Secure Date Bases; Protection Against User Inference

David Dobkin,[1] Anita K. Jones,[2]
and Richard Lipton[3]

Research Report #65

April 1976

[1] Yale University.

[2] Carnegie-Mellon University.
Supported in part by the National Science Foundation under contract DCR-75-07251.

[3] Yale University.
Supported in part by the National Science Foundation under contract DCR-74-24193.

Abstract

Users may be able to compromise data bases by asking a series of questions and then inferring new information from the answers. The complexity of protecting a data base against this technique is discussed here.

I. Introduction

In computer systems we encode information in data bases. We often want to control what information a user can obtain from these data bases. For example, we may wish to control the use of a census data base so that, although it contains records describing individuals, only statistical information is available. No sequence of queries should be sufficient to deduce exact information about any individual described in the data base. Determining and then enforcing a policy specifying what information in a data base can be given in response to queries is the data base security problem [5,6].

Security is also an issue for operating systems; unfortunately, the solutions for operating systems are not sufficient to solve the data base security problem. Most operating system protection mechanisms are "access control mechanisms" [4], that is, they enforce rules about who can perform what operation or access what information. For example, users can access file objects via READ, WRITE, SORT, DELETE FILE, or APPEND. But different users may be permitted different access to individual files. While user A may be permitted to READ and WRITE File X, user B may be able only to READ and APPEND it. And user A may be permitted only to READ File Y, while user B may both READ and WRITE that one. There are two common schemes for effecting such protection mechanisms; one is the "authority lists mechanism" used in most file systems (e.g. the MULTICS file system [10]) and the other is the capability mechanism [8,10,11].

In operating systems, protection mechanisms allow different users different access to an object; they allow some users to read part or all of the contents of a file, others to alter it in perhaps limited ways. In data bases, all users are essentially performing read access. An access control mechanism that only distinguishes between read and alter accesses is not useful. Thus the operating system approach is not sufficient for the data base problem.

Another contrast between data bases and operating systems concerns queries that involve many data elements. In the operating system, a complex operation can be broken down into a set of accesses to individual objects and each access permission determined independent of the others. In a data base, a decision must be made whether the entire query should be permitted in the first place. This decision depends not only on the relationship of data elements being interrogated but also on the query history, the information that has already been divulged to the user.

Newer access control mechanisms take into account the flow of information out of one object and into another as part of the effect of an access. These access control mechanisms incorporate a notion of the interdependence of objects [3,7]. Yet even such sophisticated mechanisms make no interpretation of the content of the data base and have no notion of a history of information already given out. We conclude that such mechanisms are not appropriate tools for solving the data base security problem.

Let us now restate the data base security problem: There is a set of data elements in the data base called the UNKNOWN set that user U is not permitted to know. For some reason, perhaps as a result of previous queries, user U knows a set of data elements called the KNOWN set. Some elements in KNOWN may not be explicitly encoded in the data base. User U asks a sequence of queries q1,q2,...,qn and enlarges his set of KNOWN data elements. The security of the data base is compromised if the KNOWN and the UNKNOWN sets intersect.

We will now proceed by introducing an example to highlight the issues germane to data base security.

Example: The following data describes fund raising for major political parties. C1,...,C9 are specific contributors with the following attributes:

			Favoritism	
	Business	*Political*	*Shown by*	*Geographic*
Contributor	*Area*	*Leaning*	*Administration*	*Area*
C1	Steel	Democrat	High	Northeast
C2	Steel	Republican	Medium	West
C3	Steel	Independent	Low	South
C4	Sugar	Democrat	Medium	Northeast
C5	Sugar	Republican	Low	Northeast
C6	Sugar	Independent	High	West
C7	Oil	Democrat	Low	South
C8	Oil	Republican	High	South
C9	Oil	Independent	Medium	West

Suppose that the only data that can be obtained from the data base is the sum given by all contributors sharing a common attribute -- contributions from the steel industry (C1 + C2 + C3) or contributions from those with Republican leanings (C2 + C5 + C8). The information that can be obtained from all possible queries is listed in Table I.

The political fund data base is considered secure if the precise contribution of an individual cannot be determined. Is this data base secure?

Contributing Group	*Amount*
Steel	270,000
Sugar	120,000
Oil	540,000
Democrats	186,000
Republicans	564,000
Independents	180,000
High favoritism	510,000
Low favoritism	174,000
Medium favoritism	246,000
Northeast	90,000
West	330,000
South	510,000

Table I

No, we can compute that C1 gave $60,000. Values for contributors C2,...,C9 can also be computed on the basis of the query responses listed in Table I.

We are interested in obtaining criteria that allow us to determine exactly when a data base can be compromised.

II. Basic Concepts

We are interested in the question of determining the security of data bases. We now define precisely what this means by presenting a general model. This model is an abstraction of the concept of data base; we do not suggest that it be used in place of, say, the relational data base model [1]. But we do propose this model as realistic for our discussion.

Definition: A *data base* D is a function from $\{1,\ldots,n\}$ to $\mathbf{N}$, the natural numbers. n is the number of elements or objects in the data base; $\mathbf{N}$ is the set of possible attributes.

In our fund-raising example, D(1) is $60,000. D(i) is the contribution of Ci. We will often use the following notation for data bases. Instead of defining D explicitly we just say that $\{d_1,\ldots,d_n\}$ is a data base. We mean of course that $D(i) = d_i$ for $1 \leq i \leq n$.

We will now define "query" and "compromise."

Definition: Fix n as the number of objects in the data base. A *query* q is a function of n variables. If $D = \{d_1,\ldots,d_n\}$ is a data base and q is a query, then $q(D) = q(d_1,\ldots,d_n)$ is the result of the query q on the data base D.

In our example $q(d_1,\ldots,d_9)$ is an allowed query provided

$$q(d_1,\ldots,d_9) = \sum_{k\in A} d_k$$

where A is a set of contributors that corresponds to an entry in Table I. Thus there are exactly 12 queries of this form.

A *security problem* has several components:

(1) A particular data base $D = \{d_1,\ldots,d_n\}$ is given.

(2) A subset D_0 of D is given. We interpret $d_i \in D_0$ as meaning that d_i is known to the user before he begins his queries.

(3) A set of queries is given. We assert that not all sequences of queries are allowed. (In section III we restrict the "overlap" of queries.)

Given these, we are to determine whether or not there is an allowed sequence of queries that can determine the value of some $d_i \notin D_0$. Thus a sequence of allowed queries $q_1,\ldots,q_m$ *compromises* a data base provided there is an i such that, for any data base D' with the same responses to the queries $q_1,\ldots,q_m$ as D, $d_i = d_i'$ $(D' = \{d_1',\ldots,d_n'\})$.

Our claim that C1 gave $60,000 is equivalent to the statement: Any data base with the same responses to the 12 queries of Table I must have D(1) = $60,000.

Our definition of a security problem has two important features. First, we allow that a user may know in advance parts of the data base. For example, suppose that C1's contribution is known in advance. Then two queries suffice to determine the contribution of C6 as

$$C6 = \text{steel} - \text{northeast} + C1.$$

Second, we allow that not all sequences of queries may be permitted. Suppose that a particular data base allows averages of size k and a user knows just one value. Then in just two queries he can compromise the data base. He asks

(1) What is the average of $x, y_1, \ldots, y_{k-1}$?

(2) What is the average of $x', y_1, \ldots, y_{k-1}$?

If he already knows x, he can determine x'. The reason the user was so successful is that he was allowed to ask two queries that *overlapped* greatly; his queries overlapped in k-1 elements. In the next section we will consider the problem of whether one can compromise such a data base if no two average queries can overlap very much.

III. Applications to a Particular Model: Averages

In this section, we assume that we are given a data base $\{x_1,\ldots,x_n\}$ of numbers and that queries may be made about the sum of any subset of the data base consisting of exactly k elements (this is equivalent to averages of k-element sets). We assume the further restriction that no two queries may overlap in more than r positions. And we assume that the values of $x_1,\ldots,x_\ell$ are known in advance by the user $(0 \le \ell < k-1)$. We then wish to study the behavior of the quantity $S(n,k,r,\ell)$, the smallest number of queries that suffice to compromise the data base. Compromising the data base will consist of generating the value of one previously unknown element, e.g. $x_{\ell+1}$.

Before proceeding, we present some sample values of the function $S(n,k,r,\ell)$.

Examples:

i) $S(n,3,2,0) \le 4 \qquad n \ge 4$

Let the queries be Q_1,Q_2,Q_3,Q_4 where

$$Q_1 = x_1 + x_2 + x_3$$
$$Q_2 = x_1 + x_2 + x_4$$
$$Q_3 = x_1 + x_2 + x_4$$
$$Q_4 = x_2 + x_3 + x_4.$$

Then x_4 can be found as $\frac{1}{3}(-2Q_1 + Q_2 + Q_3 + Q_4)$ and this is optimal.

ii) $S(n,4,1,1) \le 6$

Let the queries be $Q_1,\ldots,Q_6$ where

$$Q_1 = x_1 + x_3 + x_4 + x_5 \qquad Q_4 = x_2 + x_3 + x_6 + x_9$$
$$Q_2 = x_1 + x_6 + x_7 + x_8 \qquad Q_5 = x_2 + x_4 + x_7 + x_{10}$$
$$Q_3 = x_1 + x_9 + x_{10} + x_{11} \qquad Q_6 = x_2 + x_5 + x_8 + x_{11}.$$

Then $\frac{1}{3}\,[(Q_1 + Q_2 + Q_3) - (Q_4 + Q_5 + Q_6)] = x_1 - x_2$, which yields the value of x_2 since x_1 is known.

We begin our study of the properties of the function S by establishing a lower bound on its value.

Theorem 1: $S(n,k,r,\ell) \geq 1 + \frac{k-(\ell+1)}{r}$.

Proof: Suppose that after t queries we can determine the value of $x_{\ell+1}$, and let the queries be represented as

$$Q_i = \sum_{j=1}^{k} x_{i_j} \qquad i = 1,\ldots,t$$

for $1 \leq i_1 < i_2 < i_k \leq n$ where we assume the set $\{i_1,\ldots,i_k\} \cap \{j_1,\ldots,j_k\}$ has at most r members. This can then be represented as being able to satisfy the relation

$$\sum_{i=1}^{t} \alpha_i Q_i = \sum_{j=1}^{\ell+1} \beta_j x_j \qquad (*)$$

symbolically for $\beta_{\ell+1} \neq 0$. We proceed now by a counting argument, observing that the left-hand side of (*) can be rewritten as

$$\sum_{i=1}^{t} \alpha_i Q_i = \sum_{i=1}^{t} \alpha_i \sum_{j=1}^{k} x_{i_j} = \sum_{\sigma=1}^{n} \left(\sum_{i=1}^{t} \alpha_i x_{i_\sigma} \right) x_\sigma$$

where x_{i_σ} is 1 if x_σ belongs to query i and 0 otherwise. Thus t must be such that at most $\ell+1$ of the terms

$$\sum_{i=1}^{t} \alpha_i x_{i_\sigma} \qquad \sigma = 1,\ldots,n$$

are nonzero. In order for such a term to be zero, it must be the case that $x_{i_\sigma} = 0$, $i = 1,t$, or that i,j are distinct such that $x_{i_\sigma} = x_{j_\sigma} = 1$. Thus every x_σ that appears in some query must appear in at least two queries for $\sigma > \ell+1$. After the first query k x_i have been accessed, and $\frac{k-(\ell+1)}{r}$ more queries are required to access all the x_i at least twice. Hence $t \geq 1 + \frac{k-(\ell+1)}{r}$ is a lower bound for $S(n,k,r,\ell)$. □

As a dividend of the argument above, we observe that k-ir new variables of the data base are added in the p+1*th* query, $1 \leq p \leq \frac{k}{r}$, and thus the following corollary results.

Corollary: $S(n,k,r,\ell) = \infty$ if $n < \frac{k^2}{2r} - \frac{k}{2} + \frac{\ell+1}{2} - \frac{(\ell+1)^2}{2r}$.

Proof: This follows from the argument above since

$$k + \sum_{i=1}^{\frac{k-(\ell+1)}{r}} (k - ir) = \frac{k^2}{2r} - \frac{k}{2} + \frac{\ell+1}{2} - \frac{(\ell+1)^2}{2r}. \qquad \square$$

These results provide, then, a measure of the limitations of compromising a data base. We turn next to the question of actually implementing algorithms to perform these functions in order to get a sense of the tightness of these bounds.

Theorem 2:

i) $S(n,k,1,0) \le 2k-1 \qquad n \ge k^2-k+1$

ii) $S(n,k,1,1) \le 2k-2 \qquad n \ge (k-1)^2+2$

iii) $S(n,kp+d,p,2d-1) \le 2k \qquad n \ge k^2p+2\alpha$

iv) $S(n',kr,r,r-1) \le S(n,k,1,0) \qquad n' \ge rk^2.$

Proof: In each case, our proof consists of an algorithm that performs the given task within the desired bound.

i) Let the queries be

$$Q_i = \sum_{j=1}^{k} x_{k(i-1)+j} \qquad i = 1,\ldots,k-1$$

$$Q_{k+i-1} = \sum_{j=1}^{k-1} x_{k(j-1)+i} + x_{k^2-k+1} \qquad i = 1,\ldots,k.$$

Then

$$\frac{\sum_{i=0}^{k-1} Q_{k+i} - \sum_{i=1}^{k-1} Q_i}{k} = x_{k^2-k+1}.$$

ii) Let the queries be

$$Q_i = x_1 + \sum_{j=2}^{k} x_{(i+1)(k-1)+j} \qquad i = 1,\ldots,k-1$$

$$Q_{k-1+i} = \sum_{j=1}^{k-1} x_{1+i+(j-1)(k-1)} + x_{(k-1)^2+2} \qquad i = 1,\ldots,k-1.$$

Then

$$\frac{\sum_{i=1}^{k-1} (Q_i - Q_{k-1+i})}{k-1} = x_1 - x_{(k-1)^2+2}$$

and $x_{(k-1)^2+2}$ may be determined since x_1 is known in advance.

iii) Let the queries be

$$Q_i = \sum_{j=1}^{kr} x_{kr(i-1)+j} + \sum_{\ell=1}^{\alpha} x_{k^2r+\ell} \qquad i = 1,\ldots,k$$

$$Q_{k+1} = \sum_{j=1}^{k} \sum_{\ell=1}^{r} x_{kr(j-1)+i-1)r+\ell} + \sum_{m=1}^{\alpha} x_{k^2r+\alpha+m} \qquad i = 1,\ldots,k.$$

Then

$$\sum_{i=1}^{k} (Q_i - Q_{k+1}) = k \sum_{\ell=1}^{\alpha} x_{k^2r+\ell} - \sum_{m=1}^{\alpha} x_{k^2r+2+m}$$

and x_{k^2r+1} can be determined if the values of

$x_{k^2r+2},\ldots,x_{k^2r+\alpha},x_{k^2r+\alpha+1},\ldots,x_{k^2r+2\alpha}$ are known in advance.

iv) This statement follows by letting $y_i = x_{(i-1)r+1} + \ldots + x_{ir}$, using an algorithm for S(n,k,1,0) to find y_1, and using known values of $x_1,\ldots,x_{r-1}$ and y_1 to find x_1.

The reason we are interested in the results of this section is that we feel they are hard evidence of how complex it will be to secure a data base. The complex and obscure techniques used in Theorem 2 to crack the data base demonstrate how difficult it will be to determine whether a series of simple queries can compromise a data base. On the other hand, Theorem 1 points out that there do exist mechanisms (bounding the overlap and the number of queries) that can protect a data base.

IV. Applications to a Particular Model: Medians

In this section we assume that we are given a data base $\{x_1,\ldots,x_n\}$ of distinct numbers and that queries can be made about the median of k elements for some fixed odd k. A median query of $y_1,\ldots,y_k$ returns the median's value but *not* which y_i is equal to this value. We also assume that no values are known in advance. We then wish to study the quantity $M(n,k)$, the smallest number of queries that suffices to determine some element of the data base perfectly. Our main result is

Theorem 3: $M(n,k) \le \frac{3}{2}(k+1) + 1$ provided $n \ge k+2$.

This result demonstrates clearly that even the simple operation of median can be used to compromise a data base; indeed, this can be done in very few queries.

Proof: Let $p = \frac{k+1}{2}$. Also let $x_1,\ldots,x_{k+2}$ be k+2 objects of the data base. First perform all possible k medians involving the objects $x_1,\ldots,x_{k+1}$. Clearly there are $\binom{k+1}{k}$ such medians. These medians result in exactly two answers, say s and ℓ with $s < \ell$. Let

$$S = \{x_i \mid x_i \le s\}$$

and

$$L = \{x_i \mid x_i \ge \ell\}.$$

Then $x_i \in S$ iff the median of $\{x_1,\ldots,x_{k+1}\} - \{x_i\}$ is ℓ; $x_i \in L$ iff the median is s. An easy argument shows that $|S| = |L| = p$.

We now form the median of $S' \cup L \cup \{x_{k+2}\}$ where

$$S' = S - \text{any two elements of } S.$$

There are three cases:

1) The answer = ℓ. Then $x_{k+2} < \ell$.

2) The answer > ℓ. Then $x_{k+2} > \ell$.

3) The answer < ℓ. This is impossible.

Thus this query determines whether x_{k+2} > or < ℓ. Without loss of generality assume that $x_{k+2} < \ell$.

We now fix a set of p-1 elements of L and call if L_0. Let us finally examine all the medians of the set

$$S \cup \{x_{k+2}\} \cup L_0 - \{x_i\}$$

where $1 \le i \le p$ or $i = k+2$. We now claim that one median (say m) occurs p times and one median occurs once. This follows by a simple argument. Then $m = x_i$ where x_i is not in the set when m does not occur. □

V. Conclusions

A precise model of the security problem for data bases has been presented. In this model we were able to demonstrate how to control the queries a user could make in order to stop him from compromising the data base. While we did this only for queries about averages and medians, we can extend the model to handle queries of other types. This model gives rise to a number of interesting combinatorial problems which have applications to problems of applicability to designers or data bases. While we have presented an introduction to problems in this area here, a number of related problems remain open. For example, suppose we change our constraints on overlapping queries to allow queries to overlap only in certain co-ordinates. Or, suppose we allow queries of varying lengths. Or, suppose we may ask for medians but wish to determine a specific data base entry. Furthermore, in each case studied here, we have considered worst case behavior, the number of queries necessary to guarantee that the data base is compromised. We could do a similar analysis for best case behavior, asking for the fewest queries after which the data base may be compromised. Many of these issues will be studied in a forthcoming paper [2].

Acknowledgement

We wish to thank Rich Demillo, Mary-Claire van Leunen, and Steven Reiss for a number of helpful comments on an earlier draft.

References

1] E. F. Codd.
A relational model of data for large and shared data banks.
CACM 13(6):377-387, 1970.

2] R. Demillo, D. Dobkin and R. Lipton.
In preparation.

3] J. S. Fenton.
Memoryless subsystems.
Computer Journal 17(2):143-147, 1974.

4] G. S. Graham and D. J. Denning
Protection -- principles and practice.
AFIPS Conference Proceedings 40:417-429, 1972.

5] M. Haq.
Insuring individuals' privacy from statistical data base users.
AFIPS Conference Proceedings 43:941-946, 1975.

6] L. J. Hoffman and W. F. Miller.
Getting a personal dossier from a statistical data bank.
Datamation 74-75, May 1970.

7] A. K. Jones and R. J. Lipton.
The enforecment of security policies for computation.
Proceedings of the 5th Symposium on Operating System Principles, 197-206. 1975.

8] A. K. Jones and W. A. Wulf.
Towards the design of secure system software practices and experiences.
To appear.

9] B. W. Lampson.
Proceedings of the 5th Private Symposium on Information Sciences and Systems, 437-443. 1971.

10] E. I. Organick.
The MULTICS System: An Examination of Its Structure.
MIT Press, 1972.

11] W. A. Wulf et al.
HYDRA: The kernel of a multiprocessor system.
CACM 17(6), 1974.