Linear Queries in Statistical Databases

M. D. SCHWARTZ
Tektronix, Inc.
and
D. E. DENNING AND P. J. DENNING
Purdue University

A database is compromised if a user can determine the data elements associated with keys which he did not know previously. If it is possible, compromise can be achieved by posing a finite set of queries over sets of data elements and employing initial information to solve the resulting system of equations. Assuming the allowable queries are linear, that is, weighted sums of data elements, we show how compromise can be achieved and we characterize the maximal initial information permitted of a user in a secure system. When compromise is possible, the initial information and the number of queries required to achieve it is surprisingly small.

Key Words and Phrases: confidentiality, database security, data security, inference, linear query, secure query functions, statistical database.
CR Categories: 3.7

1. INTRODUCTION

Statistical databases contain sensitive information about individuals. Their purpose is to provide summary statistics about groups of people, while permitting only authorized access to the records of any one individual. But this objective is difficult to meet, for seemingly innocuous summaries contain vestiges of the original data. By correlating enough summaries, an intruder may infer confidential data. We understand much more about controlling access to individual records than we do about controlling inference.

A *query* is a function applied to a given subset of the records in the database; this subset is called the *query set.* Given the form of allowable queries and an assumption about the information known initially by inquirers, we wish to specify whether or not the database is secure. A proof that a database can be compromised amounts to exhibiting a method for constructing a sequence of queries whose responses imply the value associated with a given, arbitrary key. A proof that a database is secure is much more difficult, for it requires us to demonstrate that no such sequence exists.

This work was supported in part by the National Science Foundation under and by Purdue David Ross Grant 7994-56-13985.
Authors' addresses: M.D. Schwartz, Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077; D.E. Denning and P. J. Denning, Department of Computer Science, Purdue University, West Lafayette, IN 47907.

Inference has long been a concern of census statisticians, who have developed sophisticated methods for choosing samples for analysis. The principle is to apply queries against a random subfile; deducing an element of the subfile is of little help because the relation between the subfile and the full population is unknown. Unfortunately, many contemporary applications center on small or medium databases where random sampling cannot be used.

Much existing literature on inference applies to *counting queries,* which return the number of individuals satisfying given characteristics, e.g., "Female astronomers over 40 years of age" [10–12, 15]. These studies have shown that the danger of compromise is greatest when the system responds for small query sets. Some protection is afforded when the system refuses answers for small query sets or for query sets whose differences are small [1, 7, 16]. However, this protection is far from complete [4, 17].

Another class of easily compromised queries is *selection queries.* Such a query returns a specific value from the query set, such as the largest, the smallest, or the median. These queries are easily compromised even if the database system somehow enforces a restriction that no two query sets overlap by more than one record [2, 3, 6, 14].

The subject of this paper is the class of *linear queries.* Each of these queries returns a weighted sum of elements in the query set. Our results extend the results of Dobkin, Jones, and Lipton [6], whose queries computed simple (unweighted) sums of elements in the query sets. One aspect of the results is striking: a user who knows as little as the value associated with a single key can often achieve a full compromise of the entire database—with a query sequence whose length is not much longer than the number of stored data elements.

As a running example, we will use the database displayed in Table I, referred to hereafter as the "runner's database." It contains facts about foot racers. Users of the database might be allowed to discover general properties such as average running paces; however, they are not supposed to be able to deduce facts about individual runners from these queries.

2. MODEL OF A DATABASE

A statistical database contains a set of records about individuals. Each record contains an identifier field, one or more category fields, and one or more data fields. Some data fields may also be category fields.

The identifier field contains a unique identifier for the record. Reading and writing in single records via identifiers is controlled by an access mechanism. A user presents the symbolic name of a record, the system looks up the identifier corresponding to that name and checks authorization.

A *data element* is a value stored in a given data field of a given record. The size of the database is the total number, N, of data elements in it. In a *partitioned database,* the data elements are divided into mutually exclusive, collectively exhaustive classes. The jth class, denoted Y_j, contains N_j data elements, and $N = \sum_j N_j$. The columns of the runner's database are examples of classes. A *key* $p = (i, j)$ identifies a data element, which is denoted either x_p or x_{ij}.

A *query set* is a collection of data elements identified by a list of keys or by a characteristic formula. (A characteristic formula is a logical formula which is matched against the values in category fields.) A *query* computes some value for

Table I. Runner's Database

Keys	1	2	3	4	5	6	7
			Prior 8 weeks				
i *j*	Max VOX[a]	Train pace (seconds/mile)	Total miles	Longest run	Fastest 1 mile (seconds)	Fastest 10 miles (seconds)	Year of birth
1 Smith	68	380	680	22	260	3183	1948
2 Jones	61	405	530	18	287	3520	1950
3 Burns	56	440	460	20	316	3818	1940
4 Cohen	48	485	410	18	368	4447	1930
5 Cook	49	470	375	20	360	4394	1962
6 Bloom	54	440	430	20	328	3980	1948
7 King	53	440	405	20	334	4072	1943
8 Frank	72	370	705	23	251	2997	1947

[a] "Max VOX" is the maximum volume of oxygen processable in milliliters per kilogram body weight per minute.

the query set. To prevent trivial compromises a query program will not respond if the query set contains fewer than k data elements, where $k \geq 2$ is a parameter of the system.

Example 1. The runner's database contains eight records and seven data fields, a total of $N = 56$ data elements. The keys are pairs (i, j) where i is a row index and j is a column index. The user would specify symbolic name pairs which would be translated to keys internally; for example (Smith, Max VOX) translates to (1, 1) and (Cohen, Year of birth) to (4, 7). The corresponding data elements are $x_{11} = 68$ and $x_{47} = 1930$. A possible key-specified query is "What is the average train pace for (Smith, Jones, Frank)?" The same query expressed with a characteristic formula is "What is the average train pace among runners whose Max VOX is greater than 60?"

This paper analyzes threats to statistical database security for key-specified query sets. Security for characteristic-specified query sets is treated in other papers [4, 17]. A summary of important results for both types of query sets is given in [5].

Compromising the Database

A user has *identified a key* (i, j) as soon as he knows the value of its data element x_{ij}. A user usually has some *initial information* about the database: he may know operational details of some query programs, or the values of some data elements. Relative to an assumption about initial information, a database is insecure if it is possible to identify keys not known initially. Identifying previously unknown keys is a *compromise*, and identifying all the keys is a *full compromise*. If the database is insecure there will exist a finite sequence of queries $q_1, \ldots, q_m$, query sets $X_1, \ldots, X_m$, and responses $v_1, \ldots, v_m$ for which it is possible to solve the system of equations

$$\begin{aligned} v_1 &= q_1(X_1) \\ v_2 &= q_2(X_2) \\ &\cdots \\ v_m &= q_m(X_m) \end{aligned} \tag{1}$$

for some unknown key.

We are interested in characterizing the amount of initial information that can be tolerated in a secure system, and the amount of work of compromise in an insecure system.

Example 2. A regression analysis has yielded a formula to predict the time a runner will take to run a marathon [8]:

$$q(i) = \sum_{j=1}^{4} a_j x_{ij} + 319 \text{ seconds.} \tag{2}$$

Suppose that this query is available in the database and that the user knows the coefficients a_j (e.g. from reading the article). If that user also knows any three of x_{i1}, x_{i2}, x_{i3}, and x_{i4}, he can use the system's response to $q(i)$ to solve eq. (2) for the fourth.

Linear Queries

To pose a query, a user specifies a list of distinct keys $p_1, \ldots, p_k$; the system determines the query set $\{x_{p_1}, \ldots, x_{p_k}\}$ and computes the value of the query function for it. To keep the notation simple, we will show the query set as argument to the query, rather than the list of keys. Thus $q(z_1, \ldots, z_k)$ denotes a query over a list of distinct keys that specify the data elements $z_1, \ldots, z_k$. A *linear query* has the form

$$q(z_1, \ldots, z_k) = \sum_{j=1}^{k} a_j z_j \tag{3}$$

for some fixed $k \geq 2$ and fixed "query weights" a_j. This form of query has been called a "weighted sum query" [18].

The assumption that the query set size is fixed at k elements for all queries is a degenerate case of the restriction that each query set contains at least k elements. When query set size can vary, as for characteristic-specified queries, compromise can be even simpler [4]. Our objective here is to show how easy compromise can be even when the intruder cannot exploit variation in query set size.

A *sum query* is a linear query with unit weights (all $a_j = 1$). Such queries are used to compute averages. Dobkin, Jones, and Lipton [6] developed upper and lower bounds on the number of sum queries sufficient to compromise under the restriction that no two query sets can overlap by more than r elements. For N data elements with $r = 1$ and no initial information, compromise is possible within $2k - 1$ queries if $N \geq N_2 = k^2 - k + 1$; however, compromise is impossible when $N < N_1 = (k^2 + k)/2$. Whether or not compromise is possible when $N_1 \leq N < N_2$ was not resolved. Davida et al. tightened these bounds [2].

Kam and Ullman studied the security of a class of key-matching sum queries [13]. Each key is a bit-string of b bits, and there are 2^b data elements. To make a query, the user specifies the values of some set of a bits, where $a \leq b$; the query program returns the sum of the data elements whose keys match in the given a positions. The database is secure whenever users have no initial information, if a

$< b$, and if the ranges of values of the data elements are unknown. Otherwise compromise may be possible.

A *partitioned query* is a linear query in a partitioned database with the restriction that the jth key specifies a data element from the jth class; there are k classes in all. Equation (2) is an example with four classes.

Our results for linear queries and partitioned queries are outlined in the next sections. The proofs are given in the Appendices.

3. SECURITY OF LINEAR QUERIES

We consider queries of the form of eq. (3) applied to a database of N elements. All queries use the same value of k and the same weights a_j.

We observe first that, knowing one weight a_j and one data element x, we can compromise the entire database. Let q_1 and q_2 be the responses to the two queries

$$q_1 = q(z_1, \ldots, z_{j-1}, x, z_{j+1}, \ldots, z_k)$$

$$q_2 = q(z_1, \ldots, z_{j-1}, y, z_{j+1}, \ldots, z_k)$$

which differ only in their use of data elements x and y in the jth position. Equation (3) shows that

$$(q_1 - q_2) = a_j(x - y), \tag{4}$$

which can be solved for y. Now the entire database is vulnerable: we pose q_1 once and a new q_2 for each of the $N - 1$ unknown data elements, effecting a full compromise with N queries and $N - 1$ applications of eq. (4). As soon as two data elements, x and y, are known, we can also use eq. (4) to solve for any unknown weight.

Example 3. Suppose that the runner's database implements the query

$$q(z_1, z_2, z_3) = 0.2z_1 + 0.5z_2 + 0.3z_3.$$

A user knows that the weight of the first key is 0.2 and that Smith's Max VOX is 68; thus the initial information is

$$a_1 = 0.2$$

$$x_{11} = 68.$$

To determine Jones's Max VOX, the user proceeds as follows. He poses the two queries $q(x_{11}, x_{12}, x_{13})$ and $q(x_{21}, x_{12}, x_{13})$, to which the system will respond, respectively,

$$q_1 = 407.6$$

$$q_2 = 406.2.$$

Equation (4) can be used to solve for x_{21}:

$$\begin{aligned} x_{21} &= x_{11} - (q_1 - q_2)/a_1 \\ &= 68 - 1.4/0.2 \\ &= 68 - 7 \\ &= 61 \end{aligned}$$

which is Smith's Max VOX.

This reasoning suggests that security, if it is possible at all, must require keeping all the weights secret. But a more complex argument, given in Appendix A, shows that we can pose a set of queries which yield a set of equations in the unknowns a_j. If these equations are linearly independent, we can solve for the a_j and then use the known value of one data element to compromise the rest of the database. In other words, knowing only one data element and none of the weights often suffices for a compromise. The exact statement of the result is:

COMPROMISING WEIGHTED SUMS. *Suppose that x_p is known for some key p. With no more than $k(k + 1)$ queries it may be possible to determine $x_{p1}, \ldots, x_{pk}$ for k additional keys $p1, \ldots, pk$. The remaining $N - k - 1$ data elements can be deduced with an additional $N - k - 1$ queries using the method of eq.* (4). *Thus full compromise is possible within $N + k^2 - 1$ queries.*

The technique will be illustrated in an example below. These arguments lead to the conclusion that security depends on denying the inquirer any initial information about the database, for example, by restricting his access to records about which he knows nothing. In Appendix B we prove that such a restriction would work by showing the impossibility of solving any system of equations for any weight or any data element:

SECURE WEIGHTED SUMS. *If the inquirer knows no data element and none of the weights, the database is secure under his queries.*

Unfortunately there is no way to be absolutely sure that a user knows nothing about the records to which his access has been restricted. Even if such an idea were enforceable, the presence of other types of queries can invalidate this result. The unweighted-sum compromise of Dobkin, Jones, and Lipton, for example, requires no initial knowledge [6].

The next example illustrates the method of compromising a database using weighted-sum queries when one data element is known.

Example 4. Let $y_i = x_{i1}$ denote the Max VOX for runner i. Let $k = 2$ and suppose $y_1 = 68$ is known. The database implements the query

$$q(z_1, z_2) = 2z_1 + 3z_2$$

but the weights $(a_1, a_2) = (2, 3)$ are unknown. Form three sets of two queries:

$$\begin{aligned} q_{10} &= q(y_2, y_3) = 290 & \quad q_{11} &= q(y_3, y_2) = 295 \\ q_{20} &= q(y_1, y_3) = 304 & \quad q_{21} &= q(y_3, y_1) = 316 \\ q_{30} &= q(y_1, y_2) = 319 & \quad q_{32} &= q(y_2, y_1) = 326 \end{aligned}$$

The numbers show the system's responses. Let $A = a_1 + a_2$ and observe that

$$\begin{aligned} q_{10} + q_{11} &= A(y_2 + y_3) = 585 = Q_1 \\ q_{20} + q_{21} &= A(y_1 + y_3) = 620 = Q_2 \\ q_{31} + q_{31} &= A(y_1 + y_2) = 645 = Q_3 \end{aligned}$$

whereupon

$$A = Q_1/(y_2 + y_3) = Q_2/(y_1 + y_3) = Q_3/(y_1 + y_2).$$

This can be manipulated into two equations in unknowns y_1 and y_2:

$$-Q_2y_2 + (Q_1 - Q_2)y_3 = -Q_1y_1$$

$$Q_2y_2 - Q_3y_3 = (Q_3 - Q_2)y_1.$$

For the given numbers

$$-620\,y_2 - 35\,y_3 = -\,(585)(68)$$

$$620\,y_2 - 645\,y_3 = (25)(68).$$

It is easily verified that $y_2 = 61$ and $y_3 = 56$ is the solution. We may solve for a_1 and a_2 using the equations

$$a_1y_2 + a_2y_3 = q_{10}$$

$$a_1y_1 + a_2y_3 = q_{20}.$$

To find any other y_i we solve $q(y_1, y_i) = 136 + 3y_i$ once the system's response is known.

4. PARTITIONED QUERIES

We consider weighted-sum queries with the restriction that each weight a_j is associated with a separate class Y_j of data elements, and that the jth key in a query's specification list must refer to the class Y_j. We denote the number of data elements in Y_j by N_j, so that $N = \Sigma_j N_j$. The columns of the runner's database are examples of classes.

We observe first that the method of eq. (4) can be used to compromise all of class Y_j if the weight a_j and some element x of that class both are known. In all, N_j queries are required for this. If we know all the weights and one data element in each class, we can compromise the entire database with N queries.

Suppose that we know all k of the weights and data elements from only $k - 1$ classes. We can determine an element from the unknown class (say Y_j) with one query using the known elements from the other classes (say $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_k$):

$$q(z_1, \ldots, z_k) = a_1z_1 + \cdots + a_kz_k. \tag{5}$$

After the system responds with the value of this query, z_j is the only unknown quantity in this equation. It follows that full compromise is possible with $N + 1$ queries when all weights are known and data elements for all but one class are known.

Similarly, if we know one element from each class, and all but one of the weights, we can apply eq. (5) using the known $z_1, \ldots, z_k$; the missing weight is the only unknown quantity. Full compromise is again possible with $N + 1$ queries.

The method illustrated in Example 4 was able to effect a compromise when none of the weights was known and the entire database was a single class. This method depends on permuting the same keys among different positions in the query specification lists. Partitioning the databases prevents these permutations and makes compromise more difficult. In Appendix C we prove that: If one knows all the weights, one data element from each of $k - 2$ classes, and no data element from the other two classes, then full compromise of the $k - 2$ known classes is possible. The two unknown classes are secure.

The proof shows that no system of equations constructed contains enough information to solve for data elements in the unknown classes. The proof of this result shows further that every unknown class is secure provided that at least two classes are unknown initially.

These results show that keeping the weights secret is essential to security especially if users may know data elements from some of the classes. However, if other types of queries are possible in the same database, compromise may be possible without prior knowledge. For example, DeMillo, Dobkin, and Lipton have shown that $0(k^2)$ selection queries (median, maximum, minimum, etc.) can determine an element from an unknown class [3]; such queries can provide the initial information needed to compromise with partitioned queries (see also [17]).

6. CONCLUSIONS

Our results have shown that the maximum tolerable information to prevent compromise using weighted-sum or partitioned-weighted-sum queries is quite low. Security under weighted-sum queries can be assured only if users have no prior knowledge of any data element; the weights must be kept secret. Security in partitioned databases can be assured only if at least two of the weights or at least two of the classes are kept secret. It is not always possible to keep weights secret, and it may be impossible to enforce the restriction that the user knows no data element.

When the inquirer has sufficient initial information for a compromise, his task is easy. At worst, he needs to pose one query for each unknown data element in the (class of the) database which he is compromising, plus possibly a small additional but fixed number of queries. If the database supports other types of queries such as selection queries, the user may be able to deduce unknown data elements with small cost, before using algorithms of this paper for a compromise.

Compromise of statistical databases is usually possible, and it is cheap.

APPENDIX A. COMPROMISING LINEAR QUERIES

We wish to show that knowing one data element, say z_1, may be sufficient to compromise a database under linear queries. The compromise proceeds in three parts. First, $k(k + 1)$ queries are used to derive $k + 1$ equations whose solution yields k additional data elements $z_2, \ldots, z_{k+1}$. Then the $k + 1$ known data elements are used to derive k more equations whose solution is the weights $a_1, \ldots, a_k$. Finally, the method of eq. (4) is used to deduce the remaining $N - (k + 1)$ unknown data elements. Full compromise is achievable with $N - (k + 1) + k(k + 1) = N + k^2 - 1$ queries in all.

Let $Z = (z_1, \ldots, z_{k+1})$ be a list of data elements for which z_1 is known. Let Z_i be Z with z_i deleted, and let Z_{ij} denote the jth one-step cyclic permutation of Z_i. (Note that $Z_{i0} = Z_{ik}$.) Let q_{ij} denote the response to the query $q(Z_{ij})$, where $i = 1, \ldots, k + 1$ and $j = 0, \ldots, k - 1$. Note that $k(k + 1)$ queries are needed to determine all the q_{ij}.

Each element of Z_i is permuted into each of the k positions by the queries $q(Z_{i0}), \ldots, q(Z_{i,k-1})$. If $S = z_1 + z_2 + \cdots + z_{k+1}$ then the sum of elements in Z_i is $S - z_i$ and the sum of these permuted queries is

$$Q_i = \sum_{j=0}^{k-1} q_{ij} = \sum_{j=0}^{k-1} a_j(S - z_i) = A(S - z_i)$$

where $A = a_1 + \cdots + a_k$. It follows that $Q_i/(S - z_i) = A = Q_{i+1}/(S - z_{i+1})$. This gives a set of k equations in the k unknowns $z_2, \ldots, z_{k+1}$:

$$S(Q_i - Q_{i+1}) + Q_{i+1}z_i - Q_i z_{i+1} = 0, \qquad i = 1, \ldots, k.$$

If these equations are linearly independent, their solution will yield the values of $z_2, \ldots, z_{k+1}$. (If these equations are not linearly independent, nothing can be concluded about the security of the database, because another choice of $z_2, \ldots, z_{k+1}$ might produce a solvable set of equations.) We can use the known values of $z_1, \ldots, z_k$ to invert the equations corresponding to the queries $q(Z_{i0}), \ldots, q(Z_{ik})$ and find the unknown coefficients:

$$\begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_k \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} q_{10} \\ q_{20} \\ \vdots \\ q_{k0} \end{bmatrix}$$

Now the method of eq. (4) can be applied to find the remaining data elements.

APPENDIX B. SECURE LINEAR QUERIES

We suppose that the user has no initial knowledge about any data element. Our objective is proving that it is impossible for him to find enough linearly independent equations to solve for any unknown data element or query weight.

Consider any m queries $q_1, \ldots, q_m$, among which appear the data elements $Z = \{z_1, \ldots, z_s\}$. We assume $m \geq k + s$ since we can always pose more queries without impairing our ability to compromise. We will apply the implicit function theorem (IFT) [9], which states that a set of functions $q_1, \ldots, q_m$ over variables $x_1, \ldots, x_n$ are mutually independent if the $\det(M) \neq 0$, where

$$M = \begin{bmatrix} \dfrac{\partial q_1}{\partial x_1} & \cdots & \dfrac{\partial q_1}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial q_m}{\partial x_1} & \cdots & \dfrac{\partial q_m}{\partial x_n} \end{bmatrix}$$

For our problem, the queries $q_1, \ldots, q_m$ depend on the $n = k + s$ quantities $a_1, \ldots, a_k$ and $z_1, \ldots, z_s$. The matrix is

$$M = \left[\begin{array}{ccc|ccc} z_{11} & \cdots & z_{1k} & b_{11} & \cdots & b_{1s} \\ & \cdots & & & \cdots & \\ z_{m1} & \cdots & z_{mk} & b_{m1} & \cdots & b_{ms} \end{array}\right]$$

where z_{ij} is the member of Z appearing in the jth position of the ith query, and

$$b_{it} = \begin{cases} a_j, & \text{if } z_{ij} = z_t \\ 0, & \text{otherwise.} \end{cases}$$

Let $r = \operatorname{rank}(M)$. We observe that $r < k + s$ because column $k + p\,(p = 1, \ldots, s)$ can be written as a linear combination of the others:

$$\sum_{j=1}^{k} a_j z_{ij} - \sum_{\substack{t=1 \\ t \neq p}}^{s} z_t b_{it} = b_{ip} z_p, \qquad i = 1, \ldots, m.$$

Note that $z_{ij} = z_t$ if and only if $b_{it} = a_j$, which implies that the nonzero terms in the second sum cancel all but $b_{ip}z_p$ in the first. Since $r < k + s$, $\det(M) = 0$, which implies that some of $q_1, \ldots, q_m$ are redundant.

Consider an $r \times r$ submatrix N of M formed by taking any r linearly independent rows, and any r linearly independent columns except column $k + p$. Renumber the queries so that $q_1, \ldots, q_r$ correspond to the rows of N. By construction $\det(N) \neq 0$, but any determinant of $r + 1$ rows and columns of M must be zero. The IFT shows that $q_1, \ldots, q_r$ are mutually independent and that each query among $q_{r+1}, \ldots, q_{k+s}$ can be expressed in terms of $q_1, \ldots, q_r$ [9].

We will show that x_p is not a function of any of the queries $q_1, \ldots, q_r$. To do this we treat x_p as a new function of the same quantities (a_j or z_t) on which $q_1, \ldots, q_r$ depend according to the matrix N; we add the appropriate r + 1st row and column of N:

$$N' = \left[\begin{array}{ccc|c} & & & \frac{\partial q_1}{\partial x_p} \\ & N & & \vdots \\ & & & \frac{\partial q_r}{\partial x_p} \\ \hline 0\;0 & \cdots & 0 & 1 \end{array}\right]$$

It is not hard to see that $\det(N') = (-1)^r \det(N) \neq 0$, which implies that x_p is not a function of any quantity on which $q_1, \ldots, q_r$ depends and, therefore, that x_p is not a function of any of $q_1, \ldots, q_r$.

APPENDIX C. SECURE PARTITIONED QUERIES

We wish to show that if all weights are known but no element from at least two classes is known, then the unknown classes are secure. The classes from which we know at least one element are vulnerable to compromise via eq. (4).

It is sufficient to prove security when exactly two classes are unknown since additional unknown classes only make compromise more difficult. Let Y_1 and Y_2 be the two unknown classes, and let $y_{11}, \ldots, y_{1N_1}$ be the elements of Y_1 and $y_{21}, \ldots, y_{2N_2}$ be the elements of Y_2. Because it is assumed that data from the known classes are functionally independent of data in the unknown classes, $Y_3, \ldots, Y_k$ are irrelevant to compromising Y_1 and Y_2. Therefore, we need only to prove that queries of the form

$$q(z_1 z_2) = a_1 z_1 + a_2 z_2$$

cannot reveal any element of Y_1 or Y_2.

Suppose that queries $q(z_{i1}, z_{i2})$ have been posed for $i = 1, \ldots, m$, yielding responses $q_1, \ldots, q_m$. These queries can be represented by the matrix equation

$$\left[\begin{array}{ccc|ccc} a_{11} & \cdots & a_{1N_1} & b_{11} & \cdots & b_{1N_2} \\ & \cdots & & & \cdots & \\ a_{m1} & \cdots & a_{mN_1} & b_{m1} & \cdots & b_{mN_2} \end{array}\right] \left[\begin{array}{c} y_{11} \\ \vdots \\ y_{1N_1} \\ \hline y_{21} \\ \vdots \\ y_{2N_2} \end{array}\right] = \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix}$$

where

$$a_{ij} = \begin{cases} a_1, & \text{if } z_{i1} = y_{1j} \\ 0, & \text{otherwise} \end{cases}$$

$$b_{ij} = \begin{cases} a_2, & \text{if } z_{i2} = y_{2j} \\ 0, & \text{otherwise.} \end{cases}$$

Now let A_j represent the jth column in the left part of the p by $(N_1 + N_2)$ coefficient matrix, and B_j represent the jth column of the right part. Note that

$$\frac{1}{a_1} \sum_{j=1}^{N_1} A_j - \frac{1}{a_2} \sum_{j=1}^{N_2} B_j = 0.$$

This means that the columns are not independent and that the rank of the coefficient matrix is less than $N_1 + N_2$, i.e., the system of equations cannot be solved for any of the $N_1 + N_2$ unknowns.

REFERENCES

1. Chin, F.Y. Security in statistical databases for queries with small counts. *ACM Trans. Database Syst. 3,* 1 (March 1978), 92–104.
2. Davida, G.I., et al. Data base security. TR-CS-76-14, Dept. E.E. and Comptr. Sci., U. of Wisconsin, Madison, Wis., July 1976
3. DeMillo, R.A., Dobkin, D., and Lipton, R.J. Even data bases that lie can be compromised. *IEEE Trans. on Software Engrg. SE-4,* 1 (Jan. 1977), 73-75.
4. Denning, D.E., Denning, P.J., and Schwartz, M.D. The tracker: A threat to statistical database security. *ACM Trans. Database Syst: 4,* 1 (March 1979), 76–96.
5. Denning, D.E. Are statistical data bases secure? Proc. AFIPS 1978 NCC, Vol. 47, AFIPS Press, Montvale, N.J., pp. 525-530.
6. Dobkin, D., Jones, A.K., and Lipton, R.J. Secure databases: Protection against user inference. *ACM Trans. Database Syst. 4,* 1 (March 1979), 97-106.
7. Fellegi, I.P., and Phillips, J.L. Statistical confidentiality: Some theory and applications to data dissemination. *Anals Econ. and Social Measmt. 3,* 2 (April 1974), 399–409.
8. Foster, C., and Daniels, J. Running by the numbers. *Runner's World* (July 1975), 14-17.
9. Garsoux, J. *Analyse Mathematique.* Dunod, Paris, 1968, p. 444.
10. Haq, M.I. Security in a statistical data base. *Proc. Amer. Soc. Inform. Sci. 11* (1974), 33–39.
11. Haq, M.I. Insuring individual's privacy from statistical data base users. Proc. AFIPS 1975 NCC, Vol. 44, AFIPS Press, Montvale, N.J., pp. 941-946.

12. HOFFMAN, L.J., AND MILLER, W.F. Getting a personal dossier from a statistical data bank. *Datamation 16,* 5 (May 1970), 74-75.
13. KAM, J.B., AND ULLMAN, J.D. A model of statistical databases and their security. *ACM Trans. Database Syst. 2,* 1 (March 1977), 1-10.
14. REISS, S.P. Medians and data base security. Comptr. Sci. Dept., Brown U., Providence, R.I., Oct. 1977.
15. SCHLÖRER, J. Indentification and retrieval of personal records from a statistical data bank. *Methods of Inform. in Medicine 14,* 1 (1975), 7-13.
16. SCHLÖRER, J. Confidentiality of statistical records: A threat monitoring scheme for on-line dialogue. *Methods of Inform. in Medicine 15,* 1 (1976), 36-42.
17. SCHWARTZ, M.D. Inference from statistical data bases. Ph.D. Th., Comptr. Sci. Dept., Purdue U., W. Lafayette, Ind., Aug. 1977.
18. SCHWARTZ, M.D., DENNING, D.E., AND DENNING, P.J. Securing data bases under linear queries. IFIP Congress 77, North-Holland Pub. Co., Amsterdam, 1977, pp. 395-398.

Received January 1977; revised October 1978