sets of auditing problems that fall in the appropriate complexity class. For each pair $c_1 = m^{i_1} n^{j_1}$ and $c_2 = m^{i_2} n^{j_2}$ in C, the least upper bound is $m^{min(i_1,i_2)} n^{min(j_1,j_2)}$, and the greatest lower bound is $m^{max(i_1,i_2)} n^{max(j_1,j_2)}$. If $c_1 = m^i n^j$ and $c_2 = NP\text{-}hard$, then the least upper bound is $m^i n^j$, and the greatest lower bound is $NP\text{-}hard$. If $c_1 = c_2 = NP\text{-}hard$, then both the least upper bound and the greatest lower bound are $NP\text{-}hard$. If $c = m^i n^j$, it means that auditing interval based inference will take $\Theta(m^i n^j)$ time (we may also interpret it in terms of space). Similarly, $c = NP\text{-}hard$ indicates that the auditing problem is NP-hard. We use $O(m^i n^j)$ to denote those c satisfying $m^i n^j \subseteq_c c$, and $\Omega(m^i n^j)$ those c satisfying $c \subseteq_c m^i n^j$.

Definition 4.4. (Auditing System State) *The space of auditing system states* $\langle \mathcal{O}, \subseteq_o \rangle$ *is defined as the product lattice of the three underlying lattices: inference security* $\langle S, \subseteq_s \rangle$*, database usability* $\langle U, \subseteq_u \rangle$*, and auditing complexity* $\langle C, \subseteq_c \rangle$*. The* auditing system state *of an auditing system is an element* $o \in \mathcal{O}$*, denoted as a triple* $o = \langle s, u, c \rangle$ *where* $s \in S$*,* $u \in U$*, and* $c \in C$*.*

[5] Note that we assume $NP \neq P$ in definition 4.3. If $NP = P$, the node of "$NP\text{-}hard$" can be simply removed from the lattice $\langle C, \subseteq_c \rangle$ without affecting its validity.

Auditing policy	Auditing result "safe"	Auditing result "unsafe"
Exact auditing "safe"$\Leftrightarrow$ $s = \{x_1, \ldots, x_n\}$	No inference	Presence of inference
Optimistic auditing "safe"$\Leftarrow$ $s = \{x_1, \ldots, x_n\}$	Possible presence of inference (false negatives)	Presence of inference
Pessimistic auditing "safe"$\Rightarrow$ $s = \{x_1, \ldots, x_n\}$	No inference	Possible no inference (false positives or alarms)

Fig. 1. Definition of different auditing policies

4.2 Auditing Policies

Now we define different auditing policies for the purpose of classifying various auditing systems. Given a set of queries, an auditing policy stipulates that under which condition the auditing result "safe" or "unsafe" is given. The auditing result "safe" means that the given set of queries is answerable and that response "unsafe" means unanswerable. Either "safe" or "unsafe" must be given as an auditing result. We define three types of auditing policies given a set of queries:

Exact Auditing The auditing result "safe" is given *iff* no interval-based inference exists; that is, "safe"$\Leftrightarrow$ $s = \{x_1, \ldots, x_n\}$. We denote the auditing system state for exact auditing (policy) as $o_{exact} = \langle s_{exact}, u_{exact}, c_{exact} \rangle$.

Optimistic Auditing The auditing result "safe" is given *if* no interval-based inference exists; that is, "safe"$\Leftarrow$ $s = \{x_1, \ldots, x_n\}$. We denote the auditing system state for optimistic auditing (policy) as $o_{opti} = \langle s_{opti}, u_{opti}, c_{opti} \rangle$.

Pessimistic Auditing The auditing result "safe" is given *only if* there is no interval-based inference; that is, "safe"$\Rightarrow$ $s = \{x_1, \ldots, x_n\}$. We denote the auditing system state for pessimistic auditing (policy) as $o_{pessi} = \langle s_{pessi}, u_{pessi}, c_{pessi} \rangle$.

The definition of the auditing policies is summarized in figure 1. We classify various auditing systems into three categories according to the auditing policies they enforce: *exact auditing system, optimistic auditing system* and *pessimistic auditing system*. The characteristics of these auditing systems can be described in terms of *soundness* and *completeness*. An auditing system is *sound* means that if the auditing system says that there is an interval-based inference, then an inference exists. An auditing system is *complete* means that if there exists an interval-based inference, then the system will say so. In terms of soundness and completeness, exact auditing system is both sound and complete; optimistic auditing system is sound but may not be complete; and pessimistic auditing system is complete but not necessarily sound. In other words, optimistic auditing system may produce false negatives (i.e., some inference may not be detected) but no false positives (false alarms); on the contrary, the pessimistic auditing system may produce false positives but no false negatives. Exact auditing system produces neither false negatives nor false positives.

4.3 Auditing Systems

Now we study the auditing systems that enforce auditing policies. We focus on the trade-offs among the three components of auditing system state $o = \langle s, u, c \rangle$ that can be achieved by various auditing systems. To enforce the optimistic auditing policy, certain interval-based inference can be tolerated if only it is "hard" for database attackers to obtain it. To enforce pessimistic auditing policy, even some restrictions on "safe" queries can be allowed if its impact on database usability is negligible. In both cases, auditing complexity is balanced against inference security and/or database usability.

Exact Auditing System Exact auditing system is defined as an auditing system that enforces exact auditing policy. Exact auditing system can be described as follows: (i) it guarantees no interval-based inference on inference security ($s_{exact} = \{x_1, \ldots, x_n\}$); (ii) it imposes an appropriate restriction on database usability ($u_{exact} = u_{approp}$); (iii) its auditing complexity is NP-hard for interval variables ($c_{exact} = NP\text{-}hard$) and polynomial for real variables or for integer variables in the case of one-dimensional range queries ($c_{exact} \supseteq_c mn^{5.5}$).

Exact auditing system requires that the length of each inference interval $|I_k|$ $(1 \leq k \leq n)$ be exactly computed; that is, the *exact* optimal solutions to IPs/LPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ must be obtained. Due to the fast development of computing techniques, these IP problems (for auditing integer variables) can be commonly solved in hundreds or thousands of variables and constraints, so do the LP problems (for auditing real variables) with tens or hundreds of thousands of variables[18]. As a result, exact auditing system can be implemented for small or medium size auditing problems. However, database attackers benefit more from this than auditing systems do. To compromise sensitive information, a database attacker can use dedicate computing resources for a single set of queries. In comparison, to detect all possible inferences, an auditing system must perform the computation for all potential attackers (database users). It is thus impractical in a large auditing system with many users, especially in an on-line environment.

Optimistic Auditing System Optimistic auditing system is defined as an auditing system that enforces optimistic auditing policy. Optimistic auditing system is based on the belief that if it is computationally infeasible to audit an interval-based inference in a set of queries, it is also infeasible for a database attacker to compromise the sensitive information in the same set of queries even though the information is actually revealed in principle. Compared with exact auditing system, we have (i) $s_{opti} \subseteq_s s_{exact}$, (ii) $u_{opti} \not\subseteq_u u_{exact}$[6], and usually we require that $c_{opti} \supseteq_c c_{exact}$. In other words, optimistic auditing system seeks to improve database usability and/or auditing complexity at the expense of inference security. We illustrate this through the following case studies.

[6] As to database usability, optimistic auditing resorts to weak restriction or inappropriate restriction. Note that inappropriate restriction can always be transferred to weak restriction by updating database usability $u = u_{opti} \cup u_{approp}$ while at the same time keeping inference security $s = s_{opti}$ unchanged.

Case study 4.1. (Optimistic auditing: auditing by LP relaxation) Consider auditing integer variables by solving the IPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$. It is NP-hard to solve those IPs under exact auditing policy. However, optimistic auditing policy allows us to compute each "inference interval" by solving LP relaxations of those IPs rather than the IPs themselves (the length of the "inference interval" computed this way is greater than its true value – leading to false negatives). Consequently, auditing complexity is improved from NP-hard to polynomial. □

Case study 4.2. (Optimistic auditing: auditing with relaxed bounds) Consider using relaxed bounds $l' \leq x \leq r'$ in stead of $l \leq x \leq r$ in solving the MPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$, where $l' \leq l$ and $r' \geq r$. By doing so, the length of the "inference interval" computed is greater than its true value – leading to false negatives, and database usability is improved compared with exact auditing system. □

Optimistic auditing system might be impractical in some applications since it is usually difficult to decide when it is infeasible for a database attacker to compromise the sensitive information that is actually released. Therefore, the implementation of optimistic auditing could be "dangerous" due to the presence of false negatives. This inspires the study of pessimistic auditing system.

Pessimistic Auditing System Pessimistic auditing system is defined as an auditing system that enforces pessimistic auditing policy. Compared with exact auditing system, we have (i) $s_{pessi} = s_{exact}$, (ii) $u_{pessi} \subseteq_u u_{exact}$, and usually we require that $c_{pessi} \supseteq_c c_{exact}$. In other words, pessimistic auditing system ensures no inference on inference security. It seeks to achieve a better result on auditing complexity by imposing strong restriction on database usability. We examine pessimistic auditing system through the following case studies.

Case study 4.3. (Pessimistic auditing: auditing by trace) Given a set of sum queries $a_k x = b_k$, $1 \leq k \leq n$ (or $Ax = b$) w.r.t. $x^* = (x_1^* \dots x_n^*)^T$, we define the *trace* $\tau(x_i^*)$ of value x_i^* as the set $\{k | a_{ki} = 1, 1 \leq k \leq n\}$. Auditing by trace is based on the following observation: If for each value x_i^*, there exists a value x_j^* such that $|x_i^* - x_j^*| \geq \epsilon$ and $\tau(x_i^*) = \tau(x_j^*)$, then no interval-based inference exists, where ϵ is the tolerance level. The observation is true because a database attacker can never discriminate between variable x_i and x_j.

To implement auditing by trace, we maintain a graph whose vertex set is the collection of the variables. We join variables x_i and x_j by an edge if $|x_i^* - x_j^*| \geq \epsilon$ and $\tau(x_i^*) = \tau(x_j^*)$. Before any set of queries has been presented, we have $\tau(x_i^*) = \tau(x_j^*) = \emptyset$ for any two different variables x_i and x_j. Define the k-th *query set* as $Q_k = \{i | a_{ki} = 1, 1 \leq i \leq n\}$, $k = 1, \dots, n$. With each query set Q_k, we delete the edge (x_i, x_j) iff $|Q_k \cap \{i, j\}| = 1$. As long as the graph has no isolated nodes, no inference exists. It is easy to know that the complexity of such implementation is $O(mn^2)$, and that the complexity of checking whether a new query can be answered is $O(n^2)$ – similar to the complexity result of audit expert in the case of auditing exact inference. □

Case study 4.4. (Pessimistic auditing: auditing by approximation) Auditing by approximation seeks to obtain each inference interval with its length smaller than its true value. To achieve this, we solve the MPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$ using approximate algorithms rather than exact algorithms, as long as these approximate algorithms stop at feasible solutions of the original MPs. In mathematical programming literature, many approximate algorithms provide feasible solutions and have better time complexity compared with corresponding exact algorithms. □

Case study 4.5. (Pessimistic auditing: auditing by feasible solution) Auditing by feasible solution seeks the same objective as auditing by approximation; however, it directly uses the exact algorithms of the MPs and stop in the middle as long as feasible solutions are obtained. If the exact algorithms search the optimal solutions within feasible sets (actually most exact algorithms do so), we can always achieve polynomial time complexity since we can stop the algorithms at any time (it should be balanced against database usability requirement). □

Case study 4.6. (Pessimistic auditing: auditing with enhanced bounds) Consider using enhanced bounds $l' \leq x \leq r'$ in stead of $l \leq x \leq r$ in solving the MPs $\mathcal{P}^{min}[k]$ and $\mathcal{P}^{max}[k]$ $(1 \leq k \leq n)$, where $l' \geq l$ and $r' \leq r$. By doing so, the length of the "inference interval" computed is less than its true value – leading to false alarms, and database usability is decreased compared with exact auditing system. □

Integrated Auditing Systems In practice, there might exist many different ways to implement pessimistic auditing (or optimistic auditing). Each way can be considered as an independent auditing system, called *pessimistic (or optimistic) auditing unit*. A number of such auditing units can be integrated into a generic framework, called *integrated pessimistic (or optimistic) auditing system*. In the following discussion, we focus on integrated pessimistic auditing system; while the extension to integrated optimistic auditing system is trivial.

Integrated pessimistic auditing system is shown in figure 2. For convenience, denote the auditing system state of unit i using $o_i = \langle s_i, u_i, c_i \rangle$ and the integrated system using $o_{sys} = \langle s_{sys}, u_{sys}, c_{sys} \rangle$. In figure 2, when a unit receives its input, it first determines whether it is applicable for the input. If not, it hands over the input to the next unit (if available); otherwise, it audits the input. If the auditing result is "safe", the whole auditing process is terminated with result "safe" (due to that pessimistic auditing produces no false negatives); otherwise, the input is handed over to the next unit (if available). When the last unit outputs "unsafe", the whole auditing process is terminated with the result "unsafe" (though it is still possible the input is "safe" – false alarm).

Briefly speaking, integrated pessimistic auditing system responses "safe" iff one of its unit answers "safe". It is clear that the integrated system enforces pessimistic auditing policy; we have (i) $s_{sys} = \cap_i s_i$ $(s_{sys} = s_{exact} = \{x_1, \ldots, x_n\})$, (ii) $u_{sys} = \cup_i u_i$ $(u_{sys} \subseteq_u u_{exact})$, and (iii) c_{sys} is no worse than the greatest lower bound of those c_i in the auditing complexity lattice (see definition 4.3). Note

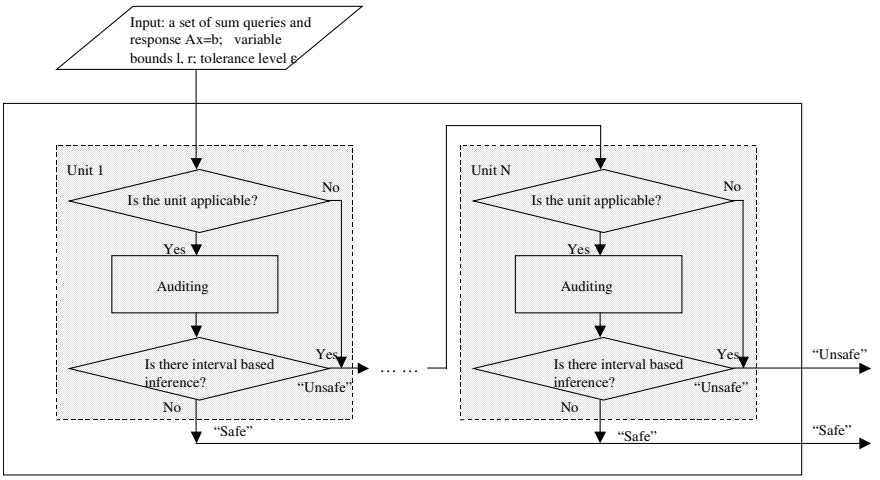

Fig. 2. Integrated pessimistic auditing system

that sorting the units in certain orders may help improve auditing complexity of the system; however, we do not address this issue in this paper.

For integrated optimistic auditing system, our purpose is to increase the auditing security at the expense of database usability and/or the auditing complexity. The system responses "safe" iff all (or many) its units response "safe". In this case, we have (i) $s_{sys} = \cup_i s_i$, (ii) $u_{sys} = \cap_i u_i$, and (iii) c_{sys} is no worse than the greatest lower bound of those c_i in the auditing complexity lattice.

Finally, the auditing system state for various types of auditing discussed in this paper is summarized in figure 3.

5 Conclusion and Future Directions

In this paper, we pointed out the significance to investigate interval based inference and formulated the auditing problem formally. Our study showed that: (1) The auditing problem invalidates existing methods (e.g., audit expert) designed for auditing exact inference. (2) Auditing interval-based inference is possible; however, it involves complicated computation of mathematical programming. The complexity of auditing integer variables is NP-hard, while auditing real variables is polynomial. (3) Under different auditing policies, various auditing systems can be classified into three categories: exact auditing, optimistic auditing, and pessimistic auditing. Trade-offs can be achieved by different auditing systems among inference security, database usability and auditing complexity.

Auditing policy	auditing system state $o = \langle s, u, c \rangle$		
	Inference security s	Database usability u	Auditing complexity c
Exact auditing	No inference	Appropriate restriction	1. NP-hard: auditing integer variables 2. Polynomial ($O(mn^{5.5})$): auditing real variables
Optimistic auditing	Possible presence of some inferences	Weak restriction or inappropriate restriction	Polynomial ($O(mn^{5.5})$): auditing by LP relaxation for auditing integer variables
Pessimistic auditing	No inference	Strong restriction	1. $O(mn^2)$: auditing by trace 2. Possible to achieve better results by other methods e.g. auditing by approx. and auditing by feasible solution

Fig. 3. Auditing system state for different types of auditing

For practical auditing systems, we believe that the optimistic auditing and pessimistic auditing are promising. Also, observe that many real world applications, such as OLAP, restrict user queries to specific meaningful forms; we may take advantage of such restrictions to build efficient audit systems for such applications. It would also be interesting to reconsider other inference control methods such as microaggregation (see, e.g., [16,22]) and random data perturbation (RDP) techniques for interval-based inference [21]. Although such techniques would provide users imprecise (perturbed) query responses rather than exact responses, the upper side is that no restrictions are imposed on user queries and that the complexity for perturbing data is low.

6 Acknowledgments

We thank the anonymous referees for their valuable comments.

References

1. N.R. Adam and J.C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989. 553
2. L.L. Beck. A security mechanism for statistical databases. *ACM Trans. on Database Systems*, 5(3):316–338, 1980. 554
3. A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Trans. Knowledge and Data Engineering*, 12(6):900–919, 2000. 557
4. A. Brodsky, C. Farkas, D. Wijesekera, and X.S. Wang. Constraints, inference channels and secure databases. In *the 6th International Conference on Principles and Practice of Constraint Programming*, pages 98–113, 2000. 557
5. F.Y. Chin, P. Kossowski, and S.C. Loh. Efficient inference control for range sum queries. *Theoretical Computer Science*, 32:77–86, 1984. 554

6. F.Y. Chin and G. Özsoyoglu. Security in partitioned dynamic statistical databases. In *Proc. of IEEE COMPSAC*, pages 594–601, 1979. 554
7. F.Y. Chin and G. Özsoyoglu. Statistical database design. *ACM Trans. on Database Systems*, 6(1):113–139, 1981. 554
8. F.Y. Chin and G. Özsoyoglu. Auditing and inference control in statistical databases. *IEEE Trans. on Software Engineering*, 8(6):574–582, 1982. 554, 559
9. L.H. Cox. Suppression methodology and statistical disclosure control. *Journal of American Statistic Association*, 75(370):377–385, 1980. 554
10. D.E. Denning. Are statistical data bases secure? In *AFIPS conference proceedings*, volume 47, pages 199–204, 1978. 553
11. D.E. Denning. Secure statistical databases with random sample queries. *ACM Trans. on Database Systems*, 5(3):291–315, 1980. 554
12. D.E. Denning and P.J. Denning. Data security. *ACM computing surveys*, 11(3):227–249, 1979. 553
13. D.E. Denning, P.J. Denning, and M.D. Schwartz. The tracker: A threat to statistical database security. *ACM Trans. on Database Systems*, 4(1):76–96, 1979. 554
14. D.E. Denning and J. Schlörer. Inference controls for statistical databases. *IEEE Computer*, 16(7):69–82, 1983. 553
15. D. Dobkin, A.K. Jones, and R.J. Lipton. Secure databases: protection against user influence. *ACM Trans. on Database Systems*, 4(1):97–106, 1979. 554
16. J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Trans. Knowledge and Data Engineering (to appear)*. 566
17. L.P. Fellegi. On the qestion of statistical confidentiality. *Journal of American Statistic Association*, 67(337):7–18, 1972. 554
18. R. Fourer. Linear programming frequently asked questions. Optimization Technology Center of Northwestern University and Argonne National Laboratory, 2001. http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html. 562
19. J.P. Ignizio and T.M. Cavalier. *Linear Programming*. Prentice Hall, 1994. 558, 559
20. J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proc. of the 9th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 86–91, 2000. 554, 555, 557
21. Y. Li, L. Wang, and S. Jajodia. Preventing interval-based inference by random data perturbation. In *Workshop on Privacy Enhancing Technologies (to appear)*. 566
22. Y. Li, S. Zhu, L. Wang, and S. Jajodia. A privacy-enhanced microaggregation method. In *Proc. of the 2nd International Symposium on Foundations of Information and Knowledge Systems*, pages 148–159, 2002. 566
23. F.M. Malvestuto and M. Moscarini. Computational issues connected with the protection of sensetive statistics by auditing sum-queries. In *Proc. of IEEE Scientific and Statistical Database Management*, pages 134–144, 1998. 554
24. M.A. Palley. Security of statistical databases compromise through attribute correlational modeling. In *Proc. of IEEE Conference on Data Engineering*, pages 67–74, 1986. 553
25. J. Schlörer. Security of statistical databases: multidimensional transformation. *ACM Trans. on Database Systems*, 6(1):95–112, 1981. 554
26. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986. 558
27. J.F. Traub, Y. Yemini, and H. Woźnaikowski. The statistical security of a statistical database. *ACM Trans. on Database Systems*, 9(4):672–679, 1984. 554

28. S.L. Warner. A survey technique for eliminating evasive answer bias. *Journal of American Statistic Association*, 60(309):63–69, 1965. 554