A NEW DETERRENT TO COMPROMISE OF CONFIDENTIAL INFORMATION FROM STATISTICAL DATABASES

by

NANDA KAUSHIK

B.Tech., Indian Institute of Technology, New Delhi, INDIA, 1981

M.Sc., Deakin University, Waurn Ponds, AUSTRALIA, 1986

M.S., Kansas State University, Manhattan, KS, 1986

Ph.D., Kansas State University, Manhattan, KS, 1988

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

Kansas State University

Manhattan, Kansas

1988

Approved by:

Major Advisor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to express my gratitude and appreciation to Professor Elizabeth A. Unger and Professor Sallie K. McNulty for their guidance, support, and valuable advice during the course of this study.

I would like to thank Professor Virgil E. Wallentine for being on my examination committee and for encouraging me during my stay at Kansas State University.

Finally, I wish to thank my wife, Mamta, and my parents for their patience during my graduate study.

Chapter I

INTRODUCTION

Computers are used widely for the storage of large volumes of data. Most data stored is not in the public domain as it contains either vital business/governmental/military information or confidential information about individuals. The violation of privacy of an individual is "making such information available to others ... without his or her consent ..." [FELL72]. Consequently, the issue of data security is of great concern to the owners of databases and has been receiving great attention from researchers over the last two decades. According to Miranda [MIRA80], data security is of greater importance in database management systems than in any other software because it can be changed and also because data access is made available to many via powerful and convenient user interfaces.

Two kinds of security control can be imposed. One is external security control in which personnel and physical access to the computer is limited. The second kind of control is internal security control. It is in this type of control that abuse by authorized sophisticated computer users must be restricted or denied.

Internal security mechanisms were surveyed by Denning and Denning [DENN79b]. They discuss four areas of internal security control: access controls, flow controls, inference controls and cryptographic controls. All these controls involve regulating the operations of a computer system. Access controls regulate modification of data and programs. Flow controls regulate flow of information from one object

to another. Inference controls regulate the inference of confidential information from statistical databases. Cryptographic controls regulate the encryption of the data stored in a computer system or those transmitted on communication lines. Considerable research has and is being done in each of the above areas.

Although the above security controls are all equally important and interesting, the focus of this work is in the area of inference control in statistical databases. Statistical databases provide statistical information such as frequency counts, means, medians, sums, ..., etc, of a certain subset of data in a given population. The objective is that no confidential information about a particular individual is revealed but good statistical population measures can be obtained. However, users can often deduce (or infer) confidential information from the statistical information. This type of compromise is very difficult to control because the database is compromised by legitimate queries. Such compromise is effected through the use of trackers, logical formulas and a process of manipulating data obtained from a few overlapping subsets of records.

The exact nature of this form of compromise, the previous research done in the areas of avoiding this form of compromise and to making such compromise very difficult is discussed in Chapter II. It must be noted however that all the methods proposed to date either have flaws in that the database is still susceptible to compromise or are very expensive to implement. The present research is an attempt to find a relatively inexpensive solution to the problem of preventing

the compromise of an individual's privacy via inferential methods from a statistical database.

Chapter II

LITERATURE SURVEY

INTRODUCTION

Any survey on databases must begin with the specification of the database model. The terms in this work are defined and examples are provided to clarify these definitions. The database in Denning, Denning and Schwartz [DENN79a] was chosen as a basis for most of the examples in this study. This database which contains information about employees in a hypothetical university's College of Mathematical Sciences is shown in Table 2.1.

Table 2.1. Database Containing Information On Employees in a Hypothetical University's College of Mathematical Sciences.

No.	Name	Sex	Dept	Position	Salary	Political Contribution
1	Adams	M	CS	Prof	20	50
2	Baker	M	Math	Prof	15	100
3	Cook	F	Math	Prof	25	200
4	Dodd	F	CS	Prof	15	50
5	Engel	M	Stat	Prof	18	0
6	Flynn	F	Stat	Prof	22	150
7	Grady	M	CS	Adm	10	20
8	Hayes	M	Math	Prof	18	500
9	Irons	F	CS	Stu	3	10
10	Jones	M	Stat	Adm	20	15
11	Knapp	F	Math	Prof	25	100
12	Lord	M	CS	Stu	3	0

DATABASE MODEL

A statistical database is a collection of records for each entity. An entity is a "thing that exists and is distinguishable" [ULLM82]. In the database of Table 2.1, each correspondent (or individual) is an entity. The database contains information about 12 individuals (sometimes referred to as the size of the database). Entities have properties called attributes. In our example, Name, Sex, Dept, Position, Salary and Political contributions are all attributes. Each attribute has many possible values in its domain. The possible values for each of the attributes (or the domains) are:

Name	: a character string
Sex	: M, F
Dept	: CS, Math, Stat
Position	: Prof, Adm, Stu
Salary	: any integer $\geq$ 0
Political contribution	: any integer $\geq$ 0

An attribute or a set of attributes whose values uniquely identify each entity is called a key. In the above example, the attribute "Name" is a key. Information about any particular individual is considered confidential and consequently keys are not considered to be part of statistical databases. However, an individual's record can possibly be identified by another group of attributes. For example, Dodd could be specified by having the values F, CS and Prof for the attributes Sex, Dept and Position respectively.

We can also view the database records to contain category and data fields [DENN78]. In the above example, the attributes Sex, Dept and Position may be considered as category fields (the values do not represent numerical data). The attributes Salary and Political contribution are data fields (the values are numerical data).

A query is a question which can be asked about a database. Queries could be of two forms: key specified and characteristic specified. Key specified queries request statistics for a set of individuals identified by keys. It would be useful, at this point, to present the model proposed by Kam and Ullman [KAMU77] and Chin [CHIN78]. They view a statistical database as a function f from strings of k bits (called the key) to integers. In Chin's model, the range of f is an ordered pair {0,1} x R. 0 indicates that there are no records with the specified keys in the database, and a 1 indicates that there are records with the specified keys in the database. The value/result of the query is a real number (R). The database about employees in the hypothetical university above could be represented by keys consisting of 25 bits abbccddddddeeeeeeeeeeeee interpretted as:

(1) a is the Sex of the individual; 0=M, 1=F

(2) bb is the Dept; 00=CS, 01= Math, 10=Stat

(3) cc is the Position; 00=Prof, 01=Adm, 10=Stu

(4) dddddd is the Salary

(5) eeeeeeeeeeeee is the Political contribution

The database is queried by specifying some of the bits and leaving the others unspecified. An unspecified bit is denoted by a *. For

example, if the only queries allowed are the queries on salaries, the sum of the salaries of all males in the CS department could be obtained from the query:

000**********

To find the sum of the salary of all female Professors with a contribution of $100, the query would be:

1**00******00000001100100

It is easily seen that this is very cumbersome and consequently not very popular. Therefore, the model considered in this work deals with characteristic specified queries. However, it must be mentioned that using the above model, Kam and Ullman [KAMU77] and Chin [CHIN78] guarantee that the database is secure (definition of which appears later). For a database of this nature, one can only ask queries involving either the operators SUM or COUNT.

Characteristic specified queries, q(C), uses a characteristic formula C to group records in a database. A characteristic formula is a logical formula over the values of category fields. This logical formula uses boolean operators: and (&), or (+) and not ($\bar{}$). The operands are values of category fields. For example:

C = (Sex=M) & (Dept=CS)

is a characteristic formula which specifies all males in the CS department. The set of records which satisfy a characteristic formula C is called a query set, X_C. The size of the query set is denoted by $|X_C|$. Records corresponding to Adams, Grady and Lord would satisfy

the characteristic formula C = ((Sex=M)&(Dept=CS)) and hence would be members of the query set X_C of size three. To give another example, Dodd is the only member of the query set X_C (of size one) for the characteristic formula:

C = (Sex=F) & (Dept=CS) & (Position=Prof)

The characteristic specified queries q(C) can take many forms [DENN78]. Some of the forms are:

COUNT(C) = $|X_C|$, where $|X_C|$ is the size of the query set X_C satisfying the characteristic formula C. (1)

SUM(C;j) = $\sum_{i \epsilon X_C} v_{ij}$, where v_{ij} is a data field j of record i. (2)

= Sum of all the values in the data field j for all individuals satisfying C.

select(C;j) = $\underset{i \epsilon X_C}{\text{select}}\ v_{ij}$ where select is MEDIAN, SMALLEST, LARGEST, MEAN etc. (3)

= MEDIAN, SMALLEST, LARGEST or MEAN of the data fields j for all individuals satisfying C.

The COUNT and SUM queries can be written in a more general from [DENN79a] as:

$$q(C;j,m) = \sum_{i \epsilon X_C} v_{ij}^m \qquad (4)$$

where

m = 0 for the COUNT query,

and m = 1 for the SUM query.

Examples of some characteristic specified queries are:

COUNT((Sex=M) & (Dept=CS))) = 3

= Number of males in the CS department.

SUM ((Sex=M) & (Dept=CS); Salary) = 33K

= sum of the salaries of all males in the CS department.

MEDIAN ((Sex=M) & (Dept=CS); Salary) = 10K

= median of the salaries of all males in the CS department.

SUM ((Sex=M) & (Dept=CS); Political Contribution) = $70

= sum of the political contributions of all males in the CS department.

COMPROMISE

As mentioned earlier, the aim of a statistical database is to provide statistical information about a group of individuals without revealing information about any specific individual. The example below shows how a user can deduce information from the database of Table 2.1 about an individual, say Dodd. If the user has pre-knowledge that Dodd is a female Professor in the CS department and wants to find Dodd's salary, the user could first ask the query:

COUNT ((Sex=F) & (Dept=CS) & (Position=Prof))

The user would now conclude that Dodd is the only one with the above characteristics because the response to the above query is one. It is now quite trivial to deduce Dodd's salary. The query to determine Dodd's salary would be:

SUM ((Sex=F) & (Dept=CS) & (Position=Prof); Salary)

Since information about an individual (Dodd in this case) was not known previously and has been deduced, compromise or disclosure is said to have taken place. More formally, compromise or disclosure occurs when one can gather information which is not previously known about an individual from one or more queries.

Most databases do not answer all queries. As an example, the query SUM(C;j) may not be supported/permitted by some databases. Even when queries such as SUM(C;j) are not allowed, a user can still deduce information (say salary) about an individual (Dodd). The scheme to do this was given by Hoffman and Miller [HOFF70]. Once it is established that the characteristic C = ((Sex=F) & (Dept=CS) & (Position=Prof)) identifies Dodd, the user could check if Dodd's salary is any value (say $20K). This query would be:

COUNT((Sex=F) & (Dept=CS) & (Position=Prof) & (Salary=20))

A response of 0 indicates that Dodd does not earn $20K. The user could infer the exact salary of $15K when the response to a query is one. In this example, the query would be:

COUNT ((Sex=F) &(Dept=CS) & (Position=Prof) & (Salary=15))

Compromise still seems to have taken place when a user deduces that Dodd does not earn $20K because the user did not have this information earlier. This kind of compromise in which it is revealed that an individual does not have a particular value in one of the data fields of that individual is called negative compromise. In a positive compromise, it is revealed that an individual has a particular data value in one of the fields of that individual's record. For example, deducing that Dodd's salary is $15K is positive compromise.

Complete compromise occurs when one deduces everything in the database. Partial compromise occurs if deductions regarding some individuals can be made but the entire database is not deduced. Further, if no positive or negative compromise can occur in a database, then the database is strongly secure. If only negative compromise can occur in a database then the database is weakly secure.

Since databases can be compromised, many queries on the database are not allowed. Queries that are permitted by the database are permitted queries, otherwise they are restricted queries. Schlörer [SCHL80] distinguishes the knowledge a user possesses because of permitted queries (working knowledge) from the knowledge a user learns/possesses from "anything which cannot be learned from publicly available system discription plus normal statistical evaluation" (supplementary knowledge). It must be pointed out that it is very difficult to know exactly how much supplementary knowledge a user

possesses. It may be reasonable to assume that for the database of Table 2.1, a user would know the sex, department and position of an individual.

Disclosure could be of two types [SCHL80, HAQ75]: statistical disclosure and personal disclosure. Statistical disclosure occurs if a user learns a restricted statistical quantity. Statistical disclosure could either be resultant disclosure (when only working knowledge is used) or external disclosure (when supplementary knowledge must be used). Personal disclosure occurs when a user gains a piece of new information about an individual.

PROTECTION MECHANISMS

It was seen earlier that compromise/disclosure is possible from statistical databases. Before any protection mechanisms are presented, one must look into the various ways in which data is distributed to the users. Once the means of dissemination are identified, mechanisms to prevent disclosure can be presented. Fellegi [FELL77] pointed out three kinds of dissemination programs:

(1) Printed publications

(2) Public use of tapes

(3) Custom-made retrievals or query-based statistical outputs [SCHL83b]

Although these dissemination programs seem to be different, the security problems in all the above dissemination programs are similar. The mechanisms for protection apply to all three dissemination

programs. Greater emphasis would however be put on query-based statistical outputs. The reason is that printed publications and tapes are planned dissemination programs. One can either control, restrict or control and restrict the amount of information published. Custom-made retrievals are made on multi-purpose databases and the consequences of this are [FELL77]:

(1) The information in the public domain increases.

(2) Each answered query represents a potential risk in compromising the database for future retrievals.

There are possibly many ways to categorize the protection mechanisms [PALM74, SCHL83b, DENN78, DENN80b]. Basically, there are two possibilities regarding the storage of data in the database:

(1) Dummy/modified data is stored in the database.

(2) Actual data is stored in the database.

The strategies for protection against compromise change with the way in which the data is stored. Hence, they are considered seperately.

STORAGE OF DUMMY DATA IN THE DATABASE

In this scheme, actual data is not stored in the database. There are three basic schemes for modifying the database:

(1) Micro-aggregation

(2) Random modification of data

(3) Data Swapping

Micro-aggregation

In micro-aggregation, individuals with similar characteristics are grouped together to form single "aggregate individuals" [FEIG70]. These "aggregate individuals" replace the actual data. Statistics are computed for these aggregates rather than the real ones.

Questions do arise as to how much to aggregate and which individuals need to be chosen for aggregation. It was rightly pointed out by Feige and Watts [FEIG70] that the cost of aggregation must be measured in terms of the usefulness of the aggregated data for research purposes. They examined the usefulness of this data when they were taken as inputs to regression analysis. When the regression model is known in advance, it is possible to devise grouping schemes that avoid disclosure of individual microdata and still maintain the property that the grouped estimators are unbiased. It seems quite unreasonable to expect the knowledge of the regression model for query based databases.

For published data (such as printed publications and public use of tapes), this strategy could probably be implemented at a high cost. For multipurpose data which allows custom-made retrievals, this strategy is almost impossible to implement. The database is usually not static. Modifications may be made continuously to the database. Under such circumstances, the choice of individuals for aggregations may need continuous change. The cost of this evaluation after every modification of the database can be prohibitive and hinder the usefulness of the database.

Random modification of the data

In this strategy, some of the data could be modified at random and stored. Reed [REED73] related security in data banks with information theory. He defined a privacy transformation T which transformed each record as it was saved for use in data banks. Associated with each element in the privacy transform T_k was a probability P_k. Each record was transformed to a new value which depended both on the privacy transform and the associated probabilities. This transformation can also be applied to the data every time it is retrieved from storage. The data stored could be the actual data itself. The cost of doing this is going to be large since the transformation is applied to each record.

More recently, Traub, Yemeni and Wozniakowski [TRAU84] suggested that instead of storing the actual record, a record distorted by a random perturbation vector be stored. The components of the perturbation vector are random with a mean zero. For example, in the database of Table 2.1, the values of salary and political contribution for Dodd could be stored as \$14K and \$53 respectively. A problem with this method is that there could be queries which could result in large errors. This could easily happen if the user chooses a group of records for which all perturbations are on one side of the mean and therefore the resultant error for all the records grouped together may be unacceptable. A suggestion by the authors was to monitor the error and to take appropriate action when it exceeded a certain threshold.

This strategy would require storage of the actual data also. The cost of storing both the actual as well as the perturbed data could be large and therefore maybe unacceptable for most organizations.

Data Swapping

In multidimensional transformation or data swapping, the values of fields of records are interchanged [SCHL81]. The data field in a record for any particular individual need not be correct. In a sense, the database is transformed to a new database. Data swapping therefore reduces the risk of compromise. However, there is no efficient way of finding which records are to be used for data swapping.

For example, in the database of Table 2.1, one could swap the salaries of Adams and Dodd because they are both professors in the CS department. A question could be raised, however, as to why a swap is made between individuals of opposite sexes. The determination of which records to use in a swapping operations is not a trivial process and can be very costly.

STORAGE OF ACTUAL DATA IN THE DATABASE

In these schemes, the actual data is stored in the database. While presenting the data, or while answering queries, two control strategies can be implemented:

(1) Output restriction techniques:

A query is answered or a value published only if some conditions are satisfied. However, the response is always the true value. Some of the techniques are:

(i) Cell suppression techniques

(ii) Controls on the size of the query set.

(iii) Controls on the overlap of queries.

(iv) Table restrictions

(2) Output perturbation techniques

In these techniques the answer to the query or the value to be published is perturbed from the true value. There are two categories of perturbation techniques:

(i) Record based perturbations

(ii) Rounding techniques

Cell suppression techniques

These techniques are popular among the census agencies [COX75, COX77, COX79, COX80, JABI77, ZEIS77] where the dissemination programs are printed publications or public use tapes. The published data are viewed as tables. These tables consist of cells. Under cell suppression techniques, all cells identified as disclosure cells (or sensitive cells) are suppressed from publication. A cell is considered sensitive if an unacceptable estimate of the value of the data cell is made from the data. Merely suppressing sensitive cells is not enough since users may find out what the sensitive cells are

and find the values of these cells by algebraic manipulation. Therefore, related non-sensitive cells, called complementary suppressions, may also be suppressed from publication. The non-sensitive cells are chosen so as to ensure that no value of sensitive cells may be derived from the published data.

It may be feasible to apply cell suppression for custom-made or query based retrievals. This, however, needs further study [SCHL83b].

Controls on the size of the query set

Once it is established that a set of characteristics identifies a specific individual, the database is easily compromised as shown in earlier examples. It was shown for the database of table 2.1, that Dodd's salary could be deduced from either of the queries:

SUM ((Sex=F) & (Dept=CS) & (Position=Prof); Salary)

COUNT ((Sex=F) & (Dept=CS) & (Position=Prof) & (Salary=15))

A simple solution to avoid identification of an individual could be to have controls on the query size n_C for any characteristic formula C. A query q(C) is answered only if the query size n_C is in the range [k,N-k], where k is a chosen parameter and N is the total number of records in the database.

Although this seems to be a good idea, this control is easily subverted by a tool called the tracker [DENN79a, SCHL75, SCHL80, DENN80a]. A tracker is a set of characteristic formulas which help in

padding the query set of the original formula to form answerable queries.

Schlorer [SCHL75] considered the case where k was in the range (1,N/2] and the query set size was in the range [k,N-k]. He introduced the concept of an individual tracker. If a user wants to find the answer to a query q(C) which is restricted, then, to deduce q(C), the user could find a split of C into disjoint sub-characteristics A and B where C = A & B such that q(A & $\bar{B}$) and q(A) are both answerable. q(C) is then calculated from:

$$q(C) = q(A) - q(A \& \bar{B}) \qquad (5)$$

The formula T= A & $\bar{B}$ is called the individual tracker. To illustrate the use of the individual tracker, consider the database of Table 2.1. If a user knows that Dodd is identified uniquely by the characteristic:

C = (Sex=F) & (Dept=CS) & (Position=Prof)

and if only those queries whose size is in the range [2,10] (i.e. k=2) are answered, then an unanswerable query q(C) is calculated from Eq. (5) using the tracker T = A & $\bar{B}$ where A = (Sex=F) and B = ((Dept=CS) and (Position=Prof)). In fact, Eq. (5) could be used to verify that Dodd is the only individual with the characteristics given by C:

COUNT((Sex=F) & (Dept=CS) & (Position=Prof))

= COUNT(Sex=F) - COUNT((Sex=F)&($\bar{\text{(Dept=CS)}}$ & $\bar{\text{(Position=Prof)}}$))

= 5-4

=1

It is to be noted that both q((Sex=F) & ¬(Dept=CS & Position=Prof)) and q(Sex=F) are permitted queries. To determine Dodd's Salary, Eq. (5) can be applied utilizing the SUM query:

SUM((Sex=F) & (Dept=CS) & (Position=Prof); Salary)

= SUM(Sex=F;Salary)-SUM((Sex=F)&¬(Dept=CS)&¬(Position=Prof);Salary)

= $90K - $75K

= $15K

Individual trackers must be found for each individual for complete compromise. A tracker called the general tracker applicable to all individuals in the database was presented by Denning, Denning and Schwartz [DENN79a].

A general tracker is any characteristic formula T whose query set size is in the range [2k,N-2k]. Therefore, the value of k is restricted to [0,n/4]. Any restricted query q(C) may be calculated from:

$$q(C) = q(C+T) + q(C + \bar{T}) - q(T) - q(\bar{T}) \quad \text{if COUNT}(C)<k \qquad (6a)$$

$$q(C) = 2q(T) + 2q(\bar{T}) - q(\bar{C} + T) - q(\bar{C} + \bar{T}) \quad \text{if COUNT}(C)>N-k \qquad (6b)$$

It was shown that all the queries on the right hand side of Eq. (6) were answerable. For example, if k=2, then the general tracker must have a query set size in the range [4,8] for the database of Table 2.1. A general tracker could be T=(Sex=M) since COUNT(T) is 7. Dodd's salary can be determined from:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= SUM(((Sex=F)&(Dept=CS)&(Position=Prof)) + (Sex=M); Salary) +

SUM(((Sex=F)&(Dept=CS)&(Position=Prof)) + ($\tilde{\text{Sex=M}}$); Salary) -

SUM((Sex=M);Salary) - SUM(($\tilde{\text{Sex=M}}$);Salary)

= $119K + $90K - $104K - $90K

= $15K

There could be many general trackers. For example, T = (Dept=CS) is also a general tracker since COUNT(T) is 5 and is in the range [4,8]. Denning, Denning and Schwartz [DENN79a] found an even more powerful tracker called the double tracker. For a general tracker to be found, k must be in the range [0,n/4]. In the case of a double tracker, k needs to be in the range [0,n/3]. A double tracker is a pair of characteristic formulas (T,U) satisfying:

$$X_T \subseteq X_U \tag{7a}$$

COUNT(T) is in the range [k,N-2k] (7b)

COUNT(U) is in the range [2k,N-k] (7c)

Any restricted query q(C) is found from:

$$q(C) = q(U) + q(C+T) - q(T) - q(\tilde{C}\&\tilde{T}\&U) \text{ for COUNT}(C)<k \tag{8a}$$

$$q(C) = q(\tilde{U}) - q(\tilde{C}+T) + q(T) + q(\tilde{\tilde{C}}\&T\&U) \text{ for COUNT}(C)>N-k \tag{8b}$$

For example, if k=4, there cannot be any general tracker because of range restrictions. However, (T,U) = ((Dept=Math),(Position=Prof)) is a double tracker since it satisfies Eq. (7):

X_T = records of Baker, Cook, Hayes and Knapp

X_C = records of Adams, Baker, Cook, Dodd, Engel, Flynn, Hayes and Knapp

$X_T \subseteq X_U$

COUNT(T)=4 and is in the range [4,4]

COUNT(U)=8 and is in the range [8,8]

To determine Dodd's salary:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= SUM((Position=Prof);Salary)+

SUM(((Sex=F)&(Dept=CS)&(Position=Prof))+(Dept=Math);Salary)

- SUM((Dept=Math);Salary)

- SUM((Sex=F)&(Dept=CS)&(Position=Prof)&(Dept=Math)&

(Position=Prof); Salary)

= $158K + $98K - $83K - $158K

= $15K

Clearly, trackers are powerful tools for disclosure. It was shown that trackers can be discovered using only a few queries and in addition that there are an abundance of trackers for most databases [SCHL80, DENN80a]. It is therefore obvious that in order to avoid disclosures by controlling the query set size, one would have to severely restrict the range of allowable queries and this could render the database useless for normal statistical processing.

Controls on the overlap of queries

It was seen in the discussion of the control of query set size that equations involving trackers isolate a single record. The queries in the right hand side of Eqs. (5), (6) and (8) have many records in common, and these queries are manipulated algebraically to nullify the effect of these common records.

Davida et al. [DAVI78], Dobkin, Jones and Lipton [DOBK79], and DeMillo, Dobkin and Lipton [DEMI78] have shown how a set of queries with large overlap of records could be used to compromise the database. In some databases, the response to a query is a weighted sum of the elements in the query set. These weights are usually kept secret. By a clever overlap of query sets, Schwartz, Denning and Denning [SCHW79] have shown that the database can be compromised if the user has sufficient information about the records in the database.

A strategy which would not allow compromise would be to stop the overlap of records in queries. There are three ways of implementing this strategy:

(1) Keeping history

(2) Implied queries

(3) Database partitioning

Keeping history

One way to stop overlap is to keep history of all the queries by a user. The programs that monitor all requests to the system and keep audit trials are called threat monitoring control programs [HOFF70].

A technique for managing the past history of user's queries was given by Chin and Ozsoyoglu [CHIN82].

A query is not answered if the number of records common to two queries is more that a specified quantity. An implementation of this approach is to check the number of records common to two consecutive queries. If a user decides to intersperse the queries with dummy queries, this implementation is easily subverted. Another implementation could be to remember all the queries. The number of queries to be monitored and compared can increase rapidly. However, there is no guarantee that a user does not get the answers to the queries by colluding with some of his/her cohorts. Another problem could be that this restriction may hinder a genuine user from getting needed information from the database.

Implied queries

Friedman and Hoffman [FRIE80] introduced the concept of an implied query. For any query or a set of queries, the queries that can be deduced are called implied queries. A query is answered only if the query and its associated implied queries have query set sizes in the range [k,N-k], where k is a given parameter. For example, for the database of Table 2.1:

a_1 = COUNT(Sex=M) = 7

a_2 = COUNT((Sex=M) & (Dept=CS)) = 4

One can deduce:

a_3 = COUNT((Sex=M) & (Dept=CS)) = a_1-a_2 = 3

Therefore, COUNT((Sex=M)&(Dept=CS)) is an implied query. Note that if k was 4, the size of the query sets must be in the range [4,8] for a query to be answerable. Thus, the query COUNT((Sex=M)&(Dept=CS)) is not answerable. However, by knowing a_1 and a_2 from allowable queries, a_3 can be deduced.

In the method proposed by Friedman and Hoffman [FRIE80], the query COUNT((Sex=M)&(Dept=CS)) would also be restricted and therefore unanswered, because the associated implied query COUNT((Sex=M)& (Dept=CS)) has a value outside the range [4,8].

This approach avoids the difficulties due to history keeping. Denning [DENN81] has shown that there is an exponential growth of the number of implied queries as the number of specified attributes in a query increases. She has also shown that this control would not prevent deduction of sensitive statistics.

Database partitioning

Another approach to preventing compromise is to partition the database into groups [YUCH77, SCHL83c]. A database is partitioned into mutually exclusive, non-overlapping record sets called "atomic populations" [SCHL83c]. Each of the atomic populations must have either no records or more than one record. A query is answered only if its query set is a union of some of the atomic populations. The

attributes used in the partitioning of the database must be used as characteristic formulas in queries.

For example, the database of Table 2.1 could be partitioned into the groups shown below:

Group	Characteristic	Members
1	(Sex=F)	Cook, Dodd, Irons, Knapp, Flynn
2	(Sex=F)& (Position=Prof)	Adams, Baker, Engel, Hayes
3	(Sex=M) & ((Position=Stu) + (Position=Adm))	Grady, Jones, Lord

Only queries involving entire groups are allowed. For the partitions given above, only queries whose query sets are subsets of either group 1, 2 or 3 are allowed. Query sets whose members are in the intersection of the member sets in different groups are not allowed. Yu and Chin [YUCH77] showed that partitioning could prevent compromise even when the database is being modified. The partitioning will often result in either high information loss or serious distortion of important statistical functions [SCHL83c].

Table restrictions

A table is defined by the set of characteristic attributes whose values occur in a characteristic formula [SCHL83b]. An m-table has m attributes. A relative table size s_m/N for an m-table is the ratio of

the product of the domain sizes of the m-attributes that specify the table and the total number of records in the database. For the example of Table 2.1, N=12, since there are twelve records. For the query:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

the table is a 3-table because there are three attributes (Sex, Dept and Position) in the characteristic formula. The absolute table size is:

s_3 = |Sex| * |Dept| * |Position|

= 2 * 3 * 3

= 18

The domain sizes of Sex, Dept and Position are 2 (M and F), 3 (CS, Math, Stat) and 3 (Prof, Adm, Stu) respectively. The relative table size would be $s_3/N = 1.5$.

From empirical investigations, it was determined that for s_m/N in the range [0.01,0.1], the risk of identification of an individual from actual databases were similar for a given table size. Thus, a criterion of s_m/N was used to estimate the risks of identification. In the table restriction technique, for each query, the size of the table is determined and the identification risk is extracted from a look-up table. If the risk exceeds a predetermined (threshold) quantity, the query is witheld. Table restriction does not eliminate loss of information and the threshold value must be tuned for each database.

Record based perturbations

There are two categories in this method for avoiding compromise:

(1) Random sample queries:

The records used to find the required statistic is not the entire query set. There is a probability associated with selecting any record from the query set. The sample of records chosen from the query set is used to determine the statistic [DENN80b]. The implementation is such that the same query will result in the same statistic because the same records would be chosen as a sample. If the selection of records were completely random, the value of the statistic returned would be different each time a user queries the database with the same query. The user could then estimate the true value of the statistic by querying the database several times with the same query.

This method works well for large databases but the cost could be very high because the method requires checking each record for inclusion in the sample.

(2) Random perturbation:

In the method proposed by Beck [BECK80], each data item used in calculating the statistic is perturbed. The perturbations to each record could be varied independently. An implementation which minimized the error involved in determining the statistic was given. This method is expensive because the data value for each record must be perturbed.

Rounding techniques

In these techniques, the true statistic is calculated and then the final result is perturbed. There are many ways of doing this:

(1) Systematic rounding:

The final statistic is rounded to the closest integer multiple of a given base. Fellegi and Phillips [FELL74] showed how rounding to multiples of integers could be subverted in printed publications.

Another variation of systematic rounding is to report a range (e.g. 0-5, 5-10, ...). According to Karpinski [KARP70], this is subverted if a user is allowed to add (or delete) records to the database. A user can add/deleted records with known data values till there is a change in the reported range. By arithmetic manipulation, the user can now find the actual response to the query that he/she was seeking. Even if modification of the database is not allowed, as is the case in most statistical databases, the database could still be compromised if a user has knowledge of some of the data values in the database.

(2) Random rounding:

Fellegi and Phillips [FELL74] suggested random rounding for published data. They rounded a table value to the nearest integer multiple of a chosen number. There was a probability associated with the rounding scheme. The choice of the rounding base was discussed by Nargundkar and Saveland [NARG72].

Chapter III

A NEW DETERRENT TO COMPROMISE

INTRODUCTION

In the previous chapter, various methods of compromise and the means to avoid compromise were discussed. All the methods to avoid compromise were either very expensive to implement or would let the database be compromised under certain situations/conditions. In the present study, a new scheme for avoiding compromise is presented.

In this investigation, compromise of an individual's confidential information is considered. The present study can be extended to consider the compromise of confidential information about groups of individuals. In addition, compromise is assumed to occur if a user can infer the exact value of any field of an individual's record in the database. In the case of data fields of a record, statistical compromise may also be defined. This study does not deal with statistical compromise.

COMPROMISE AVOIDANCE STRATEGY

The proposed method is to report results from a set of records which is obtained by duplicating/deleting a record from the query set. The scheme is the following:

1. A query $q(C)$ is answerable if the query set size, $|q(C)|$, is in the range $[k, N-k]$, where k is a chosen parameter and N is the total number of records in the database.

2. If a query, q(C), is answerable, then one of the following three options is chosen in order to report the results to the user:
   a. The query response is calculated from the set of records obtained after duplicating a record in the query set.
   b. The query response is calculated from the set of records formed by deleting a record from the query set.
   c. The query response is the true value.
3. The decision to choose one of the three options is random. However, it is necessary that the two conditions below be satisfied:
   a. The same option must be chosen for any query with the same query set.
   b. If two queries result in the same query set, and if the option chosen is to duplicate/delete a record, the same record must be duplicated/deleted from the two query sets regardless of the order in which the records are put together in the query sets.

The reason for these restrictions is that a compromise would occur if different options are chosen whenever the same query is posed repeatedly to the database. An accurate estimate of the true response would be the average of the all the responses.

EFFECTIVENESS AGAINST INDIVIDUAL TRACKERS

It would be of interest to see how the scheme proposed in the current investigation responds to the problem of trackers. As

mentioned earlier, any unanswerable query q(C), where C identifies an individual, can be made answerable if the characteristic C can be split into two characteristics A and B such that q(A) and q(A&$\bar{B}$) are both answerable:

$$q(C) = q(A) - q(A\&\bar{B})$$

The individual tracker is $T = A\&\bar{B}$.

For a statistical analysis of compromise from individual trackers, the following assumptions were made:

1. The following probabilities were assumed:

   p_1 = Probability of choosing the option to duplicate a record in the query set.

   p_2 = Probability of choosing the option to delete a record from the query set.

   p_3 = Probability of choosing the option to return the true response.

2. Should the decision be to duplicate/delete a record, it was assumed that it is equally likely that any of the records in the query set be chosen for duplicating the record or for deleting the record.

3. The data values in any data field for the records in a query set were assumed to be distinct.

Since the characteristic C identifies an individual, the query set sizes $|q(A)|$ and $|q(A\&\bar{B})|$ satisfy the following formula:

$$|q(A)| = |q(A\&\bar{B})| + 1$$

Given the above assumptions, compromise can occur only if one of the following conditions hold:

1. True values are returned for both $q(A)$ and $q(A\&\bar{B})$.
2. The same record is duplicated from the query sets corresponding to the queries $q(A)$ and $q(A\&\bar{B})$.
3. The same record is deleted from the query sets corresponding to the queries $q(A)$ and $q(A\&\bar{B})$.

It is easy to see that should any of the above conditions be false, the true value is not reported and compromise will not occur unless the user knows the following:

1. The option taken when the response is given to his/her query.
2. The ordering of the records in the query sets.
3. The data values in the data fields of the records.

In this investigation it is assumed that the user does not have such a large amount of information regarding the database. Under such circumstances, the probability that a compromise occurs using the individual tracker can be determined. Let p_a, p_b and p_c be defined as follows:

p_a = probability that true values are returned for the queries q(A) and q(A&$\bar{B}$).

= $p_3 p_3$

= p_3^2

p_b = Probability that the same record in the query sets q(A) and q(A&$\bar{B}$) was duplicated.

= $p_1 \cdot \frac{1}{n} \cdot p_1$ where n is the query set size for the query q(A).

p_c = Probability that the same record in the query sets q(A) and q(A&$\bar{B}$) was deleted.

= $p_2 \cdot \frac{1}{n} \cdot p_2$ where n is the query set size for the query q(A).

The probability of compromise is:

$P = p_a + p_b + p_c$

$= p_3^2 + \frac{1}{n} (p_1^2 + p_2^2)$

To get an upper bound on the probability, n-1 has to be at least k for the query q(A&$\bar{B}$) to be answerable. Therefore:

$P \leq p_3^2 + \frac{1}{k+1} (p_1^2 + p_2^2)$

Further, if the probabilities of duplicating/deleting a record are equal (= p), then:

$P \leq (1-2p)^2 + 2p^2/(k+1)$

or

$$P \leq (4 + 2/(k+1))p^2 - 4p + 1$$

To lessen the possibility of compromise, P must be as small as possible. Therefore, the minimum value of the function $(4+2/(k+1))p^2-4p+1$ needs to be determined:

$$\frac{d}{dp}\left[(4+2/(k+1))p^2-4p+1\right] = 0$$

or

$$p = \frac{k+1}{2k+3}$$

To check if the value obtained is a minimum, the second derivative of the function $(4+2/(k+1))p^2-4p+1$ needs to be taken. The second derivative is positive for all positive values of k, indicating that the minimum value of the function is when $p = (k+1)/(2k+3)$. Table 3.1 gives the values of p and the upper bound for P for various values of k.

Table 3.1. Values of p and upper bounds for P for various values of k.

k	p	P
1	0.4000	0.2000
2	0.4286	0.1429
3	0.4444	0.1111
4	0.4545	0.0909
5	0.4762	0.0476
10	0.4878	0.0244
20	0.4884	0.0231

From Table 3.1, it may be concluded that for large values of k, the probability of deletion/duplication of a record should be high (approximately 0.5) to keep the probability of compromise low.

EFFECTIVENESS AGAINST GENERAL TRACKERS

General trackers were discussed in the previous chapter. These trackers could be used to obtain confidential information about all individuals in a database. As mentioned earlier, a general tracker T is a characteristic formula whose query set size is in the range [2k,N-2k] where N is the number of records in the query set. Any restricted query q(C) may be calculated from:

$$q(C) = q(C+T) + q(C+\bar{T}) - q(T) - q(\bar{T}) \qquad \text{if COUNT}(C)<k$$

$$q(C) = q(T) + 2q(\bar{T}) - q(\bar{C}+T) - q(\bar{C}+\bar{T}) \qquad \text{if COUNT}(C)>N-k$$

In order to obtain bounds on the probability of compromise, any of the above two equations may be considered. However, for this analysis, the case when COUNT(C)<k is considered:

$$q(C) = q(C+T) + q(C+\bar{T}) - q(T) - q(\bar{T}) \qquad \text{if COUNT}(C)<k$$

The same assumptions made in the analysis of the effectiveness of the proposed method against individual trackers is made in this analysis also.

Compromise can occur in many different ways. This is best summarized in Table 3.2. Each row in Table 3.2 represents the conditions that must hold for compromise to occur.

Table 3.2. Conditions under which compromise can occur when general trackers are used to compromise the database.

Group	No.	q(C+T)	q(C+$\bar{T}$)	q(T)	q($\bar{T}$)
A	1	t	t	t	t
B	2	a1	t	a1	t
B	3	t	a1	a1	t
B	4	a1	t	t	a1
B	5	t	a1	t	a1
C	6	a1	a2	a1	a2
C	7	a1	a2	a2	a1
C	8	d1	d2	d1	d2
C	9	d1	d2	d2	d1
D	10	d1	t	d1	t
D	11	t	d1	d1	t
D	12	d1	t	t	d1
D	13	t	d1	t	d1
E	14	a1	d2	a1	d2
E	15	a1	d2	d2	a1
E	16	d2	a1	a1	d2
E	17	d2	a1	d2	a1
E	18	a1	d1	t	t
E	19	d1	a1	t	t

In the table, a "t" under a query indicates that a true value is returned for the query. An "a1" indicates that a record is duplicated (added) and "d1" indicates that a record is deleted. Two a1's in a row indicates that the same record is duplicated in response to the corresponding queries where the a1's appear. The d1's are similar

except that the same records are deleted. Some examples are given below:

1. Row 1 in Table 3.2 corresponds to the case where true responses are returned for all queries.
2. Row 2 in Table 3.2 corresponds to the case where true responses are returned for queries $q(C+\bar{T})$ and $q(\bar{T})$, and the same record is added when computing the responses to queries $q(C+T)$ and $q(T)$.
3. Row 14 in Table 3.2 corresponds to the case where the same record is added when computing the responses to queries $q(C+T)$ and $q(T)$, and the same record (possibly different from the previous one) is deleted when computing the response to queries $q(C+\bar{T})$ and $q(\bar{T})$.

There are other ways by which compromise can occur. For example, records may be duplicated in queries $q(C+T)$ and $q(C+\bar{T})$ and a record having a data value equal to the sum of the data values in the duplicated records may be duplicated in either $q(T)$ or $q(\bar{T})$. It is assumed that the probabilities of such situations occurring are small and hence are neglected.

The various conditions given in Table 3.2 are divided into six groups A, B, C, D, E and F, in order to calculate the probability of compromise. To write the probabilities for each group, let the query

set sizes $|q(C\&T)|$, $|q(C\&\bar{T})|$, $|q(\bar{C}\&T)|$ and $|q(\bar{C}\&\bar{T})|$ be u, v, w and x respectively. This is shown as a Venn diagram in Fig. 3.1.

Fig. 3.1. Venn diagram showing the query sets $q(C\&T)$, $q(C\&\bar{T})$, $q(\bar{C}\&T)$ and $q(\bar{C}\&\bar{T})$

	T	$\bar{T}$
C	u	v
$\bar{C}$	w	x

Let the probabilities that the conditions in groups A, B, C, D, E, and F of Table 3.2 hold be p_a, p_b, p_c, p_d, p_e, and p_f respectively. Since the probability of returning the true response is p_3, p_a may be written as:

$$p_a = p_3^4$$

The probability, p_b, that the conditions in group B of Table 3.2 holds is:

$$p_b = p_1^2 p_3^2 \left[\frac{1}{u+v+w} + \frac{u}{(u+v+x)(u+w)} + \frac{v}{(u+v+w)(v+x)} + \frac{1}{u+v+x} \right]$$

Similarly,

$$p_c = \frac{1}{(u+v+w)(u+v+x)} \left[p_1^4 + p_2^4 + \frac{uvp_1^4}{(v+x)(u+w)} + \frac{uvp_2^4}{(v+x)(u+w)} \right]$$

$$p_d = p_2^2 p_3^2 \left[\frac{1}{u+v+w} + \frac{u}{(u+v+x)(u+w)} + \frac{v}{(u+v+w)(v+x)} + \frac{1}{u+v+x} \right]$$

$$p_e = 2p_1^2 p_2^2 \left[\frac{1}{(u+v+w)(u+v+x)} + \frac{uv}{(u+v+x)(u+w)(u+v+w)(v+x)} \right]$$

and

$$p_f = 2p_1 p_2 p_3^3 \left[\frac{u+v}{(u+v+x)(u+v+w)} \right]$$

The probability that a compromise occurs is:

$$P = p_a + p_b + p_c + p_d + p_e + p_f$$

or

$$P = p_3^4 + p_3^2(p_1^2+p_2^2)\left[\frac{1}{u+v+w} + \frac{u}{(u+v+x)(u+w)} + \frac{v}{(u+v+w)(v+x)} + \frac{1}{u+v+x} \right]$$
$$+ (p_1^2+p_2^2)^2 \left[\frac{1}{(u+v+w)(u+v+x)} + \frac{uv}{(u+v+x)(u+w)(u+v+w)(v+x)} \right]$$
$$+ 2p_1 p_2 p_3^2 \left[\frac{u+v}{(u+v+x)(u+v+w)} \right]$$

In order to find the upper bound for the probability of disclosure of an individual's confidential information, the following relation may be written:

$$u + v = 1$$

Also, since $q(C+T)$, $C+T)^-$, $q(T)$ and $q(T)^-$ are answerable and by the definition of general trackers, the following bounds are obtained:

$$k \leq |q(C+T)| \leq N-k \quad \text{or} \quad k \leq u+v+w \leq N-k$$

$$k \leq |q(C+\bar{T})| \leq N-k \quad \text{or} \quad k \leq u+v+x \leq N-k$$

$2k \leq |q(T)| \leq N-2k$ or $2k \leq u+w \leq N-2k$

$2k \leq |q(\bar{T})| \leq N-2k$ or $2k \leq v+x \leq N-2k$

If the probabilities for duplicating and deleting a record are equal (= p), then an upper bound for the probability of compromise may be written as:

$$P \leq (1-2p)^4 + 2(1-2p)^2p^2 \left[\frac{2}{k} + \frac{u}{2k^2} + \frac{v}{2k^2} + \frac{1}{k^2} \right] + 4p^4 \left[\frac{1}{k^2} + \frac{uv}{4k^4} \right]$$

Since u+v=1, u≥0, and v≥0, either u or v must be 0. Therefore:

$$P \leq (1-2p)^4 + \frac{p^2}{k^2}(1-2p)^2(4k+3) + \frac{4p^4}{k^2}$$

Optimum values for p for given values of k may be found from the above equation. However, this involves solving the roots of a cubic equation, which has three roots. To simplify the analysis, the values of P were calculated for the optimum values of p obtained in the analysis for individual trackers. The values of P obtained are given in Table 3.3.

Table 3.3. Values of P for general trackers.

k	p	P
1	0.4000	0.1488
2	0.4286	0.0431
3	0.4444	0.0216
4	0.4545	0.0107
5	0.4762	0.0087
10	0.4878	0.0023
20	0.4884	0.0006

Comparing the values of P in Tables 3.1 and 3.3, it is clear that the proposed scheme is very effective in avoiding compromise. It must be mentioned that these are probabilities that an exact answer may be computed by algebraically manipulating the query responses. However, a user trying to obtain confidential information would not know when the true answer is computed.

EFFECTIVENESS AGAINST DOUBLE TRACKERS

Double trackers were also discussed in the previous chapter. These trackers were more powerful than general trackers. A double tracker is a pair of characteristic formulas (T,U) satisfying:

$X_T \subseteq X_U$

COUNT(T) is in the range [k,N-2k]

COUNT(U) is in the range [2k,N-k]

Any restricted query q(C) is found from:

$q(C) = q(U) + q(C+T) - q(T) - q(\bar{C}\&\bar{T}\&U)$ for COUNT(C)<k

$q(C) = q(\bar{U}) - q(\bar{C}+T) + q(T) + q(\bar{\bar{C}}\&T\&U)$ for COUNT(C)>N-k

Without loss of generality, only the the first of the last two equations may be considered. To obtain the bounds on the probability of compromise, the same assumptions made in the analysis of the effectiveness of the proposed method against individual trackers is made in this analysis also.

The different ways in which compromise can occur is summarized in Table 3.4. The format of the table is similar to the format of Table 3.2. The symbols in Tables 3.2 and 3.4 have the same meaning.

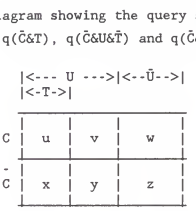
Table 3.4. Conditions under which compromise can occur when double trackers are used to compromise the database.

Group	No.	q(C+T)	q(U)	q(T)	q(C̄&T̄&U)
A	1	t	t	t	t
B	2	a1	t	a1	t
B	3	t	a1	a1	t
B	4	a1	t	t	a1
B	5	t	a1	t	a1
C	6	a1	a2	a1	a2
C	7	a1	a2	a2	a1
C	8	d1	d2	d1	d2
C	9	d1	d2	d2	d1
D	10	d1	t	d1	t
D	11	t	d1	d1	t
D	12	d1	t	t	d1
D	13	t	d1	t	d1
E	14	a1	d2	a1	d2
E	15	a1	d2	d2	a1
E	16	d2	a1	a1	d2
E	17	d2	a1	d2	a1
F	18	a1	d1	a2	d2
F	19	d1	a1	a2	d2
F	20	a1	d1	d2	a2
F	21	d1	a1	d2	a2
G	22	a1	d1	t	t
G	23	d1	a1	t	t
G	24	t	t	a1	d1
G	25	t	t	d1	a1

Table 3.4 is divided into seven groups A, B, C, D, E, F and G. In order to write the probabilities of compromise for each group, let the query set sizes |q(C&T)|, |q(C&U&T̄)|, |q(C&Ū)|, |q(C̄&T)|,

$|q(\bar{C}\&U\&\bar{T})|$ and $|q(\bar{C}\&\bar{U})|$ be u, v, w, x, y and z respectively. These query set sizes are shown in Fig. 3.2.

Fig. 3.2. Venn diagram showing the query sets $q(C\&T)$, $q(C\&U\&\bar{T})$, $q(C\&\bar{U})$, $q(\bar{C}\&T)$, $q(\bar{C}\&U\&\bar{T})$ and $q(\bar{C}\&\bar{U})$.



Let the probabilities that the conditions in groups A, B, C, D, E, F, and G of Table 3.4 hold be p_a, p_b, p_c, p_d, p_e, p_f, and p_g respectively. Following the same procedure as in calculating the probabilities of compromise for general trackers, the following equations are obtained:

$$p_a = p_3^4$$

$$p_b = p_1^2 p_3^2 \left[\frac{1}{u+v+w+x} + \frac{1}{u+v+x+y} + \frac{v+x}{(u+v+w+x)(v+x+y)} + \frac{1}{u+v+x+y} \right]$$

$$p_c = \frac{p_1^4+p_2^4}{(u+v+w+x)(u+v+x+y)} + \frac{(p_1^4+p_2^4)(v+x)}{(u+v+x+y)(u+v+w+x)(v+x+y)}$$

$$p_d = p_2^2 p_3^2 \left[\frac{1}{u+v+w+x} + \frac{1}{u+v+x+y} + \frac{v+x}{(u+v+w+x)(v+x+y)} + \frac{1}{u+v+x+y} \right]$$

$$p_e = \frac{2p_1^2 p_2^2}{(u+v+w+x)(u+v+x+y)} + \frac{2p_1^2 p_2^2}{(u+v+x+y)(u+v+w+x)(v+x+y)}$$

$$p_f = 4p_1^2 p_2^2 \left[\frac{u+v+x}{(u+v+x+y)(u+v+w+x)} \cdot \frac{x}{(u+x)(x+v+y)} \right]$$

$$p_g = 2p_1 p_2 p_3^2 \left[\frac{u+v+x}{(u+v+x+y)(u+v+w+x)} + \frac{x}{(u+x)(x+v+y)} \right]$$

The probability that a compromise occurs is:

$$P = p_a + p_b + p_c + p_d + p_e + p_f + p_g$$

or

$$P = p_3^4 + (p_1^2+p_2^2)p_3^2 \left[\frac{1}{u+v+w+x} + \frac{1}{u+v+x+y} + \frac{v+x}{(u+v+w+x)(v+x+y)} + \frac{1}{u+v+x+y} \right] + (p_1^2+p_2^2)^2 \left[\frac{1}{(u+v+w+x)(u+v+x+y)} + \frac{(v+x)}{(u+v+w+x)(v+x+y)(u+v+x+y)} \right] + 4p_1^2 p_2^2 \left[\frac{(u+v+x)x}{(u+v+x+y)(u+v+w+x)(u+x)(x+v+y)} \right] + 2p_1 p_2 p_3^2 \left[\frac{u+v+x}{(u+v+x+y)(u+v+w+x)} + \frac{x}{(u+x)(x+v+y)} \right]$$

In addition, the following constraint holds if an individual's confidential information is sought:

$$u + v + w = 1$$

For the above queries to be answerable:

$$k \leq |q(T)| \leq N-2k \quad \text{or} \quad k \leq u+x \leq N-2k$$

$$2k \leq |q(U)| \leq N-k \quad \text{or} \quad 2k \leq u+v+x+y \leq N-k$$

$$k \leq |q(C+T)| \leq N-k \quad \text{or} \quad k \leq u+v+w+x \leq N-k$$

$$k \leq |q(\bar{C}\bar{\&}\bar{T}\&U)| \leq N-k \quad \text{or} \quad k \leq v+x+y \leq N-k$$

If the probabilities for duplicating and deleting a record are equal (= p), then an upper bound for the probability of compromise may be written as:

$$P \leq (1-2p)^4 + 4p^2(1-2p)^2 \left[\frac{1}{k} + \frac{1}{2k} \right] + 4p^4 \left[\frac{1}{k^2} + \frac{1}{2k^2} \right]$$

$$+ 2p^2(1-2p)^2 \left[\frac{1}{2k} + \frac{1}{k} \right]$$

or

$$P \leq (1-2p)^4 + \frac{9}{k} p^2(1-2p)^2 + \frac{6}{k^2} p^4$$

Once again, the values of P were calculated for the optimum values of p obtained in the analysis for individual trackers. The values of P obtained are given in Table 3.5.

Comparing the values of P in Tables 3.1, 3.3, and 3.5, it is clear that the proposed scheme is very effective in avoiding compromise. The value of P obtained for double trackers for k=1 is higher than that for individual and general trackers. There is no

specific reason that may be given except that the bounds for P found are not the least upper bounds and the method of calculating the upper bounds affects the values of P obtained. In general, the probabality of compromise is higher for double trackers than for general trackers because the number of ways by which compromise occurs is higher, as seen from tables 3.2 and 3.4.

Table 3.5. Values of P for double trackers.

k	p	P
1	0.4000	0.2128
2	0.4286	0.0679
3	0.4444	0.0335
4	0.4545	0.0199
5	0.4762	0.0133
10	0.4878	0.0035
20	0.4884	0.0009

STATISTICAL CONSEQUENCES

From the above analysis it is clear that the proposed scheme is effective against the problem of trackers.

With any output perturbation scheme, one must be careful that the response is not distorted to an extent that the response is not close enough to the actual or true response to be useful.

A quantification of the loss in precision due to the proposed strategy is given below.

Let the values of the data fields in n records of a query set be $X_1, X_2, \ldots, X_n$. For the sake of brevity, the set $\{X_1, X_2, \ldots, X_n\}$ will

be referred to as the query set. The following are two possibilities with regard to the query sets:

Case I. The query set could be a random sample from some population with the same characteristics queried and with a mean of μ and a variance of σ^2. An example of this possibility is a public domain census database.

Case II. The query is the population in which case

$$\mu = \sum_{i=1}^{n} X_i = \bar{X}$$

and

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

This corresponds to the case where the database includes the whole population. This is the more common situation where the database is for all the employees in an organization. It is in this situation that compromise is more likely.

With the perturbation strategy μ is estimated as

$$\hat{\mu} = I_1\hat{\mu}_1 + I_2\hat{\mu}_2 + I_3\hat{\mu}_3$$

where

$$\hat{\mu}_1 = \frac{1}{(n+1)} \left(\sum_{i=1}^{n} X_i + X_*\right)$$

$$\hat{\mu}_2 = \frac{1}{(n-1)} \left(\sum_{i=1}^{n} X_i - X_*\right)$$

$$\hat{\mu}_3 = \frac{1}{n} \sum_{i=1}^{n} X_i$$

X_* is the data value deleted or duplicated in the proposed method, and

$$I_i = \begin{cases} 1 \text{ with probability } p_i \\ 0 \text{ otherwise} \end{cases}$$

$$I_1 + I_2 + I_3 = 1$$

where

p_1, p_2 and p_3 are the probabilities of taking the option to duplicate a record, delete a record or returning the true response. Note that the I_i's and the $\hat{\mu}_i$'s are statistically independent.

The expected value of $\hat{\mu}$ is

$$E(\hat{\mu}) = E(I_1\hat{\mu}_1) + E(I_2\hat{\mu}_2) + E(I_3\hat{\mu}_3)$$

$$= E(I_1)E(\hat{\mu}_1) + E(I_2)E(\hat{\mu}_2) + E(I_3)E(\hat{\mu}_3)$$

$$= p_1E(\hat{\mu}_1) + p_2E(\hat{\mu}_2) + p_3E(\hat{\mu}_3)$$

For case I,

$$E(\hat{\mu}) = p_1 \frac{(n+1)}{(n+1)} \mu + p_2 \frac{(n-1)}{(n-1)} \mu + p_3 \frac{n}{n} \mu$$

$$= \mu$$

For case II,

$$E(\hat{\mu}) = p_1 E(\frac{1}{n+1} [n\mu + X_*]) + p_2 E(\frac{1}{n-1} [n\mu - X_*]) + p_3 E(\mu)$$

$$= \mu$$

Therefore, the estimator $\hat{\mu}$ is an unbiased estimator for μ under both the cases.

The accuracy of $\hat{\mu}$ is measured by the standard error of $\hat{\mu}$. The variance of $\hat{\mu}$ is

$$\mathrm{Var}(\hat{\mu}) = \mathrm{Var}(I_1\hat{\mu}_1) + \mathrm{Var}(I_2\hat{\mu}_2) + \mathrm{Var}(I_3\hat{\mu}_3) + 2\mathrm{Cov}(I_1\hat{\mu}_1, I_2\hat{\mu}_2)$$

$$+ 2\mathrm{Cov}(I_2\hat{\mu}_2, I_3\hat{\mu}_3) + 2\mathrm{Cov}(I_1\hat{\mu}_1, I_3\hat{\mu}_3)$$

Since, $I_1+I_2+I_3=1$, for i,j = 1,2 or 3 and $i \neq j$

$$\mathrm{Cov}(I_i\hat{\mu}_i, I_j\hat{\mu}_j) = E(I_i\hat{\mu}_i I_j\hat{\mu}_j) - E(I_i\hat{\mu}_i)E(I_j\hat{\mu}_j)$$

$$= - p_i p_j \mu^2$$

and

$$\mathrm{Var}(I_i\hat{\mu}_i) = E((I_i\hat{\mu}_i)^2) - (E(I_i\hat{\mu}_i))^2$$

$$= E(I_i^2)E(\hat{\mu}_i^2) - p_i^2\mu^2$$

$$= p_i[\mathrm{Var}(\hat{\mu}_i) + \mu^2] - p_i^2\mu^2$$

Hence, the variance of $\hat{\mu}$ becomes

$$\mathrm{Var}(\hat{\mu}) = p_1\mathrm{Var}(\hat{\mu}_1) + p_1\mu^2 - p_1^2\mu^2 + p_2\mathrm{Var}(\hat{\mu}_2) + p_2\mu^2 - p_2^2\mu^2 +$$

$$p_3\mathrm{Var}(\hat{\mu}_3) + p_3\mu^2 - p_3^2\mu^2 - 2p_1p_2\mu^2 - 2p_1p_3\mu^2 - 2p_2p_3\mu^2$$

$$= p_1\mathrm{Var}(\hat{\mu}_1) + p_2\mathrm{Var}(\hat{\mu}_2) + p_3\mathrm{Var}(\hat{\mu}_3)$$

For case I,

$$\mathrm{Var}(\hat{\mu}_1) = \mathrm{Var}\left(\frac{1}{(n+1)} \left(\sum_{i=1}^{n} X_i + X_* \right) \right)$$

$$= \frac{1}{(n+1)^2} \left(n\sigma^2 + \sigma^2 + 2\mathrm{Cov}\left(\sum_{i=1}^{n} X_i, X_* \right) \right)$$

$$= \frac{(n+3)}{(n+1)^2} \sigma^2 \ ,$$

$$\mathrm{Var}(\hat{\mu}_2) = \mathrm{Var}\left(\frac{1}{(n-1)} \left(\sum_{i=1}^{n} X_i - X_* \right) \right)$$

$$= \frac{1}{(n-1)^2} \left(n\sigma^2 + \sigma^2 - 2\mathrm{Cov}\left(\sum_{i=1}^{n} X_i, X_* \right) \right)$$

$$= \frac{\sigma^2}{(n-1)} \ ,$$

$$\mathrm{Var}(\hat{\mu}_3) = \mathrm{Var}\left(\frac{1}{n} \sum_{i=1}^{n} X_i \right)$$

$$= \frac{\sigma^2}{n} \ .$$

Using the relation $p_3 = 1 - p_1 - p_2$,

$$\mathrm{Var}(\hat{\mu}) = p_1\sigma^2 \frac{(n+3)}{(n+1)^2} + p_2 \frac{\sigma^2}{(n-1)} + p_3 \frac{\sigma^2}{n}$$

$$= \sigma^2 \left[p_1 \frac{(n+3)}{(n+1)^2} + p_2 \frac{1}{(n-1)} - \frac{p_1+p_2}{n} \right] + \frac{\sigma^2}{n}$$

The first term in the above equation is the loss in precision due to the proposed strategy for Case I.

As assumed previously, if $p_1=p_2=p$,

$$\mathrm{Var}(\hat{\mu}) = \sigma^2 p \left[\frac{(n+3)}{(n+1)^2} + \frac{1}{(n-1)} - \frac{2}{n} \right] + \frac{\sigma^2}{n}$$

The standard error is

$$\hat{SE}(\mu) = (\sqrt{p(n+3)/(n+1)(n+1) + p/(n-1) - 2p/n + 1/n})*\sigma$$

The term $f(n) = \left[\frac{(n+3)}{(n+1)^2} + \frac{1}{(n-1)} - \frac{2}{n} \right]$ decreases with an increase in the value of n. The loss in precision decreases as n increases. The quantity of interest is $|\hat{SE}(\mu)-\sigma/\sqrt{n}|$. Values of k (substituted for n) and p from tables 3.1, 3.3, and 3.5 may be used to determine the loss in precision due to the proposed strategy. The results summarized in Table 3.6 are for $\sigma^2=1$.

From Table 3.6, it may be concluded that the loss in precision is small when the minimum query set size is large. From Tables 3.1, 3.3, and 3.5, the probability of compromise is also small for larger values of k. It must be pointed out that this study deals with control for exact compromise and not statistical compromise. For larger values of k, or large query set sizes, statistical compromise is very likely.

Table 3.6. Values of n, p, standard error and $1/\sqrt{n}$.

n	p	$\hat{SE}(\mu)$	$1/\sqrt{n}$
2	0.4286	0.8591	0.7071
3	0.4444	0.6526	0.5774
4	0.4545	0.5491	0.5000
5	0.4762	0.4841	0.4472
10	0.4878	0.3302	0.3162
20	0.4884	0.2288	0.2236

For Case II,

$$\mathrm{Var}(\hat{\mu}_1) = \mathrm{Var}\left(\frac{n\mu + X_*}{(n+1)} \right) = \frac{\sigma^2}{(n+1)^2}$$

$$\mathrm{Var}(\hat{\mu}_2) = \mathrm{Var}\left(\frac{n\mu - X_*}{(n-1)} \right) = \frac{\sigma^2}{(n-1)^2}$$

$$\mathrm{Var}(\hat{\mu}_2) = 0$$

Therefore,

$$\mathrm{Var}(\hat{\mu}) = p_1 \frac{\sigma^2}{(n+1)^2} + p_2 \frac{\sigma^2}{(n-1)^2}$$

The above quantity is the penalty for not returning the true value $\bar{X}=\mu$. Again, for the case when $p_1=p_2=p$, the above equation reduces to:

$$\mathrm{Var}(\hat{\mu}) = \frac{p\sigma^2}{(n+1)^2} + \frac{p\sigma^2}{(n-1)^2}$$

From the above equation, it is clear, that the penalty is higher for "large" values of p and σ^2 and "small" values of n. The precision estimates reflect the consequences for potentially adding or deleting records far from the mean. This is reflected in σ^2. Also, the greater the probability of choosing the option to duplicate a record or delete a record, the chances of distorting the data is larger. The distortion in the response is greater if a record is duplicated or deleted for the case when the query set size is "small" than for the case when the query set size is "large". The above equation

quantifies the penalty paid when the proposed method is employed to return query responses.

Chapter IV

IMPLEMENTATION

STRATEGY

A method was proposed in Chapter III to avoid compromise of an individual's confidential information. It was shown that the proposed strategy was effective against trackers. In this chapter, an implementation of the proposed strategy is given.

It is to be recalled that in the proposed scheme, there were three options which may be chosen when responding to a query. It was also pointed out that the same option must be taken for the same query set regardless of how the query is formed; this is referred to as condition 1. Also, for all such queries, a second condition is required. The same record must be duplicated or deleted should the option to duplicate or delete a record be chosen in condition 1. The first condition can easily be implemented if a random number is generated from the same seed from which to select the option. One such seed is the query set size. This guarantees that the same option is chosen for query sets having the same number of records.

To satisfy the second condition, an implementation could be to use the same random number generated above to select the record to be deleted/duplicated. An obvious strategy would be to delete/duplicate a record by position in the query set. This would require that the records come in the same order no matter how the query is created to retrieve the same query set. The order in which records are retrieved depends on the implementation of the database system used. Of

interest were two database systems: INGRES and ORACLE. It seems clear in [STON76] as to the order in which a standard INGRES implementation should return the records in a query set; however, little could be determined about ORACLE's retrieval and query optimization algorithms.

Consequently, the database given in Table 2.1 was created in both INGRES and ORACLE. It was established using this database and other databases that so long as there was only one relation in the database (as in the example of Table 2.1), the records in the query set were always retrieved in the same order for a given query set no matter how the query was formulated (or how the query set was characterized).

Thus, the following implementation is proposed:

(1) Determine the query set size, $|q(C)|$.

(2) If the query set size is not in the range [k,N-k] where k is a chosen parameter and N is the number of records in the database, then the query is invalid.

(3) Use the query set size to seed a random number generator.

(4) The random number generated, r, is used to select one of the three options below:

   (a) Duplicate a record in the query set.

   (b) Delete a record from the query set.

   (c) Do nothing to the query set.

   Let the probabilities of choosing options (a), (b) and (c) be p_1, p_2, and p_3.

(5) If the option chosen is (a) or (b), then the same random number generated in (3) is used to determine the record to be duplicated or deleted. The query set is then modified by duplicating or deleting the record chosen. If the option chosen is (c), the query set is not modified.

(6) Return the modified query set to the user.

EXAMPLES

A program was written in C using embedded EQUEL statements (see Appendix A). The database given in Table 2.1 was used as a sample database. For simplicity, the values of p_1, p_2 and p_3 were chosen to be equal (1/3). This program was used to determine the query responses to the queries given in Chapter II for illustrating trackers. It is assumed that the value of k is appropriately chosen. The examples below show the results obtained.

Example 1 - Individual Trackers

To find the salary of Dodd (identified by the characteristic Sex=F & Dept=CS & Position=Prof) using individual trackers, the formula was:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= SUM(Sex=F; Salary) -

SUM(((Sex=F)&(Dept=CS)&(Position=Prof)); Salary)

The option chosen for the query SUM(Sex=F; Salary) was to delete a record; the record of Flynn was deleted. For the query SUM(((Sex=F) & (Dept=CS) & (Position=Prof)); Salary), the option to delete a record was chosen. The response was obtained by deleting the record of Irons. Thus, an individual trying to calculate Dodd's salary would get:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= 68 - 72

= -4K

Obviously, this is very different from Dodd's actual salary (15K).

Example 2 - General Trackers

Dodd's salary using general trackers could be found from the algebraic manipulation of four queries:

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= SUM(((Sex=F)&(Dept=CS)&(Position=Prof)) + (Sex=M); Salary) +

SUM(((Sex=F)&(Dept=CS)&(Position=Prof)) + (Sex=M); Salary) -

SUM((Sex=M);Salary) - SUM((Sex=M);Salary)

The following result was obtained; the option taken for each of the query is written within parenthesis.

SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)

= 137 (Record of Engel duplicated) + 68 (Record of Flynn deleted)

- 104 (no change) - 68 (Record of Flynn deleted)

= 33K

This response is also not Dodd's salary (15K).

Example 3 - Double Trackers

Dodd's salary was calculated using double trackers in chapter 2 from:

```
SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)
= SUM((Position=Prof);Salary)+
     SUM(((Sex=F)&(Dept=CS)&(Position=Prof))+(Dept=Math);Salary)
     - SUM((Dept=Math);Salary)
     - SUM((Sex=F)&(Dept=CS)&(Position=Prof)&(Dept=Math)&
                    (Position=Prof); Salary)
```

The result obtained was:

```
SUM((Sex=F)&(Dept=CS)&(Position=Prof);Salary)
= 173 (Record of Dodd duplicated)
  + 65 (Record of Hayes deleted)
  - 65 (Record of Hayes deleted)
  - 173 (Record of Dodd duplicated)
= 0
```

For the above examples, the proposed strategy is effective against trackers. In examples 1 and 3, a user would think that he/she has uniquely identified Dodd because if the queries were COUNT queries instead of SUM queries as written above, the result of the algebraic manipulation would give a value of one. In the above examples

however, the user can deduce that the result that is obtained is not correct because it is not likely that Dodd would earn ≤ 0 dollars. It must be pointed out that a statistician is still given close estimates of population means. For example, the average salary of individuals having the characteristic C = (Dept=Math) is $20.75K. The value returned using SUM(Dept=Math;Salary)/COUNT(Dept=Math;Salary), was $21.67K.

The implementation procedure proposed seems very inexpensive as compared to the methods of avoiding compromise presented in chapter II. The procedure requires:

1. Determination of the query set size
2. The generation of the random number r.
3. Modulus procedure used twice.
4. Deletion/duplication.

All these are relatively inexpensive and as shown above, easy to implement.

Chapter V

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

CONCLUSIONS

Release of confidential information of an individual through inference control in statistical databases should be of interest to many. An analysis of the various methods of compromise and some of the techniques used to avoid/deter compromise allowed us to discover that most of the methods were either too expensive to implement or would allow compromise to occur under certain situations.

An inexpensive method to deter compromise using an output perturbation technique has been proposed. In the method, the response to a query is distorted by randomly duplicating a record in the query set, randomly deleting a record in the query set, or returning the true response. In the method proposed, the same record must be deleted/duplicated or subjected to no change to the query set regardless of how the query request for the query set is formed. Statistically, it was shown that the proposed strategy was effective against individual, general and double trackers. A statistical analysis quantified the loss in precision in the output due to the proposed strategy. An implementation of the proposed method was also presented in Chapter IV. The implementation appears to be inexpensive compared to the methods of avoiding compromise as discussed in Chapter II.

RECOMMENDATIONS FOR FUTURE RESEARCH

1. This study proposed a method to thwart the exact disclosure of confidential information of an individual. Additional work needs to be done to avoid disclosure of confidential information about a group of individuals.

2. The implementation proposed, relies on the fact that records are returned in the same order for a genuine query no matter how a query set is formed. For databases having single relations, the database system implementations we examined returned the records in a fixed order. It was found that when there was more than one relation involved in the satisfaction of a query, the order in which records of individuals returned for a query set varied for different queries describing the query set. Additional work needs to be done to optimize the queries so that the records are retrieved faster and in the same order for a query set.

3. Control of compromise by inferential methods is only one aspect of the broader issue of information dissemination control. There is a need to quantify (or measure) the security of computer systems. There may be levels of security, and some may be considered acceptable while others may not.

REFERENCES

[BECK80] Beck, L. L., "A Security Mechanism for Statistical Databases", ACM Transactions on Database Systems, Vol. 5, No. 3, September 1980: 316-338.

[CHIN78] Chin, F.Y., "Security in Statistical Databases for Queries with Small Counts", ACM Transactions on Database Systems, Vol. 3, March 1978: 92-104.

[CHIN82] Chin, F. Y. and Ozsoyoglu, G., "Auditing and Inference Control in Statistical Databases", IEEE Transactions on Software Engineering, Vol. SE-8, No. 6, November 1982: 574-582.

[COX75] Cox, L. H., "Disclosure Analysis and Cell suppression", American Statistical Association, Proceedings of the Social Statistics Section, 1975: 380-382.

[COX77] Cox, L. H., "Suppression methodology in Statistical Disclosure Analysis", American Statistical Association, Proceedings of the Social Statistics Section, 1977: 750-755.

[COX79] Cox, L. H., "Linear Sensitivity measures in Statistical Disclosure Control", American Statistical Association, Proceedings of the Social Statistics Section, 1979: 634-639.

[COX80] Cox, L. H., "Suppression Methodology and Statistical Disclosure Control", Journal of the American Statistical Association, June 1980, Vol. 75, No. 370, Theory and Methods Section, 1980: 377-385.

[DAVI78] Davida, G. I., Linton, D. J., Szelag, C. R., and Wills, D. L., "Database Security", IEEE Transactions on Software Engineering, Vol. SE-4, No. 6, November 1978: 531-533.

[DEMI78] DeMillo, R. A., Dobkin, D., and 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.

[DENN78] Denning, D. E., "Are Statistical Data Bases Secure?", Proceedings of AFIPS, Vol. 47, 1978: 525-530.

[DENN79a] Denning, D. E., Denning, P. J., and Schwartz, M. D., "The Tracker: A Threat to Statistical Database Security", ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979: 76-96.

[DENN79b] Denning, D. E. and Denning, P. J., "Data Security", Computing Surveys, Vol. 11, No. 3, September 1979: 227-249.

[DENN80a] Denning, D. E. and Schlörer, J., "A Fast Procedure For Finding a Tracker in a Statistical Database", ACM Transaction on Database Systems, Vol. 5, No. 1, March 1980: 88-102.

[DENN80b] Denning, D. E., "Secure Statistical Databases with Random Sample Queries", ACM Transactions on Database Systems, Vol. 5, No. 3, September 1980: 291-315.

[DENN81] Denning, D. E., "Restricting Queries That Might Lead to Compromise", IEEE Proceedings of the Symposium on Security and Privacy", 1981: 33-40.

[DOBK79] Dobkin, D., Jones, A. K., and Lipton, R. J., "Secure Databases: Protection Against User Influence", ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979: 97-106.

[FEIG70] Feige, E. L. and Watts, H. W., "Protection of Privacy through Microaggregation", In Databases, Computers and the Social Sciences, R. L. Bisco Ed., Wiley-Interscience, New York, 1970: 261-272.

[FELL72] Fellegi, I. P., "On the Question of Statistical Confidentiality", Journal of the American Statistical Association, Vol. 67, No. 337, March 1972: 7-18.

[FELL74] Fellegi, I. P. and Phillips, J. L. "Statistical Confidentiality: Some Theory and Applications to Data Dissemination", Annals of Economic and Social Measurement, Vol. 3, No. 2, 1974: 399-409.

[FELL77] Fellegi, I. P., "Discussion", American Statistical Association, Proceedings of the Social Statistics Section, 1977: 762-764.

[FRIE80] Friedman, A. D. and Hoffman, L. J., "Towards a Fail-Safe Approach to Secure Databases", IEEE Proceedings of the Symposium on Security and Privacy, 1980: 18-21.

[HAQ75] Haq, M. I. U., "Insuring Individual's Privacy from Statistical Data Base Users", Proceedings of AFIPS, NCC, Vol. 44, 1975: 941-946.

[HOFF70] Hoffman, L. J. and Miller, W. F., "Getting a Personal Dossier from a Statistical Data Bank", Datamation, Vol. 16, No. 5, May 1970: 74-75.

[JABI77] Jabine, T. B., Michael, J. A., and Mugge, R. H., "Federal Agency Practices for Avoiding Statistical Disclosure: Findings and Recommendations"; American Statistical Association, Proceedings of the Social Statistics Section, 1977: 744-749.

[KAMU77] Kam, J. B. and Ullman, J. D., "A Model of Statistical Databases and Their Security", ACM Transactions on Database Systems, Vol. 6, No. 1, March 1981: 95-112.

[KARP70] Karpinski, R. H. , Reply to Hoffman and Shaw, Datamation, Vol. 16, No. 10, Oct. 1970: 11.

[MIRA80] Miranda, S. M., "Aspects of Data Security in General Purpose Data Base Management Systems", IEEE Transactions in Software Engineering, 1980: 46-58.

[NARG72] Nargundkar, M. S. and Saveland, W., "Random Rounding to Prevent Statistical Disclosure", American Statistical Association, Proceedings of the Social Statistics Section, 1972: 382-385.

[REED73] Reed, I. S., "Information Theory and Privacy in Data Banks"; Proceedings of the AFIPS, Vol. 42, 1973: 581-587.

[PALM74] Palme, J. "Software Security", Datamation, Vol. 20, No. 1, January 1974: 51-55.

[SCHL75] Schlörer, J., "Identification and Retrieval of Personal Records from a Statistical Data Bank", Methods Inform. Med., Vol. 14, No. 1, January 1975: 7-13.

[SCHL80] Schlörer, J., "Disclosure from Statistical Databases: Quantitative Aspects of Trackers", ACM Transactions on Database Systems, Vol. 5, No. 4, December 1980: 467-492.

[SCHL81] Schlörer, J., "Security of Statistical Databases: Multidimensional Transformation", ACM Transactions on Database Systems, Vol. 6, No. 1, March 1981: 95-112.

[SCHL83a] Schlörer, J., "Information Loss in Partitioned Statistical Databases", Computer Journal, Vol. 26, No. 3, 1983: 218-223.

[SCHL83b] Schlörer, J. and Denning, D. E., "Protecting Query Based Statistical Output in Multipurpose Database Systems", IFIP, North-Holland Publishing Company, 1983: 37-46.

[SCHL83c] Schlörer, J., "Information Loss in Partitioned Statistical Databases", The Computer Journal, Vol. 26, No. 3, 1983: 218-223.

[SCHW79] Schwartz, M. D., Denning, D. E., and Denning, P. J., "Linear Queries in Statistical Databases", ACM Transactions on Database Systems, Vol. 4, No. 2, June 1979: 156-167.

[STON76] Stonebraker, M., Wong, E., Kreps, P., and Held, G., "The Design and Implementation of INGRES", ACM Transactions on Database Systems, Vol. 1, No. 3, September 1976: 189-222.

[TRAU84] Traub, J. F., Yemini, Y., and Wozniakowski, H., "The Statistical Security of a Statistical Database", ACM Transactions on Database Systems, Vol. 9, No. 5, December 1984: 672-679.

[ULLM82] Ullman, J. D., "Principles of Database Systems", Second Edition, Computer Science Press, 1982.

[YUCH77] Yu, C. T. and Chin, F. Y., "A study on the Protection of Statistical Databases", ACM SIGMOD, Int. Conf. Management of Data, 1977: 169-181.

[ZEIS77] Zeisset, P., "Avoiding Disclosure in the release of microdata", American Statistical Association, Proceedings of the Social Statistics Secttion, 1977: 739-743.

APPENDIX A

A PROGRAM TO IMPLEMENT DISCLOSURE AVOIDANCE

```
/*********************************************************************
  This program is written to respond to queries by modifying
the query sets so that a person will not be able to get an
individual's confidential information.
*********************************************************************/
#define TRUE 1
#define FALSE 0
#include <string.h>
#include <math.h>
main()
{
##    char      name[11];              /* Name of individual          */
##    int       sal;                   /* Salary of individual        */
##    int       KOUNT;                 /* Query set size              */
      int       number_of_records;     /* Number of records in the
                                          scrambled version           */
      char      sex[2];                /* Sex of individual           */
      float     total_sal=0.000;       /* Actual total salary of
                                          individuals in the query
                                          set                         */
      float     scram_total_sal=0.000;/* Scrambled version of
                                          the total salary            */
      float     avg_sal;               /* Average salary              */
      float     scram_avg_sal;         /* Scrambled version of
                                          the average salary          */
      int       index;                 /* Index into the query set
                                          to duplicate/delete a
                                          record                      */
      int       random;                /* Random number generated     */
      int       choice;                /* Option to delete(=2) or
                                          duplicate(=1) or return
                                          true value(=0)              */
      int       i;                     /* Index to scan query set     */
      char      name_changed[11];      /* Name of person whose
                                          record is deleted/added     */
      int       salary_changed=0;      /* Salary of the person whose
                                          record is deleted/added     */
      /*  Initialization                                              */
      strcpy(name_changed , "          ");
##    ingres denning
##    range of p is pay_relation
      /*   Include the query                                          */
#include "wanted.c"
      /*  Find the query set size                                     */
      /*  The query set is stored in dummy                            */
```

```
##      range of d is dummy
##      retrieve(KOUNT=count(d.salary))
        /*  Find a random number and decide which option to choose   */
        srand( (unsigned) KOUNT);
        random = rand();
        choice = random%3;
        if (choice==2)
        {
            /* delete a record */
            index = random%(KOUNT-1);          /* Index into query set */
            number_of_records  = KOUNT-1;      /* No. of records in the
                                                  modified query       */
            /* Delete the record whose index was calculated above     */
            i                  = 0;
##          retrieve(name=d.#name, sal=d.salary)
##          {
                total_sal    = total_sal + sal;
                if (i!=index)
                    scram_total_sal = scram_total_sal + sal;
                else
                {
                    strcpy(name_changed, name);
                    salary_changed = sal;
                }
                i++;
##          }
        }
        else
            if (choice==1)
            {
                /* add a record */
                index = random%(KOUNT-1);        /* Index of record to
                                                    be added           */
                number_of_records = KOUNT+1;     /* No. of records in
                                                    the modified query
                                                    set                */
                /*  Duplicate the record given by index               */
                i                 = 0;
##              retrieve(name=d.#name, sal=d.salary)
##              {
                    total_sal    = total_sal + sal;
                    if (i!=index)
                        scram_total_sal = scram_total_sal + sal;
                    else
                    {
                        strcpy(name_changed, name);
                        salary_changed = sal;
                        scram_total_sal = scram_total_sal + sal*2.0;
                    }
                    i++;
```

```
##              }
             }
             else
             {
                 /* Report true response                                    */
                 number_of_records           = KOUNT;
##               retrieve(name=d.#name, sal=d.salary)
##               (
                     total_sal = total_sal + sal;
##               )
                 scram_total_sal = total_sal;
             }
##       destroy dummy
         /* Calculate the average values (true value) for the salary */
         if (KOUNT != 0)
             avg_sal           = (float) total_sal/(float) KOUNT;
         else
             avg_sal           = 0;
         /* Calculate the average values (scrambled) for the salary  */
         if (number_of_records != 0)
             scram_avg_sal     =     (float)     scram_total_sal/(float)
number_of_records;
         else
             scram_avg_sal = 0;


         /* Print the results                                                */
         printf(" TRUE VERSION \n");
         printf(" Number of records = %d\n",KOUNT);
         printf(" Average salary    = %f\n",avg_sal);
         printf(" Total   salary    = %f\n",total_sal);
         printf("\n");
         printf(" SCRAMBLED VERSION - DELETE / ADD / NO CHANGE \n");
         printf(" Number of records = %d\n",number_of_records);
         printf(" Average salary    = %f\n",scram_avg_sal);
         printf(" Total   salary    = %f\n",scram_total_sal);
         printf("\n");
         if (choice==2)
         {
             printf(" Record corresponding to %s was DELETED.  Salary was
%d\n",
                     name_changed, salary_changed);
             printf("\n");
         }
         else
             if (choice==1)
             {
                printf(" Record  corresponding  to %s was ADDED.  Salary
was %d\n",
                        name_changed, salary_changed);
```

```
            printf("\n");
        }
        else
            printf(" NO CHANGE in reporting the query.\n\n");
    printf("\n");
    printf("*****************************************\n");
    printf("\n\n");
}
```

A NEW DETERRENT TO COMPROMISE OF CONFIDENTIAL INFORMATION FROM STATISTICAL DATABASES

by

NANDA KAUSHIK

B.Tech., Indian Institute of Technology, New Delhi, INDIA, 1981

M.Sc., Deakin University, Waurn Ponds, AUSTRALIA, 1986

M.S., Kansas State University, Manhattan, KS, 1986

Ph.D., Kansas State University, Manhattan, KS, 1988

AN ABSTRACT OF MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

Kansas State University

Manhattan, Kansas

1988

A NEW DETERRENT TO COMPROMISE OF CONFIDENTIAL INFORMATION FROM STATISTICAL DATABASES

ABSTRACT

Release of confidential information of an individual through inference control in statistical databases should be of interest to many. An analysis of the various methods of compromise and some of the techniques used to avoid/deter compromise allowed us to discover that most of the methods were either too expensive to implement or would allow compromise to occur under certain situations.

An inexpensive method to deter compromise using an output perturbation technique has been proposed. In the method, the response to a query is distorted by randomly duplicating a record in the query set, randomly deleting a record in the query set, or returning the true response. In the method proposed, the same record must be deleted/duplicated or subjected to no change to the query set regardless of how the query request for the query set is formed. Statistically, it was shown that the proposed strategy was effective against individual, general and double trackers. A statistical analysis quantified the loss in precision in the output due to the proposed strategy. An implementation of the proposed method was also presented. The implementation appears to be inexpensive compared to the methods of avoiding compromise presented in the literature.