RANDOMIZING, A PRACTICAL METHOD FOR PROTECTING STATISTICAL DATABASES AGAINST COMPROMISE

Ernst Leiss

Department of Computer Science, University of Houston
Houston Texas 77004

Abstract

This paper reports on a method for protecting statistical databases against inference, developed over the past four years. The method, called randomizing, is conceptually elegant, easy to implement at very small additional cost, and requires no on-going maintenance. It can be equally applied to small and medium-size as well as large databases, because it does not depend on sampling techniques in the conventional sense. Thus we feel that randomizing is an eminently practical and effective method if the protection of statistical data against inference is of concern.

1. Introduction

In the past decade, the importance of databases to the operation of large and medium-size enterprises has gradually been realized. In parallel with this development, a significant increase in sophistication with regard to database design and database implementation has occurred. In particular, a number of questions arose out of practical consideration which previously were considered purely academic. Many of these questions are related either to concurrency of operations or to security of data.

The term data security encompasses a very large area, from the physical protection of tape reels to the most sophisticated kernel design methodologies. In this paper we are primarily interested in statistical databases. These are databases containing numerical data where access to individual entries is not permitted (except to users with special authorization), but statistics concerning several data items are supplied to the users of this database. As an example, consider a database containing information about the state of health of top government officials. While there may be a legitimate interest of the general public in, say, the number of psychiatric treatments administered to members of the government with cabinet rank during the past five years, it might have severe implications if an official could be identified as one of the recipients (cf. Sen. Eagleton). Thus security of statistical databases is concerned with the problem whether it is possible to infer from responses to legitimate queries information which is explicitly hidden from the general user. In other words, the important issue in statistical database security is inference control.

2. The Model

Throughout this paper we will assume a relational database (see [1] or [23], e. g.) consisting of keys (which may be a combination of columns, called category fields) used to access the items or records; the (numerical) values of the database items are assumed to be confidential. The assumption of a relational database is for technical reasons only. In our particular database model, access to items is by specifying the item's index (an integer denoting its position in the database). The index is to be considered the primary key in this setting. This model is called key-specified; it is an abstraction of the usual model where access is by way of characteristic formulae (see [14]). For both access mechanisms, a query is a request to supply the information associated with the items specified in the query.

In a statistical database all the items are considered confidential (secret); thus they are to be protected against unauthorized access. However in many cases (census information, e. g.) there is a legitimate interest in obtaining statistical access to these data (e. g. "average income of all fathers of six or more children"). Thus a user may not be authorized to access individual items but s/he may have access to the average or median of a number of items. These queries are called queries of type average or median.

We say a database is compromised if a user can determine the value of any item (which was previously unknown) from the responses to legal queries. A database is called secure if it can not be compromised.

3. Randomizing

The literature is replete with results demonstrating the difficulty of obtaining secure databases with the traditional approaches (see for instance [2,...,11,15,...,22]; for a general overview see [14]). However, clearly there is a need for secure databases. In the following we will outline just that: A database which is almost identical to the conventional statistical database and yet it can be shown to be secure. Moreover the method is conceptually very simple, it can be superimposed on practically any existing database, and the additional cost is quite negligible in most cases (see Section 7.). We will describe our approach exclusively in terms of queries of type average, although it works also with queries of other types. We propose a method using random selection. Rather than using queries of type average of k elements we employ the following type of queries where $v \geq 0$: The user provides (as usual) k indices $i_1,\ldots,i_k$ and the system determines the corresponding database elements $DK(i_1),\ldots,DK(i_k)$; but instead of computing the average of these k values the system first determines randomly another v elements of the database and then computes the average of these k+v elements. Clearly for v=0 we obtain the original queries of type average. For $v \geq 1$, this method will be called randomizing and the queries will be referred to as randomized queries of type average. Thus if q is the response to the randomized query of type average $(i_1,\ldots,i_k)$, then we have

$$q = \left(\sum_{j=1}^{k} DK(i_j) + \sum_{j=1}^{v} DK(s_j)\right)/(k+v)$$

where $DK(s_j)$ is a result of the random selector function S and DK(m) is the value of the database element selected by index m. In the

following we will concentrate mainly on the case where v=1. However all the results for v=1 can also be applied to the case where v>1. Note that for k not too small, the response of such a query will be a good approximation on the (desired) precise value (average of $\{DK(i_1),\ldots,DK(i_k)\}$). The main claim of the paper is now as follows:

Theorem: Let v=1, k not too small (e.g. k>5), and assume that no database items are known to the compromiser. It is impossible to compromise a database with randomized queries of type average.

Proof: The proof consists of two parts. First we assume that our way to compromise a database is by solving a system of linear equations obtained directly from a sequence of queries. We claim that in this case our database cannot be compromised. We first observe that our best "guess" for the values $DK(s_j)$ determined by the random selector function S is the average s' taken over all responses to the queries issued. Therefore rather than solving a system of equations

$$D*x = k*q$$

where D is the matrix derived from our given sequence of queries we must solve the modified system

$$D*x = q'$$

where

$$q' = ((k+1)q_1-s_1,\ldots,(k+1)q_t-s_t).$$

However all what we can solve is the system

$$D*x = q''$$

where

$$q'' = ((k+1)q_1-s',\ldots,(k+1)q_t-s');$$

in other words q' contains errors and is given by q". The sensitivity of a system of linear equations M*x=c to errors in c is usually measured by the condition number

$$cond(M) = ||M||\cdot||M'||$$

where $||\ ||$ is some matrix norm and M' denotes the inverse matrix to M. We choose the following norm $||\ ||$:

$$||M|| = \left(\sum_{i,j=1}^{n} |m(i,j)|^2\right)^{1/2}, \quad M=(m(i,j))_{1\le i,j\le n}$$

It can now be shown (see for instance [14]) that for any matrix D involved in possible compromise,

$$cond(D) \ge t.$$

This implies that all matrices D are quite sensitive to errors; note that t>k. Furthermore on the average the errors can be expected to be considerable; if we assume that the s_j are distributed uniformly, s_j-s' will be of the same order of magnitude as s' on the average. Therefore it is not possible to obtain useful results when solving this system of equations; the database can not be compromised.

In the second part of the proof we show that the method of "filtering out" the "noise" introduced by the "errors" does not achieve compromise either. Let

$$q_j : (\ i(j,1),\ldots,\ i(j,k)\ ) \text{ for } j=1,\ldots,t$$

be a sequence of queries of type randomized average corresponding to a matrix D of dimension t such that the following holds: if q_j' is q_j as a query of type (nonrandomized) average then by solving D*x=k*q' we can determine all $DK(i_1),\ldots,DK(i_t)$. Now define q(j,m) to be the response to the mth repetition of query q_j; note that in general $q(j,m)\neq q(j,n)$ since the result s_m of the random selector function S in the mth repetition will usually differ from its result s_n in the nth repetition although otherwise the queries q(j,m) and q(j,n) are identical. However

$$(q(j,1)+\ldots+q(j,N'))/N'$$

might converge to a certain value with increasing N', say to q_j'; thus we would get

$$q_j' = \sum_{n=1}^{t} d(j,n)\ DK(i_n) + s_j'$$

where D=(d(j,n)). Assuming that the s_j' are not all equal, it is not difficult to see that the database cannot be compromised. Of course we assume that no s_j' is known; note that this information cannot be retrieved from the queries if one uses the following scheme. Let T be a uniform random selector function; whenever a query of type randomized average is posed, two calls to T are made yielding t_1 and t_2. We determine which one of these two values is to be returned as value for S in such a way that it "depends on all $DK(i_j)$ for j=1,...,k in the query", i.e. if $(i_1,\ldots,i_k)$ is the sequence of indices specified by the user in the query and the choice of S based on this sequence is $\max\{t_1,t_2\}$ (is $\min\{t_1,t_2\}$) then for all i_j, j=1,...,k, there exists an index g_j in {1,...,N} such that the choice of S based on the new sequence of indices

$$(i_1,\ldots,i_{j-1},g_j,i_{j+1},\ldots,i_k)$$

is $\min\{t_1,t_2\}$ (is $\max\{t_1,t_2\}$). A concrete scheme to achieve this is the following. Let E be the boolean expression

$$E=[(DK(i_1)\leq DK(i_2))\oplus\ldots\oplus(D<(i_{k-1})\leq DK(i_k))]$$

where $\oplus$ is the exclusive-or operator defined by

$\oplus$	true	false
true	false	true
false	true	false

Then S yields the value $\max\{t_1,t_2\}$ iff E is true. This scheme satisfies the above condition and has the additional advantage that on the average $\max\{t_1,t_2\}$ and $\min\{t_1,t_2\}$ are returned equally often. Note that in this way no information about the range of the DK(s) is required, no time dependency is introduced, and no previous values of T must be stored.

We remark that other schemes for compromise, in particular trackers ([19]; also [7,6]) can not be applied within our framework.

4. Randomizing: Simulation Results

In order to provide pratical results, several extensive simulations were performed. In all simulations described in this paper, the database elements as well as the randomly selected elements were obtained through successive calls to a uniform random number generator (URAND). The first simulation ([12]) was directed at the accuracy of the responses to the queries. It turned out that for small k (k=5 and k=10), the accuracy of the responses was acceptable on the average but the worst case was quite bad; on the other hand, for larger k ($k\geq 20$) both the average as well as the worst case are very satisfactory (see Fig. 1).
The second simulation ([13]) demonstrates that a good deal of care must be exercised in the choice of the random elements. More specifically, the simulation contrasts the security of randomized databases where the choice of the random element is as described above (using the logical formula E) with the simple minded approach where the random element is selected in a uniformly distributed way. The implication of this simulation is that it is necessary to use the more complicated selection method if one is interested in security (see Fig. 2).
This last simulation also shows that "filtering out" the error in the responses by repeating the same query many times does not achieve compromise. More specifically, the simulation determines the probability with which a computed (compromised) value of a database element is afflicted with an error of a given magnitude. It shows that even for 1000 repetitions of the same query the probabilities of large errors are very substantial, if the sophisticated selection method is chosen.

k	Average relative error in percent	Maximal
5	8.9	140
10	4.7	30
20	2.4	11
50	1.0	3.5
100	0.5	1.4

Fig. 1 Accuracy of the responses to queries (No elements assumed known)

	repetitions			repetitions	
k	1	1000	k	1	1000
5	24.05	38.28	5	24.34	74.56
10	20.11	26.75	10	19.32	82.73
20	17.48	15.86	20	17.78	88.87
50	16.92	7.99	50	17.03	93.76
100	16.25	4.67	100	16.27	95.07
sophisticated method			simple method		

Fig. 2 Probability in percent that the error of a computed database element is less than 16 percent (No elements assumed known)

5. Accuracy and Restricted Randomizing

It is evident from the data in Figure 1 that for smaller k, say k<20, the accuracy of the responses can be quite bad. In fact, it is not difficult to see that the relative error introduced by randomizing a query can be arbitrarily bad. The question arises whether this can be excluded. It should be noted that it is the maximal error of the responses which is rather unpleasant; the average error is quite acceptable even for smaller k. This suggests the notion of restricted randomizing.

Consider a query $(i_1,\ldots,i_k)$ (of type average); let q be its (true) response, i.e.

$$q = [DK(i_1) + \ldots + DK(i_k)] / k.$$

In our randomized model the response will not be q but q', defined by

$$q' = [DK(i_1) + \ldots + DK(i_k) + DK(s)] / (k+1)$$

where s is the result of a call to the random selector function (assuming v=1). As pointed out, q' may differ arbitrarily from q. Let mx (mn) be the largest (smallest) of the elements $DK(i_j)$, j=1,...,k. Instead of allowing DK(s) to be arbitrary we require it to satisfy the following inequalities:

$$q - (mx+mn) / (2j) \leq DK(s) \leq q + (mx+mn) / (2j)$$

for some j>0 suitably chosen. This method of selecting the randomly chosen element is then called restricted randomizing. Clearly the crucial point is the choice of j. If j is too small (j<<1) then for all practical purposes we will end up with unrestricted randomizing; if j is too large the contribution of DK(s) will not change q at all thereby rendering the method useless as now the database can be compromised. Furthermore if j is too large, it is possible that no DK(s) satisfies the required inequalities. In these and similar cases (mx=mn, e.g.), alternative schemes must be provided. The following method was found to be useful. Let S be the sophisticated random selector function described above. Given a query $(i_1,\ldots,i_k)$ we make a call to S; let the result of this call be x. Then we test whether DK(x) satisfies the inequalities; if yes then we use this index x in the computation of q' otherwise we continue calling S (regardless of the previous seed!) until either the result does satisfy the requirements or else until a certain preset number of successive calls to S has been made (e.g. 20j) in which case the value to be used in the computation of q' is that which came closest to satisfying the conditions.

A note-worthy by-product of our method is an intriguing security-accuracy trade-off. It can be briefly stated as follows. Increasing the security of the database can be done at a price in accuracy of the responses, and conversely increasing the accuracy of the responses results in a decrease of the security of the database. There are two ways in which the accuracy of the responses to queries can be influenced, namely the choice of v and the choice of j. Choosing v greater than 1 will diminish the precision of the responses (as clearly the contribution of

the randomly chosen elements will be greater); a similar effect is obtained if j, the randomizing parameter, is chosen very small (since in this case the randomly selected elements do not depend at all on the size of the other elements in the query). Selecting a very large j will open the database to easy attacks, it will not be secure any longer.

Clearly, the choice of v and j is to be made by the owner of the database; in fact, it may even be desirable to choose different values for v and j for different users (e.g. depending on their authorization). This security-accuracy trade-off is very appropriate as it reflects reality in that not all data are to be considered equally confidential and that not all users are equally trusted. Thus a judicious choice of values for v and j permits database owners to reflect different degrees of confidentiality of their data with regards to different audiences.

6. Restricted Randomizing: Simulation Results

In order to obtain data concerning the accuracy of the responses another simulation was done exactly like the first one (whose results are reported in Figure 1) but implementing the above described method of restricted randomizing. The results for k=5,10,20 are given below in Figure 3 for various values of j, namely j=1,2,5,10,20,50.

Then we conducted another experiment by way of simulation in order to establish that this class of methods indeed results in secure databases for sensible choices of j. This simulation is compatible with all the others. In order to determine how secure the resulting database is, we determined the probability that the computed value of any (compromised) database element is afflicted with an error of a certain magnitude. Method and results of this very extensive simulation are discussed at length in [13]; here we give only part of the tables obtained. Figure 4 gives the probability that the computed value of any database element has an error less than a certain threshold, here 16% and 4%. More specifically, if we are using queries with k indices and we attempt to compromise, i.e. compute the value of a particular database element, the expected or probable error which this computed value will have (owing to the fact that the queries are randomized) can be derived from Figure 4. For example, if we know that the randomizing factor j is 10 and that k is 100, then the probability that the computed value for the element is within four percent of the true value of the database item is 21.06% if no filtering was used, and it is 19.64% if 1000 repetitions were used.

The results suggest that restricted randomizing for a suitable choice of j is a very useful way of protecting confidential data on the one hand while on the other hand providing meaningful statistical information based on these confidential database items.

k	j	average relative error in percent	maximal relative error in percent
5	1	8.2	65
	2	5.8	54
	5	2.8	28.5
	10	2.1	21.3
	20	1.9	23.4
	50	1.8	17.1
10	1	4.7	25.9
	2	3.9	19.4
	5	1.8	14.0
	10	1.2	12.0
	20	1.04	12.9
	50	0.99	10.4
20	1	2.4	11.2
	2	2.2	10.9
	5	0.93	4.6
	10	0.6	3.9
	20	0.47	3.8
	50	0.43	3.4

Figure 3: Accuracy with restricted randomizing

	repetitions		repetitions	
k	1	1000	1	1000
j=2				
5	36.22	48.29	9.93	18.33
10	24.46	35.61	6.51	5.45
20	19.54	22.46	4.93	0.84
50	17.39	10.96	4.31	0.00
100	16.65	5.98	4.14	0.00
j=5				
5	63.37	74.80	21.62	28.53
10	55.71	75.15	15.40	24.05
20	48.57	68.35	12.44	7.92
50	43.61	62.42	11.09	0.78
100	41.98	60.27	10.49	0.08
j=10				
5	75.81	74.94	27.69	24.94
10	77.03	81.90	29.50	39.81
20	74.91	85.45	25.17	32.11
50	71.42	81.11	21.85	21.29
100	70.31	80.29	21.06	19.64
j=20				
5	75.15	75.38	25.84	25.73
10	82.51	81.86	43.44	49.19
20	85.74	88.74	45.86	68.93
50	85.52	91.02	43.21	65.44
100	84.78	90.09	41.76	60.06

Probability in percent that the error of a computed database element is less than 16 percent (left two columns of the table) and less than 4 percent (right two columns of table) for various values of j and k; no elements assumed known.

Figure 4

7. Implementation Considerations

The description of our method is in terms of the conceptual level of a database. Clearly, the actual implementation of the method will depend to a significant extent on the kind of database which is used. If a relational database implementation is employed, the description given here can be directly carried over, provided it is possible to talk about the ith tuple in a table. If the database model employed in the implementation is either the hierarchical or the network model, the situation changes somewhat since it does not make sense any longer to talk about the ith record (unless the records are numbered from 1 to N contiguously -- not a very realistic assumption in view of insertion and deletion of records). In both cases, the random selection can however be performed by randomly selecting access paths. The most straight-forward (albeit possibly not always most efficient) way is probably to choose randomly at any given node in the tree or graph, which edge leaving this node is to be taken, until some termination criterion is met. A good deal of care must be exercised in the selection of the starting point for these random access paths. In the hierarchical model, the access paths will start at the root of the appropriate tree, in the network model one must probably have several starting points to ensure randomness of the resulting choice of element. The problem in the network model is that there may be several paths leading to the same record (this is in contrast to the hierarchical model where there is precisely one path starting at the root to any particular node), and moreover, different records may have vastly different "accessibility". For example, for one record there may be a single access path while for another record there are twenty different access paths; consequently without additional precautions the second record is much more likely to be "randomly" chosen in this way than the first. This problem can be overcome with several starting points for the randomly chosen access paths.

References

[1] DATE, C.J. An Introduction to Data Base Systems
Addison-Wesley Reading, Mass., 1975

[2] DAVIDA, G.I., LINTON, D.J., SZELAG, C. R., WELLS, D.L. Database Security
IEEE Transactions on Software Engineering, Vol. SE-4, No. 6, November 1978, 531-533

[3] DEMILLO, R.A., DOBKIN, D., LIPTON, R.J. Even Data Bases That Lie Can Be Compromised
IEEE Transactions on Software Engineering, Vol. SE-4, No. 1, January 1978, 73-75

[4] DENNING, D.E. Are Statistical Data Bases Secure?
Paper presented at the NCC in Annaheim, June 1978

[5] DENNING, D.E. A Review of Research on Statistical Data Base Security
In [24], 15-25

[6] DENNING, D.E., DENNING, P.J., SCHWARTZ, M.D. The Tracker: A Threat to Statistical Database Security
ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979, 76-96

[7] DENNING, D.E., SCHLÖRER, J. A Fast Procedure for Finding a Tracker in a Statistical Database
ACM ToDS 5,1 (March 1980), 88-102.

[8] DOBKIN, D., JONES, A.K., LIPTON, R.J. Secure Databases: Protection Against User Inference
ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979, 97-106

[9] DOBKIN, D., LIPTON, R.J., REISS, S.P. Aspects of the Database Security Problem
Proceedings of a Conference on Theoretical Computer Science, August 15-17, 1977, University of Waterloo, Waterloo, Ont.

[10] KAM, J.B., ULLMAN, J.D. A Model of Statistical Databases and Their Security
ACM Transactions on Database Systems, Vol. 2, No. 1, March 1977, 1-10

[11] LEISS, E. Security in Databases where Queries Involve Averages
Research Report CS-77-33, Department of Computer Science, University of Waterloo, Waterloo, Ontario, October 1977

[12] LEISS, E. Database Security and Restricted Randomizing
Proceedings, Primera Conferencia Nacional en Teoria de Computación y Desarrollo de Software, Santiago, Chile, August 22-24, 1979

[13] LEISS, E. On the Security of Randomized Databases: A Simulation
Proceedings, First International Conference on Computer Science, Santiago, Chile, 1981, pp. 135-159.

[14] LEISS, E. Principles of Data Security
Plenum Publishing Corporation, New York, New York, in press

[15] REISS, S.P. Medians and Database Security
In [24], 57-91

[16] REISS, S.P. Security in Databases: A Combinatorial Study
JACM Vol. 26, No. 1, January 1979, 45-57

[17] SCHLÖRER, J. Identification and Retrieval of Personal Records from a Statistical Data Bank
Methods of Inform. in Medicine, Vol. 14, No. 1, January 1975, 7-13

[18] SCHLÖRER, J. Confidentiality of Statistical Records: A Threat Minitoring Scheme for On-Line Dialogue
Methods of Inform. in Medicine, Vol. 15, No. 1, January 1976, 36-42

[19] SCHLÖRER, J. Union Tracker and Open Statistical Databases
Report TB-IMSD 1/78, Institut für Medizinische Statistik und Dokumentation, Universität Giessen, June 1978

[20] SCHWARTZ, M.D. Inference from Statistical Data Bases
Ph.D. Thesis, Department of Computer Science, Purdue University, W. Lafayette, Ind., August 1977

[21] SCHWARTZ, M.D., DENNING, D.E., DENNING, P.J. Linear Queries in Statistical Data Bases
ToDS Vol. 4, No. 2, June 1979, 156-167

[22] YAO, A.C. A Note on a Conjecture of Kam and Ullman Concerning Statistical Databases
Information Processing Letters Vol. 9, No. 1, July 1979, 48-50

[23] WIEDERHOLD, G. Database Design
McGraw-Hill, New York, New York, 1977

[24] DEMILLO, R.D., DOBKIN, D., JONES, A.K., LIPTON, R.J. (eds.) Foundations of Secure Computation
Academic Press, New York, 1978