Auditing Boolean Attributes

Jon Kleinberg * Christos Papadimitriou † Prabhakar Raghavan ‡

Abstract

We study the problem of auditing databases which support statistical sum queries to protect the security of sensitive information; we focus on the special case in which the sensitive information is Boolean. Principles and techniques developed for the security of statistical databases in the case of continuous attributes do not apply here. We prove certain strong complexity results suggesting that there is no general efficient solution for the auditing problem in this case. We propose two efficient algorithms: The first is applicable when the sum queries are one-dimensional range queries (we prove that the problem is NP-hard even in the two-dimensional case). The second is an approximate algorithm that maintains security, although it may be too restrictive. Finally, we consider a "dual" variant, with continuous data but an aggregate function that is combinatorial in nature. Specifically, we provide algorithms for two natural definitions of the auditing condition when the aggregate function is MAX.

1 Introduction

The explosive increase in access to sensitive information has renewed concerns on the compromise of individual privacy through queries about groups of people. Consider a relation with attributes `(name, age, salary)` supporting statistical queries of the form "give me the sum of salaries of all individuals whose age x satisfies condition $C(x)$," where C is an arbitrary predicate on the domain of `age`, such as $30 \leq x \leq 40$. Assume further that the projection `(name, age)` is publicly available, but the attribute `salary` is confidential. What measures suffice to protect the confidentiality of the salary information?

This is the classical *statistical database security* problem, studied extensively since the 1970's; see [1] for a survey. The main approaches to this problem involve perturbing the data so as to maintain their statistical characteristics but prevent their compromise [13, 11, 16], to perturb the responses for the same purpose, [2, 8], to restrict the size or overlap of the statistical queries [10, 9], or, finally (and closer to our concerns here), to *audit* the

*Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. Supported in part by a David and Lucile Packard Foundation Fellowship, an Alfred P. Sloan Research Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR-9701399.

†Computer Science Division, Soda Hall, UC Berkeley, CA 94720. christos@cs.berkeley.edu. Research supported by an NSF grant.

‡IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120. pragh@almaden.ibm.com

statistical queries in order to determine when enough information has been given out so that compromise becomes possible [3, 4, 5, 12].

Most of the work in this area assumes that the confidential data are real-valued and essentially unbounded. In certain important applications, however, data may attain discrete values, or have maximum or minimum values that are fixed *a priori* and frequently attainable. In these cases, traditional methods for maintaining security are inadequate. For example, if a statistical query only samples minimum values (e.g., if it so happens that all individuals whose age satisfies $C(x)$ are paid the minimum legal salary), then individual values are obviously compromised. Discreteness of values has even more subtle effects. *Boolean* attributes, of course, combine the problems of discrete and bounded variables; for example, consider a relation `(name, age, hivpos)`, where the last attribute has values restricted to 0 or 1. Sum queries are again allowed.

The mathematical roots of the problem lie in the fact that linear Diophantine equations are more restricting —and have greater complexity— than linear equations. For example, the system

$$x + y = 1 \tag{1}$$
$$y + z = 1 \tag{2}$$
$$x + z = 1 \tag{3}$$

has a unique solution $x = y = z = \frac{1}{2}$, but no 0-1 (or integer) solution. Consequently, the system

$$x + y + w = 1 \tag{4}$$
$$y + z = 1 \tag{5}$$
$$x + z = 1 \tag{6}$$

is secure if the variables are real (because in this case it has a one-dimensional continuum of solutions), but not if they are Boolean, because in the latter case the values of all variables are determined. Evidently, Boolean attributes make the auditing problem much more tricky. This added complexity of integer variables had been identified in the literature [14], albeit with no analytical exploration of the issue.

The present work. *In this paper we explore the novel mathematical and algorithmic problems arising when one tries to audit statistical queries on Boolean attributes.* (We also study a "dual" situation, in which the data is continuous but the query discrete; see below.) We consider a setting in which we have a collection of (secret) Boolean variables, and the results of some *statistical queries* to this set. Each such query simply specifies a subset S of the variables; the value returned in response to this query is the sum of the values of all variables in S.

We want to decide whether the value of any of the Boolean variables is *determined* by the results of these queries. In other words, the collection of responses to the queries defines a system of equations as above, and we want to know whether there is any variable x_i so that x_i has the same value in *every* solution to this system of equations. One can view the value of this variable as having been compromised by the results of the queries. We call this the *auditing problem.* A natural variant of this problem is to place a stronger auditing requirement on a set of queries: for some number $k \geq 1$, there is no set T consisting of at most k of the variables for which the sum of the values has been determined. Our basic auditing problem is then simply the case $k = 1$.

Remark: In more traditional work on auditing, there is a more subtle and generic variant of the auditing problem, in which one asks not whether the given set of queries compromises security for the *present* values of the variables, but *for any* values of the variables. In the case of Boolean-valued variables — or, indeed, variables over any bounded domain — this kind of generic auditing is impossible: For any set S of Boolean variables, there is a set of values in which all these variables are 1. Thus, if this query S is asked to that database, an attacker would compromise all the values in S. (This is not unlike the minimum wage example above). Therefore, an auditing system for Boolean attributes should, with small probability, refuse to answer any query submitted.

Our first result on the auditing problem for Boolean values is that it is *coNP-complete.*[1] It follows from our proof that it is even NP-hard to distinguish between a case in which no variable is determined, and a case in which all variables are. The generalized problem in which no sum of up to k variables is to be compromised is also NP-hard, and likewise for the variant in which we ask whether a *specific* variable x_p is compromised.

It is natural to ask whether these hardness results hold only for "pathological" sets of queries. If we consider a collection of individuals specified by tuples of attribute values — all of the attributes public with the exception of (one or more) secret Boolean attributes — then selecting subsets of individuals via range queries on their public attributes is a well-defined and natural class of "reasonable" sets of queries. In other words, we are interested in instances of the auditing problem in which the variables correspond to points in $\mathbf{R}^d$, and the query sets are the intersections of these points with d-dimensional boxes. We will refer to such instances as *the special case of d-dimensional range queries.*

Our next result is a simple, polynomial-time combinatorial algorithm for auditing one-dimensional range queries, using techniques from combinatorial optimization [15]. We also show the auditing problem is coNP-complete even for *two-dimensional* range queries (and hence for any $d \geq 2$).

For the general Boolean case, we also describe a simple and efficient method that *approximates* the auditing problem, in that it successfully preserves the security of individual values —even though it may refuse to answer queries that could be answered without compromising security (by the remark above, this last point is inherent to the problem). Our technique is akin to the *partitioning* approach to data security [6].

[1] The complexity of the problem has some intriguing consequences for the issue of auditing in general. Specifically, it is computationally infeasible for an auditor to decide whether the value of a variable will be revealed by the answers to the a set of queries; but symmetrically, it may be infeasible for an attacker to actually compute the value of such a variable that has been, in principle, revealed. This raises novel possibilities, such as the auditing policies allowing sets of queries that either do not determine variables, or constitute "hard instances" of the auditing problem; we leave these intriguing issues as directions for further work.

MAX Queries. We also consider a "dual" variant, in which the data is continuous, but the aggregate function is combinatorial in nature. In this variant the sensitive data is real-valued, and the aggregate function is MAX rather than summation. That is, we are given a set of real-valued variables, and each query returns the maximum value over a designated subset of the variables. Again, we ask: Is the value of any variable *determined* by the responses to these queries? We provide a simple and efficiently implementable characterization of the auditing condition in this case.

Recall the *generic* auditing condition discussed above — given a collection of query sets, does there *exist* a set of values for which some variable would be determined? In contrast to the Boolean case, this question becomes non-trivial in the case of MAX queries over real-valued data, and raises issues of a technically distinct flavor from the main auditing problem we study. We provide a characterization of query sets that are secure, in this generic sense, when the aggregate function is MAX.

2 Complexity

Define the BOOLEAN AUDITING PROBLEM to be the following: Given n 0-1 variables $\{x_1, \ldots, x_n\}$, a family of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of $\{1, \ldots, n\}$, and m integers $b_1, \ldots, b_m$, is there an $i \leq n$ such that in all 0-1 solutions of the system of equations $\sum_{i \in S_j} = b_j, j = 1, \ldots, m$, the variable x_i has the same value?

Theorem 2.1 *The* BOOLEAN AUDITING PROBLEM *is coNP-complete.*

Proof: It is well-known that determining whether a system of linear equations has a 0-1 solution is NP-hard even if all coefficients are 1, the right-hand side of each equation is 1, and there are at most three variables per equation. We start from this problem.

Given such a system of equations, we first replace each variable x by the expression $x_1 + x_2 + x_3 - 1$, and add the equations $x_1 + x_2 + x_3 + x_4 + x_5 = 2$, $x_1 + x'_1 = 1$, $x_2 + x'_2 = 1$, $x_3 + x'_3 = 1$, $x'_1 + x'_2 + x'_3 + x_6 + x_7 = 2$, where x_i, x'_i are new variables. The meaning of these equations is that $x_1 + x_2 + x_3$ is either 1 or 2, and thus $x_1 + x_2 + x_3 - 1$ is either 0 or 1, and therefore the latter expression can safely replace the Boolean variable x, *but the new variables are never determined* as there are always several ways to achieve the same value. Once these replacements have been made, the right-hand side of each equation is an integer no larger than 4. We introduce now 4 new variables a, b, c, d bound to be equal by the equations $a + a' = 1$, $a' + b = 1$, $b + b' = 1$, $b' + c = 1$, $c + c' = 1$, $c' + d = 1$. We finally add to the left-hand side of each equation (except for these last six involving a, b, c, d) a number of the a, b, c, d variables equal to the right-hand side of the equation. This completes the construction.

Notice that now that the system always has a 0-1 solution, one obtained by setting $a = b = c = d = 1$ and all other variables 0. If this is the only solution, then the system is insecure, because the values of all variables are determined. It is easy to see that the only way for another solution to exist is for the original system to have a solution; in that case, it is not hard to prove that no variable is determined.

Notice, incidentally, that this proof also establishes that it is coNP-hard to distinguish between the case in which *all* variables are determined and the case in which none is; as a consequence, telling whether a specific variable is determined is also coNP-complete. ■

Let us call a family of finite sets *d-dimensional* if the elements can be identified with points in $\mathbf{R}^d$ so that the minimum bounding box of each set in the family contains no other element besides those in the set. An instance of the BOOLEAN AUDITING PROBLEM is d-dimensional if the family of sets in it is. For example, any instance resulting from the `(name, age, hivpos)` example described in the introduction, with conditions on the age of the form $C(x) = \ell \leq x \leq u$, is one-dimensional. The following result suggests that the auditing problem remains intractable even in its 2-dimensional special case.

Theorem 2.2 *The* BOOLEAN AUDITING PROBLEM *is coNP-complete even if the system is restricted to be 2-dimensional.*

It is clear that d-dimensional queries, with $d > 2$, can be no easier.

Proof: We reduce the general case to the two-dimensional one as follows: Let $\mathcal{S}$ be a family of sets defining an instance of the BOOLEAN AUDITING PROBLEM. Arrange the sets in $\mathcal{S}$ in some order, and for each consider the occurrences of each variable in it, also in some arbitrary order. Each one of these occurrences will be a separate variable in the new instance, with the i^{th} occurrence of x denoted x_i.

Assign now to these new variables a point in the 2-dimensional plane, by assigning to the k^{th} such variable the point (k, k), $k = 1, \ldots, \sum_{S_i \in \mathcal{S}} |S_i|$. Notice that, this way, the equations of the original system indeed involve a set of points whose minimum bounding rectangle contains no other point.

All we need now is to enforce the additional constraints stating that all new variables corresponding to the same variable in the original problem take the same value. We achieve this as follows: Suppose that (k, k) and (ℓ, ℓ) are two points corresponding to two consecutive occurrences of the same variable, say x_i and x_{i+1}, respectively. We then introduce a new variable y_i, with point (k, ℓ), and equations $x_i + y_i = 1$, $x_{i+1} + y_i = 1$. Obviously, these two equations force x_i and x_{i+1} —and by extension all occurrences of x— to have the same value. Furthermore, the minimum bounding rectangles involving these two equations are the line segments $[(k, k), (k, \ell)]$ and $[(\ell, \ell), (k, \ell)]$, which, indeed contain no other points corresponding to variables besides the two endpoints.

It follows that the resulting system is 2-dimensional, and equivalent, *vis à vis* auditing, to the original one. ■

3 The One-Dimensional Case

We can, however, prove the following:

Theorem 3.1 *The* BOOLEAN AUDITING PROBLEM *for one-dimensional queries can be solved in polynomial time.*

Proof: We have variables $x_1, \dots x_n$ corresponding to points arranged in this order on a line, while the sets in $\mathcal{S}$ correspond to intervals of the same line. Consider the characteristic vector a_i of the set $S_i \in \mathcal{S}$, let A denote the matrix whose rows are equal to the vectors $a_1, \dots, a_m$, and let b denote the vector $(b_1, \dots, b_m)$. Note that A has the *consecutive-ones property*, in that the ones in each row are all consecutive. We let $\mathcal{P}$ denote the polytope

$$\{x : Ax = b, \quad 0 \leq x \leq 1\}.$$

It is well-known [15] that matrices with the consecutive ones property are *totally unimodular*, that is, all their square submatrices have determinants $+1, -1$, or 0. Consequently, each vertex of the polytope $\mathcal{P}$ defined by

$$\{x : Ax = b, \quad 0 \leq x \leq 1\}$$

has 0-1 coordinates. Now, suppose we let $\mathcal{P}_{i,c}$, for $1 \leq i \leq n$ and $c \in \{0, 1\}$, denote the polytope obtained by intersecting $\mathcal{P}$ with the hyperplane $x_i = c$. Each polytope $\mathcal{P}_{i,c}$ also has the property that all its vertices have integer coordinates. Thus, the value of the variable x_i is determined by the results of the queries if and only if exactly one of the two polytopes $\mathcal{P}_{i,0}$ and $\mathcal{P}_{i,1}$ is non-empty; indeed, each such polytope that is non-empty will have at least one vertex, and this vertex will constitute a set of Boolean values consistent with the results of all queries.

Thus our problem reduces to determining integer solutions to the system of equations and inequalities

$$Ax = b, \quad 0 \leq x \leq 1,$$

where A has the consecutive-ones property; and the arguments above show that this can be solved by any polynomial-time algorithm for linear programming (see e.g. [15]).

However, there is a much more direct and efficient combinatorial algorithm to determine the solvability of this system of equations and inequalities in integers. First, we define a directed graph G as follows.

- G has nodes $0, \dots, n$, with arcs $a_i = (i, i+1)$ and $a'_i = (i+1, i)$ for each $i = 0, 1, \dots, n-1$.
- Also, suppose the block of 1's in the j^{th} row of A runs from column to p to column $q > p$; then we add arcs $e_j = (p-1, q)$ and $e'_j = (q, p-1)$.
- We assign cost 1 to each arc a_i, cost 0 to each arc a'_i, cost b_j to arc e_j, and cost $-b_j$ to arc e'_j.

Now we claim that our initial system is solvable in integers if and only if the graph G has no negative-cost cycle; this latter condition can be tested in polynomial time via well-known combinatorial algorithms [7]. First, suppose the system is feasible, and let $(x_1, \dots, x_n)$ be a 0-1-valued vector that satisfies $Ax = b$. We define $s_0 = 0$ and $s_i = \sum_{j=1}^{i} x_j$ for $i = 1, 2, \dots, n$. Now observe that for every arc (u, v) in G, of cost c, we have $s_v - s_u \leq c$. Indeed, $s_i \leq s_{i+1} \leq s_i + 1$; and if $e_j = (p-1, q)$ corresponds to a row of A, then $s_q - s_{p-1} = \sum_{i=p}^{q} x_i = b_j$. The numbers $\{s_i\}$ thus provide a certificate that G has no negative cycle.

Conversely, suppose that G has no negative cycle. Then we can compute a well-defined shortest path length s'_i from node 0 to each node i. Now, for $i = 1, 2, \ldots, n$, define $x_i = s'_i - s'_{i-1}$. We claim that the vector $x = (x_1, \ldots, x_n)$ satisfies the system $Ax = b,\ \ 0 \leq x \leq 1$. First, observe that each x_i is an integer; and the sets of arcs $\{a_i\}$ and $\{a'_i\}$ force $s'_i \leq s'_{i+1} \leq s'_i + 1$ for each i, whence $x_i \in \{0, 1\}$. Second, the existence of the arcs e_j and e'_j imply that $\sum_{i=p}^{q} x_i \leq b_j$ and $-\sum_{i=p}^{q} x_i \leq -b_j$, whence the j^{th} row of A is satisfied. ∎

Example: The following simple example illustrates the algorithm. Suppose the original matrix A had only one row, consisting of n 1's. Thus the vector b consists of a single number. We know the original system $Ax = b,\ 0 \leq x \leq 1$ to be feasible if and only if $0 \leq b \leq n$, and we want the construction above to capture this. The graph we construct in this case is a bi-directed cycle; it consists of n bi-directed edges with cost 1 clockwise and 0 counter-clockwise, and a single bi-directed edge of cost $-b$ clockwise and b counter-clockwise. Indeed, there is now a negative cycle in G if and only if $b < 0$ or $b > n$.

4 Approximate Auditing

We have seen that it can be computationally infeasible to determine the safety of a collection of arbitrary statistical queries put to a database containing secret Boolean variables. Given arbitrary query sets arriving incrementally over time, then, how should we proceed? How long is it safe to continue providing answers?

A promising approach is to consider relaxed versions of our basic safety predicate. Rather than deciding precisely whether any variable's value has been determined, we can compute a conservative approximation to this predicate: For any collection of query sets, we only answer a query when it is safe to do so, but we may refuse to answer a query even when the answer would not in fact compromise safety.

We now describe a particular conservative approximation which can be implemented very efficiently. Given a collection of Boolean variables $x_1, \ldots, x_n$, and a sequence of query sets $S_1, \ldots, S_m$, we define the *trace* $\tau(x_i)$ of x_i to be the set $\{p : x_i \in S_p\} \subseteq \{1, 2, \ldots, m\}$. In terms of traces, one can express a relaxed safety condition as follows.

Theorem 4.1 *Suppose that for every variable x_i, there exists a variable x_j so that $x_i = 1 - x_j$ and $\tau(x_i) = \tau(x_j)$. Then no variable is determined by the responses to the query sets $S_1, \ldots, S_m$.*

To implement the safety test implied by the Theorem, we maintain a bipartite graph G whose vertex set is the collection of variables. We join variables x_i and x_j by an edge if they have opposite values but the same trace. Theorem 4.1 implies that as long as this graph has no isolated nodes, no variable has been determined. Before any query sets have been presented, we have the complete bipartite graph; with each query set S_p, we delete the edge (x_i, x_j) if and only if $|S_p \cap \{x_i, x_j\}| = 1$.

Note the connections between this methodology and the *partitioning* approach to maintaining security [6]. Under the method we discuss here, one is essentially monitoring an adaptive partition on the variables that is successively refined by each query response; as long as the atomic sub-populations in this partition maintain a particular property, one can continue responding to queries.

It would be interesting to consider generalizing our approach to a hierarchy of successively stronger approximations to the true safety predicate; we leave this as a direction for future work.

5 Auditing Max Queries

We now turn to the problem of auditing MAX queries over real-valued data.

We have a set of variables with labels $U = \{1, \ldots, n\}$; and each variable $i \in U$ has a value $y_i \in \mathbf{R}$. We are also given a collection of sets $S_1, \ldots, S_m \subseteq U$; with each S_p, we are given the maximum value over the variables in S_p. We will denote this maximum by $f(S_p)$.

The first problem we address is that of auditing: is the value of any of the variables in U determined by the information $\{(S_p, f(S_p))\}$? We provide an efficient algorithm to compute the set of all variables $i \in U$ that are determined.

For a variable i, let

$$\mu_i = \min\{f(S_p) : i \in S_p\}.$$

We will say that S_p is *i-extreme* if $i \in S_p$, and $\mu_i = f(S_p)$. Note that for every variable i, there is at least one i-extreme set. Conversely, for every set S_p, there is an i for which S_p is i-extreme — specifically, consider a variable $i \in S_p$ for which y_i is maximum.

Now we claim

Theorem 5.1 *The value of variable i is determined if and only if there exists a set S_p that is i-extreme, but is not ℓ-extreme for any $\ell \neq i$.*

Proof: First, consider an arbitrary variable ℓ; let S_p be an ℓ-extreme set, and let S_q be any set containing ℓ. Then we have

$$y_\ell \leq f(S_p) = \mu_\ell \leq f(S_q).$$

Moreover, if S_q is not ℓ-extreme, then the last inequality is strict.

Thus, if S_p is a set that is i-extreme but not ℓ-extreme for any $\ell \neq i$, we have $y_\ell \leq \mu_\ell < f(S_p)$ for every $y_\ell \in S_p \setminus \{y_i\}$. It follows that $y_i = f(S_p)$, and hence the value of variable i is determined.

We now consider the converse direction. To begin with, note that the setting $y_\ell = \mu_\ell$ for each ℓ is consistent with all the values $\{f(S_p)\}$. Indeed, $\mu_\ell \leq f(S_p)$ holds for every $\ell \in S_p$; and the variable ℓ for which y_ℓ attains the maximum value in S_p has $\mu_\ell = f(S_p)$.

Now, suppose variable i has the property that for every i-extreme set S_p, there is a variable $\ell(p) \neq i$ for which S_p is also $\ell(p)$-extreme. Suppose we take the setting of variables $\{y_\ell = \mu_\ell\}$ and then arbitrarily decrease the value of the variable i. We claim that all the values $\{f(S_p)\}$ remain the same. Certainly, no set S_p with $f(S_p) > \mu_i$ or $f(S_p) < \mu_i$ will change in value; and for any set S_p with $f(S_p) = \mu_i$, we have $\ell(p) \in S_p$ with $\mu_{\ell(p)} = \mu_i$, whence $f(S_p)$ will retain the value μ_i. ■

Notice that, by the above result, there is a polynomial algorithm for auditing MAX queries.

Generic safety. We now consider the *generic* notion of auditing discussed in the introduction. Given a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ as before, we will call this collection of sets *safe* if for *every* setting $y_1, \ldots, y_n$ of the variables in U, there is no y_i whose value is determined by the results of queries to $\mathcal{S}$.

We now provide a characterization of safe set systems. First, consider the following property of a set system $\mathcal{S} = \{S_1, \ldots, S_m\}$:

($*$) There exists $S_p \in \mathcal{S}$, and $S_{q_1}, \ldots, S_{q_t} \in \mathcal{S}$ so that

$$\left| S_p \setminus \left(\bigcup_r S_{q_r} \right) \right| = 1.$$

We claim

Theorem 5.2 *A set system is safe if and only if it does not have property ($*$).*

Proof: To prove the easier direction, suppose that the set system $\mathcal{S}$ has property ($*$), and choose $S_p, S_{q_1}, \ldots, S_{q_t} \in \mathcal{S}$ and $j \in U$ so that $S_p \setminus (\cup_r S_{q_r}) = \{j\}$. Suppose we define values for the variables so that $y_j = 1$ and $y_i = 0$ for all $i \neq j$. Then $f(S_p) = 1$ and $f(S_{q_r}) = 0$ for $s = 1, \ldots, t$; this implies that for any $(y'_1, \ldots, y'_n) \in \mathbf{R}^n$ consistent with these values, $y'_i \leq 0$ for $i \in S_p \setminus \{j\}$, whence $y'_j = 1$.

Conversely, suppose that $\mathcal{S}$ does not have property ($*$), and consider a setting $y = (y_1, \ldots, y_n) \in \mathbf{R}^n$ of the variables that determines the value of some variable j. We now define a setting $y' = (y'_1, \ldots, y'_n) \in \mathbf{R}^n$ that is consistent with the answers to all queries, but for which $y'_j \neq y_j$; this will be a contradiction. Let W_j denote the union of all sets in $\mathcal{S}$ that do not contain j. First, we set y'_j to an arbitrary value strictly less than y_j. Now, consider each set $S_p \in \mathcal{S}$ on which the value of f has changed; these are precisely the sets in which y_j was the unique maximum. Since $\mathcal{S}$ does not have property ($*$), and $S_p \setminus W_j$ is non-empty, it must have cardinality at least two, and hence contains an element $i_p \neq j$. We choose such an i_p and define $y'_{i_p} = y_j$. Finally, for any element k which is not equal to i_p for any p in the above construction, we set $y'_k = y_k$.

We introduce the following additional piece of notation: for a set S_r, we use $f(y|S_r)$ to denote the value $f(S_r)$ under the setting $y \in \mathbf{R}^n$, and we use $f(y'|S_r)$ to denote the value $f(S_r)$ under the setting $y' \in \mathbf{R}^n$. We claim that $f(y|S_r) = f(y'|S_r)$ for $r = 1, 2, \ldots, m$; this will conclude the proof. The crucial observation here is that if $y_i \neq y'_i$ for some $i \neq j$, then j appears in every set in which i appears. For if $y_i \neq y'_i$, it must be that $i = i_p$ for some p in the above construction; since $i \in S_p \setminus W_j$, we conclude that i does not appear in any set which omits j. Note also that for such an i, $y'_i > y_i$, since y_j was the unique maximum in this set S_p. Now, suppose by way of contradiction that $f(y|S_r) \neq f(y'|S_r)$ for some r. It follows that $j \in S_r$, and hence $f(y|S_r) \geq y_j$. If $f(y|S_r) > y_j$ or $f(y'|S_r) > y_j$, then the maximum in S_r must be attained by an element whose value did not change, and so $f(y|S_r) = f(y'|S_r)$. Thus, suppose $f(y|S_r) = y_j > f(y'|S_r)$. But in this case, y_j must have been the unique maximum in S_r, and so there is an element $i_r \in S_r \setminus \{j\}$ for which we set $y'_{i_r} = y_j$; this contradicts our supposition that $f(y'|S_r) < y_j$. ■

We note that property ($*$) —and therefore safety— can be tested in polynomial time, as follows: For each variable j, we form the set W_j consisting of the union of all sets in

$\mathcal{S}$ that do not contain j. Then, for each set S_p, and each $j \in S_p$, we determine whether $S_p \setminus W_j = \{j\}$.

References

[1] N. Adam, J. Wortman "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Comp. Surveys, 21,* 4, 1989, pp. 515–556.

[2] L. Beck "A Security Mechanism for Statistical Databases," *ACM TODS, 5*, 3, pp. 316–338, 1980.

[3] F. Chin "Security in Statistical Databases for Queries with Small Counts," *ACM TODS, 3*, 1, pp. 92–104, 1978.

[4] F. Chin, G. Ösoyoglu "Statistical Database Design," *ACM TODS, 6*, 1, pp. 113–139, 1981.

[5] F. Chin, G. Ösoyoglu "Auditing and Inference Control in Statistical Databases," *IEEE SE-8*, 1, pp. 574–582, 1982.

[6] F. Chin, G. Ösoyoglu "Security in Partitioned Dynamic Statistical Databases," *Proc. IEEE COMPSAC*, pp. 594–601.

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

[8] T. Dalenius "A Simple Procedure for Controlled Rounding," *Statistik Tidsktift, 3*, pp. 202–208, 1981.

[9] D. Dobkin, A. Jones, R. Lipton "Secure Databases: Protection Against User Influence," *ACM TODS, 4*, 1, pp. 97–106, 1979.

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

[11] C. Liew, W. Choi, C. Liew "Data Distortion by Probability Distribution," *ACM TODS, 10*, 3, pp. 395–411, 1985.

[12] G. Ösoyoglu, F. Chin "Enhancing the Security of Statistical Databases with a Question-Answering System and a Kernel," *IEEE SE-8*, 3, pp. 223–234, 1982.

[13] S. Reiss "Practical Data Swapping: The First Steps," *ACM TODS, 9*, 1, pp. 20–37, 1984.

[14] N. Rowe, "Diophantine inference from statistical aggregates on few-valued attributes," *Proc. IEEE Conf. on Data Engineering*, 1984, pp. 107–110.

[15] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, 1986.

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