Comparative Study of Two Inference Control Techniques

Susan Harford

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

April 1984

© Susan Harford, 1984

ABSTRACT

Comparative Study of Two Inference Control Techniques

Susan Harford

This thesis compares two inference control techniques, partitioning (Glaser, 1982) and random sampling (Denning, 1980). In particular, the hierarchical partitioning algorithm and response strategy developed by David Glaser is examined. A comparative measure for determining the goodness of a partition is defined. An extension to Glaser's response strategy is presented, rendering a strategy free from information loss. The theory behind a possible method of compromising random sampled databases is introduced.

ACKNOWLEDGEMENTS

I would like to acknowledge the tremendous support I received from Dr. Alagar during the development of this thesis. His guidance was invaluable and I am grateful to have had the opportunity to work with him. I am also grateful to Dr. Alagar for giving me the confidence needed to see this thesis to completion.

To my Dad.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Database Security

The area of research in data security encompasses different aspects of a database environment. Denning [13] classifies these areas into four categories: access control, flow control, data encryption, and inference control. The first of these, access control, refers to the kinds of techniques employed in verifying the validity of a user in a given situation. General access to any computer facility could be made dependent upon some initial identification scheme; a password, a fingerprint, or a voice pattern without which access to the computer would be denied. Another problem which a good access control method should handle is that of memory access. Without such a control, a program might overwrite areas of memory either accidentally or intentionally resulting in destruction of information.

Flow control is best described using a hierarchy of user types. The higher a user is on the scale, the more privileged he is as far as data manipulation is concerned. A user can make information available to a user at a higher level, but never to a user at a lower level than himself. In this way, data may be classified as top secret, at the highest level, down to general information at the lowest, corresponding to the hierarchy of user types.

If confidential information is stored on a removable medium such as magnetic tape or hard disk or must be sent in a communication channel from one location to another, then that information should be encrypted so that, should it be stolen or intercepted the culprit would not be able to decode the information to suit his purposes. Data encryption can accomodate many security aspects besides that of hiding information. Some of these include authentication of the author and prevention of unwanted data insertion.

Sometimes, it is necessary to generate a report on a population where group information is required but individual information must not be revealed. Databases containing this kind of information, called statistical databases, must make use of some security system to prevent a user from inferring confidential information. Inference happens when a user combines his preknowledge of the database contents with group information in querying the system to gain information about a particular record in the database. Many inference control techniques have been suggested and the next section gives a historical overview of some of these techniques.

1.2 Review of Research in Statistical Database Security

Hoffman and Miller [23] were the first to realize the dangers involved in releasing statistics about the confidential records stored in a database. For example, compromise is possible when a combination of the values from some of the attributes in the database uniquely identifies a record. This combination of values actually becomes the identifying feature of the record even though the usual identification; name, social insurance number, etc., have been removed. Say a user's preknowledge includes 1) the fact that there exists a record in the database describing the characteristics of an individual x, and 2) the nature of a subset of those characteristics. Let us suppose that x is a female analyst who is married and the user knows this information. Then, if the answer to the query "How many records satisfy the characteristics: female, analyst, and married?" is 1, this is sufficient information to infer the rest of the characteristics of x within the scope of the database. The response to the query "How many married female analysts earn < $20,00?" must be either 0 or 1. With the knowledge gained from the previous response it can be concluded that: if the response is 0 then x earns at least $20,000, otherwise it can be inferred that x makes less than $20,000.

A reasonable approach to avoiding this kind of breach of security would be to refuse to answer queries when the response will be based on a small group of records. If t is some threshold setting the size of the smallest groups of records upon which a response will be formulated, then queries specifying less than t or more than N-t records (N is the total number of records in the database) will not be answered. Queries specifying a large set of records are included in the set of restricted queries to prevent answering the complement of a query specifying a small set of records. This form of inference control was the first to be suggested [23].

Unfortunately, a database protected this way can be totally compromised with the use of a tracker [14]. Trackers are reviewed in more detail in chapter 5, but an example here will illustrate the point.

Let the query "How many married female analysts are there?" be unanswerable due to the fact that there are too few records satisfying it, and let the query "How many part-timers are there?" specify the required number of records to elicit a reponse from the system. If the system is able to respond to query 1 : "How many married female analysts or part-timers are there?" and to query 2 : "How many married female analysts or non part-timers are there?". Then, the number of married female analysts can be calculated using the formula : response to query 1 + response to query 2 - total number of records in the database. Note that the total number of records is the response to "How many part-timers are there?" + the response to "How many non part-timers are there?".

The remaining inference control techniques to be discussed can be catagorized into two groups as either restriction techniques or perturbation techniques. Restriction techniques aim to prevent a user from receiving a response to a query which is considered sensitive. For simplicity, a sensitive query will be defined as in [18]: a query whose characteristic formula matches either one or no record in the database. These techniques include attribute combination based restriction, implied query restriction, and query restriction based on querying history. Perturbation techniques add noise to the information revealed to the user in an attempt to hide the confidential aspects of the system. Systematic and random rounding, range response, partitioning and random sampling fall into this catagory.

Several attribute combination based query restriction techniques are outlined in Denning and Schlorer [18]. The simplest form of this method is to set a threshold d and restrict any query whose characteristic formula specifies more than d attributes. If d = 3, then the query asking for records satisfying the formula married female analyst with 2 years employment specifies 4 attributes: sex, profession, marital status, and years of employment. This query would be restricted. Presumably, the greater the number of attributes specified in a formula, the less chance there is that a record will satisfy it indicating that these queries would probably be sensitive. The problem is, if four attributes are specified in a formula, each of which has a small domain size, then the probability that the query is sensitive is smaller than for a combination of four attributes whose domain sizes are large.

It follows that the relative density of the database should come into play when predicting whether or not a query is sensitive. Relative density refers to the density of the database with respect to the attributes specified in the query being judged. The relative density = S_m/N where S_m is the product of the domain sizes of the m attributes specified in the characteristic formula and N is the number of records in the database. The query is judged sensitive, and therefore restricted, if $S_m/N > 1/k$ for some preset k. If the number of different professions defined in the database is 5 and there are 100 records stored, then the query asking for female analyst would have a relative density of 10/100. For $k > 10$ this query would be restricted.

Implied query restriction, also discussed in [18], deals with determining what can be calculated from the response to a given query along with other information obtainable from the database. If it is determined that sensitive information can be obtained in this way, then the query is restricted. In order to ensure that no sensitive information can be calculated from the response to the query, the worst case requires that the result of combining the information contained in the response with every combination of responses to all other unrestricted queries must be considered. This is too costly a method to be considered practical.

Auditing is an approach based on keeping a log of the information revealed to a user during a session. If the information seems to be leading a user into a situation where compromise is possible, the user is refused further access to the database, or at least, a subset of queries determined by the query log becomes restricted. Unfortunately, it is not feasible to expect the system to keep track of a user's history over many sessions (considering the possible number of valid use2s whose history would be stored needlessly). It is also impossible to prevent a group of users from combing their information. Hoffman and Miller [23] and Chin and Ozsoyoglu [8] have researched this area.

The major difficulty in developing a good restriction technique is keeping the information loss to an acceptable minimum. The function of a statistical database is to provide the user with information. If this information is denied to a user whose intention is not that of compromising the system, then the database loses its functionality quickly.

A different approach to statistical database security can get around this problem of restricting information. A method of giving perturbed responses in place of exact responses should suffice in securing the system. As long as the perturbation factor is controlled, the accuracy of statistics can be determined and the error kept to a minimum, ensuring the usefulness of the statistics produced. At the same time, because the responses will not be exact, trackers should have no effect on compromising the security of the database.

Data perturbation could be based on perturbing information at the level of the response as in rounding and range response, at the query level as in random sampling and partitioning or at the database level as in error inoculation and data swapping.

Rounding a statistic [24] requires a rounding base, say b. The perturbed response is a function of b. In the case of systematic rounding, the statistic is rounded to the nearest integer multiple of b. Random rounding rounds the statistic up or down to an integer multiple of b where the decision to round up or down is made at random. The problem with these methods is one of consistency. The following example with b = 5, using systematic rounding will demonstrate this.

Suppose there are 48 analysts in the database, 26 of which are female, 22 of which are male. Since an analyst must be male or female, the number of female analysts + the number of male analysts must be equal to the number of analysts. The rounded responses given to queries asking for the number of analysts, female analysts, and male analysts would be 50, 25, and 20 respectively. The inconsistency becomes apparent when it is observed that 25 + 20 = 45 $\neq$ 50.

Alagar [4] solves the inconsistency problem by suggesting that the range [kb,(k+1)b-1] be given as the response where k satisfies the condition that the true response lies within this range. Using the same example which demonstrated inconsistency in rounding, consistency will be illustrated for range response. The responses given to the query asking for the number of analysts would be [45,50], that for female analysts [25,30] and that for male analysts [20,25]. Without going into the details of range arithmetic, (this is done in Chapter 5), it is enough to say [25,30] + [20,25] = [45,50]. Hence the system is consistent.

Error inoculation, proposed by Campbell [5] replaces the true data in the database with false data. This is not desirable because in the event that recalling the true data becomes necessary it would not be possible

Data swapping, discussed in Dalenius and Reiss [10] suffers from the lack of a means of swapping the data while maintaining the integrity of the database.

A good data-perturbing technique must be consistent in the perturbing factor. That is, if a query is asked repeatedly, the same response must be returned each time; otherwise, the possibility of averaging out the pertubation factor will emerge.

One way of achieving this is to base the response to a query on a set of records satisfying the query. Partitioning and random sampling both satisfy this condition. These two techniques have been chosen as the focus of attention for this thesis pin pointing some problem areas associated with random sampling and disputing some of the criticisms leveled against partitioning.

1.3 Objective of this thesis

The objective of this thesis is to compare the performance of two inference control techniques: Denning's random sampling [16] and Glaser's hierarchical partitioning [22]. Although random sampling seems to be a fool proof protection system, there are two factors which indicate otherwise. There is some information loss for small query sets and there is a threat of compromise through range reduction. Partitioning on the other hand has been criticised for 1) the lack of a measure of the goodness of a given partition with respect to security and accuracy and 2) the amount of information loss present. Both these drawbacks to the use of partitioned databases can be overcome.

These issues concerning random sampling and partitioning will form the basis of the thesis presented here.

CHAPTER 2

OUTLINE OF METHODS

2.1 Introduction

The two methods to be evaluated will be presented in this chapter. Glaser's partitioning is discussed in Section 2.2 and includes the hierarchical algorithm used for creating a partition from a collection of records and a good response strategy appropriate for the type of resulting partition. Section 2.3 covers Denning's random sampling and gives a specific implementation of the theory.

Both techniques were implemented using randomly generated pseudo databases. A database was represented using a file of records of integers. Each field in a record represented an attribute. The domain of each attribute consisted of a range of integers, [1,d] where d is the domain size of that attribute. The integer in the ith field of a record indicated which value of the ith attribute that record contained.

An example to illustrate this will convert a small database into a pseudo database of the form described above.

Let the database environment consist of three attributes: Sex, Age, and Marital Status. Let {Male, Female}, {0-14, 15-29, 30-44, 45-59, 60-74, 75-89}, and {Single, Married, Divorced, Separated, Widowed} be the values of the respective attributes. The database contains the following records:

Record	Sex	Age	Marital Status
1	Male	15-29	Married
2	Female	0-14	Single
3	Female	75-89	Widowed
4	Male	60-74	Separated

The pseudo database corresponding to this example would create a mapping from the attributes to the integers; i.e. Sex → 1, Age → 2, Marital Status → 3, and from the values into the integers; Male→ 1, Female → 2, 0-14 → 1, 15-29 → 2, 30-44 → 3, 45-59→ 4, 60-74→ 5, 75-89→ 6, Single → 1, Married → 2, Divorced → 3, Separated → 4, and Widowed → 5. Then the records of the database would correspond in a one-one fashion to the following records:

Record	Att 1	Att 2	Att 3
1	1	2	2
2	2	1	1
3	2	6	5
4	1	5	4

To access the pseudo database, pseudo formulas were formulated. A pseudo formula consisted of operators and operands. The three logical operators NOT, OR, and AND were represented by the 3 digit sequences 000, 001, 002 repectively. The operands were alse represented by 3 bit sequences where the 1st bit specified the attribute desired and the last 2 bits, the value. Given operand x, trunc(x/100) yielded the attribute while x mod 10 yielded the value. Given x, a 3 bit integer, the following rule suffices to distinguish between an operand and an operator; if trunc(x/100) = 0 then x = an operator, otherwise x = an operand.

Take for example the formula "Sex = Female AND Marital Status = Single". The corresponding pseudo formula based on the sample database above would be 102 002 301 and the set of records satisfying it would be {2}.

From this point on, pseudo databases will be refered to as simply databases and pseudo formulas as formulas.

2.2 Glaser's Hierarchical Partitioning and Response Strategy

Unlike most partitioning schemes which try to merge records together to form groups, this hierarchical scheme starts with one group containing all records and then tries to split this group into smaller groups. Essentially, it is a top down approach. The basic algorithm actually processes the databases three times, the 2nd and 3rd passes hopefully improving the situation but not necessarily so.

2.2.1 Partitioning Algorithm

All the records in the databases are initialized to one group and this group is repeatedly split in an attempt to maximize the number of groups. The process of splitting can best be represented using a tree. Consider this initial group as the root of the tree. Splitting takes place at a node according to the values of a given attribute. Either a split is successful, in which case the records of the node being split are distributed accordingly among the new nodes, or it is not, in which case the next attribute is tried. The actual splitting requires that d number of sons of this node are created, where d is the domain size of the attribute being used to carry out the split. Distribution of records among the son nodes occurs depending upon the value of that attribute present in any record previously assigned to the node being split. A split is successful if each of the resulting d nodes contains at least t records. Recall that t is the threshold such that any collection of less than the threshold number of records does not qualify as a group. After all attributes have been tried, the first pass is complete.

The second pass is similar to first pass with the exception that the initial node, instead of consisting of all records, consists of a subset of the records. The subset is obtained by collecting the records from the groups resulting from pass 1 which have at least 2t records. This is because, a group of size 2t or more is considered large and therefore undesirable. Pass 2 may result in splits which create more groups and hence smaller groups. If pass 2 splitting does not produce any refinement of the original partition obtained in pass 1 then the initial partition is kept as the partition which then undergoes the third pass

Finally, there may still be large groups present in the partition and it is the job of the final pass to attempt to eliminate them. At this point each large group is processed separately. The records of a large group are distributed on the values of each attribute. If M is the number of attributes, there will be M different ways of splitting up the records of a large group. For each way, adjacent records are merged until groups can be formed. The split which causes the most groups and the most uniformly sized groups is kept.

To illustrate the algorithm consider the database shown in figure(2.1) with 20 records and 4 attributes with domain sizes of $A_1 = 2$, $A_2 = 3$, $A_3 = 4$, and $A_4 = 3$.

The partition resulting from applying the hierarchical partitioning algorithm to this database, with a threshold of 2, is obtained as follows.

Pass 1: Order the attributes on descending order of domain size; A_3, A_2, A_4, A_1. Splitting on A_3 results in one group containing one record only, record 11. Trying A_2 we get $g_1 = \{4,6,13,19\}$, $g_2 = \{3,8,10,14,15,17\}$, and $g_3 = \{1,2,5,7,9,11,12,16,18,20\}$. Splitting each group on A_4 yields one successful split, on g_3. Now the partition is: $g_1 = \{4,6,13,19\}$, $g_2 = \{3,8,10,14,15,17\}$, $g_3 = \{1,5\}$, $g_4 = \{7,9,11,12,16,18\}$, and $g_5 = \{2,20\}$. A split on any of these groups using A_1 will be unsuccessful.

Pass 2: Group together the big groups whose size exceed 4. i.e., g_1, g_2, g_4. Reorder the attributes placing the unused attributes first; A_3, A_1, A_2, A_4. Splitting on A_3 is unsuccessful. Splitting on A_1 gives the new partition $g_1 = \{4,6,7,9,11,12,13,14,18\}$, $g_2 = \{3,8,10,15,16,17,19\}$, $g_3 = \{1,5\}$, and $g_4 = \{2,20\}$. This partition has fewer groups then the one previously found in pass 1; therefore it is disregarded. Splits

Figure 2.1 Sample Database

Record	Att 1	Att 2	Att 3	Att 4
1	1	3	3	1
2	1	3	4	3
3	2	2	3	2
4	1	1	3	3
5	1	3	2	1
6	1	1	2	3
7	1	3	3	2
8	2	2	2	1
9	1	3	3	2
10	2	2	4	1
11	1	3	1	2
12	1	3	3	2
13	1	1	2	2
14	1	2	3	1
15	2	2	3	2
16	2	3	4	2
17	2	2	4	2
18	1	3	3	2
19	2	1	2	3
20	2	3	2	3

on A_2 and A_4 are not successful.

Pass 3: Pass 3 will process each big group formed in pass 1 separately starting with $g_2 = \{3,8,10,14,15,17\}$. The distribution of the values of the records of g_2 looks like this:

	A_1	A_2	A_3	A_4
v_1	1	0	0	3
v_2	5	6	1	3
v_3		0	3	0
v_4			2	

Combining adjacent cells to eliminate cells with less than t records we get:

A_1	A_2	A_3	A_4
6	6	4	3
		2	3

The distribution of values over the 4th attribute is the best, i.e., it produced the most groups with the smallest size variation. Therefore, g_2 is broken down into two groups $\{8,10,14\}$ and $\{3,15,17\}$.

Similarily for $g_4 = \{7,9,11,12,16,18\}$, the distribution of values for the 4 attributes is:

	A_1	A_2	A_3	A_4
v_1	5	0	1	0
v_2	1	0	0	6
v_3		6	4	0
v_4			1	

Combining adjacent cells to eliminate the cells with less than 2 records results in the following:

A_1	A_2	A_3	A_4
6	6	6	6

in this case, the records {7,9,11,12,16,18} remain as a group. The partitin resulting from pass 3 is g_1 = {1,5}, g_2 = {2,20}, g_3 = {4,6,13,19}, g_4 = {8,10,14}, g_5 = {3,15,17}, and g_6 = {7,9,11,12,16,18}. Since this partition is superior to the one resulting from pass 1 alone, (there are more groups) this becomes the partition returned.

The cost of this algorithm, as derived in Glaser[22], is the cost for pass 1 + the cost for pass 2 + the cost for pass 3. In the worst case, the cost for all three passes is O(MN) where M is the number of attributes and N is the number of records. Therefore the total cost is O(MN).

Although this will be covered in detail in Chapter 3, it is worth noting here that intuitively speaking, the partitions obtained by the hierarchical approach are good. That is, the combination of passes 1, 2 and 3 works hard to maximize the number of groups output while the very nature of the splitting maximizes the similarity of records whithin a group with respect to those attributes upon which a split was successful.

2.2.2 Response Strategy

It is important to note at this point that a response strategy for a partitioned database does not depend on the partioning algorithm used to form the groups. The strategy considers only the groupings themselves. A response is initiated when a query is posed. A query is of the form "function(logical formula, data attribute)". In our case, function will be limited to Freq (for frequency) and Avg (for average). C will stand for any logical formula, and the data attribute is absent when the function is Freq. The average age of married males from the sample database above would be represented as Avg(Sex = Male AND Marital Status = Married, Age). The query set refers to the set of records satisfying C.

The following definitions will be used in describing the response strategy for frequency and average queries.

Let G_i be the i^{th} group, n_i the number of records in the i^{th} group, c_i the number of records in the i^{th} group which satisfy C, s the number of groups in the partition, r the number of groups which contain at least one record from the query set and A_{ij} the average value in the jth field over all records in G_i.

Let Pfreq(C) be the perturbed frequency given in response to a query of the form Freq(C) and Tfreq(C) be the true frequency. We define Pfreq(C) = $(\sum_{i=1}^{r} c_i) / (\sum_{i=1}^{r} n_i) * (r/s)$. It is proved in [22] that Pfreq(C) is close to Tfreq(C) when the variation in group sizes is small. Since the hierarchical partitioning yields groups with small group size variation, this response strategy would produce good results if used with a database which was partitioned in this manner.

Let Pavg(C,j) be the perturbed response corresponding to Avg(C,j) and Tavg(C,j) be the true response. Glaser defines Pavg(C,j) to be $\sum_{i=1}^{r} ((c_i / \sum_{l=1}^{r} c_l) * A_{ij})$. In general, Pavg(C,j) is close to Tavg(C,j) when the records in a group are similar. For example, if the n_i records in G_i contain the same value for the jth attribute, (in other words, the records are identical with respect to the jth attribute,) then A_{ij} which is the average over all records in G_i would be equal to the average over the c_i records in G_i which satisfy C. If the same was true for all G_i, $i = 1$ to r, then Pavg(C,j) would equal Tavg(C,j).

This again is compatible with the hierarchically partitioned database from the stand-point of accurate responses in that the records of a group are similar. The strategy which produces the partition groups the records together according to the distribution of the values of the attributes used to split.

2.3 Denning's Random Sampling

The second method of securing a statistical database to be discussed is presented in this section. This method takes a completely different approach to the problem from that of the partitioning methods. Partitioning is based on grouping records of the database together so that the response to a query is obtained from a superset of the query set. Random sampling on the other hand bases the response on a subset of the query set.

2.3.1 Sampling algorithm

Let X_C be the query set corresponding to the logical formula C, and let X_C^* be the random sample taken from X_C. X_C^* consists of a subset of the records in x_C, chosen at random. Similarily let n_C be the size of X_C and n_C^* the size of X_C^*. Define a matching function f : (formula, record) $\rightarrow \{1,0\}$ such that $f(C,i) = 1$ implies record i is chosen, and $f(C,i) = 0$ implies record i is not chosen, and define p to be the probability that $f(C,i) = 1$.

Random sampling was implemented in the following way; see [16] for full details

1. Define a mapping R, from a record i of the database to the bit sequence $\{0,1\}^*$ of length m.

2. Define a mapping G from a logical formula C to the sequence $\{0,1,*\}^*$ of length m such that $k \leq m$ is the number of bits and $m - k$ is the number of *'s.

3. Define a match between i and C if, given that a * is a 0 or a 1, R(i) = G(C). For example, if R(i) = 1 0 0 0 1 and G(C) = 1 * 0 0 * then i matches C.

Choose i for the sample if i does not match C, i.e., f(C,i) = 0 if i matches C and 1 otherwise. Because there are k bits in G(C), the probability of a match is $(\frac{1}{2})^k$. Therefore the probability that a record is chosen to be included in the response set is $1 - (\frac{1}{2})^k$.

The algorithm for converting a record into a bit sequence is straightforward. Create a 2 dimensional table Rmap with one row for each record and m columns. Generate N random bit sequences of length m and store them in Rmap. Given record i, R(i) = i^th row of Rmap.

Finding G(C) is more involved. Given C, process C as follows:

1. Convert the 3 bit operands to m bit random sequences.

2. Process C applying the operations to the m bit operands to produce an m bit sequence representing C.

3. Place k *'s in this binarized C at random.

2.3.2 Response Strategy

Once X_C^* has been found, the reponse is formulated based on the records contained in X_C^*. Again, only frequency and average queries will be considered.

The perturbed frequency is given by the equation Pfreq(C) = n_C^* / pN.

The perturbed average of C with data attribute j is $(1/n_C^*)$ * $\sum_{i \epsilon X_C^*} v_{ij}$ where v_{ij} is the value in the j th field of the i th record.

For queries with small query sets, there is a good chance that $X_C^* = X_C$. For this reason, Denning adds the condition to the response strategy that if n_C falls below a certain threshold, the response is restricted. We also assume that the user is aware of the values of p and N for the random sampled database being used. This is necessary to give the user an idea of the amount of noise present in the statistics produced.

CHAPTER 3

GOODNESS OF A PARTITIONING STRATEGY

3.1 Introduction

A criticism of partitioning as a means of securing a statistical database stems from the fact that "good" is a relative term. How can a good partition be distinguished from a bad partition? By Glaser's definition, a good partition is one which satisfies two conditions. The first is that the groups of a good partition are as numerous as possible, maintaining that each group is at least as big as a predetermined threshold, where the variation in group size is small. The second condition is that the records in each group are as similar as possible. The justification for this can be seen in terms of the accuracy of responses desired, given that the response to a query is determined by all the records in the groups containing those records which satisfy the query. The proportion of records involved in a perturbed response which did not belong to the original query set will be small if the groups themselves are small (and therefore numerous). Every one of these records is contained in a group which contains at least one record from the query set, and if the second condition is met, this record will be similar to the rest of the records in the group and hence to those in the query set. At the extreme, if a query set is the union of a subset of groups of a partition, it is obvious that the statistic will not be perturbed at all, yielding 100%

accuracy. Fortunately, the chance of this happening is small, as this would defeat the objective of injecting noise into the information revealed to the user. Even in the event that this did happen, the user would not be aware of it and so could not use the information to compromise the system. On the other hand, the noise is kept to a minimum ensuring the usefullness of the database.

3.2 Entropy

The very nature of the two conditions leads to the idea that the goodness of a partition can be measured using the concept of entropy. Given a random variable X with p_i the probability of event x_i occurring, and the number of events is s, then the entropy of X is defined as $H(X) = -\sum_{i=1}^{s} p_i \log_2 p_i$. An important characteristic of entropy is that it is proportional to the number of events involved as well as to the similarities among the probabilities of the events occurring. In this case, H(X) is proportional to both s and the similarity of the s p_i's. These characteristics form the basis of the following section which uses the concept of entropy in comparing partitions.

3.3 Partition Entropy

Two entropy measusres have been defined, corresponding to the two conditions mentioned above, which together can determine whether or not one partition is better than another. The partition entropy, $PE_p(X)$, is equal to $-\sum_{i=1}^{s} p_i \log_2 p_i$, where X is the random variable for which the events are the groups of partition p, s is the number of groups in the partition, and $p_i = |n_i|/N$, ($|n_i|$ = size of group i), is the probability of finding a record in group i. $PE_p(X)$ is maximum for a fixed s when $p_i = p_j$ for all $i \neq j$, indicating that all the groups are the same size. $PE_p(X)$ is also proportional to s. Therefore $PE_p(X)$ is maximum when $p_i = p_j$ for all i not equal to j and s approaches N/t, where N is the number of records in the database and t is the threshold.

For example, given a database with N = 100, t = 3, and three partitions as follows:

1) Partition p such that s = 20, all groups are of size 5, $PE_p(X) = 4.32$,

2). partition p' such that s = 20, 10 groups are of size 3, 5 of size 6, and 5 of size 8, $PE_{p'}(X) = 4.19$,

3) Partition p" such that s = 33, 32 groups are of size 3, 1 of size 4, $PE_{p''}(X) = 5.04$.

The above entropy measures are absolute quatities. In order to compare two different partitions which may have different values for N and t, we can normalize the absolute entropies using the maximum entropies for a given N and t. For our example of N = 100 and t = 3, the maximum number of groups possible would be 33, with all but one group consisting of 3 records and one consisting of 4 records. If we call this "best" partition B, then $PE_b(X) = 5.04$. So the normalized entropies for p, p', and p" are: $PE_{np}(X) = .857$, $PE_{np'}(X) = .831$, and $PE_{np''}(X) = 1.000$.

Comparing p and p', $PE_p(X) > PE_{p'}(X)$. This is to be expected since for a fixed s of 20, the records in p are equally distributed among the groups; the records in p' are not.

Comparing p" with p, $PE_{p"}(X) > PE_p(X)$. The reason for this is also obvious since not only are the records evenly distributed in p", but s is maximum.

The above conclusions remain the same if we compare the normalized entropies.

3.4 Group Entropy

The second condition necessary for a good partition is that the records in a group are as similar as possible. Even though the $PE_p(X)$ of a partition is maximum, if the similarity of records in the groups is small, this situation could lead to inaccurate responses. The group entropy, $GE_{j,y}(X)$, where X is the random variable representing the distribution of the values of the jth attribute within a group y is $-\sum_{k=1}^{d_j} f_{jk}/F_j \log_2 f_{jk}/F_j$ where d_j is the domain size of attribute j, F_j is the frequency of records in group y with some value of j, (i.e. F_j is the size of the group since every record has exactly one value in its jth field), and f_{jk} = frequency of records with the kth value of attribute j. The similarity of records in a group y for a specific attribute j is inversely proportional to $GE_{j,y}(X)$.

For example, let the distribution of values of the first attribute for a group y containing 5 records be such that there are 5 values in the domain of attribute 1 and each record in the group has a different value in the field representing attribute 1. Then $f_{1k} = 1$ for all $k = 1..5$, and $F_1 = 5$. Therefore $GE_{1,y}(X) = 2.32$ for this group.

Now let the distribution of values of the first attribute in a group y' with 5 records be such that each record contains value 2 in the field representing attribute 1. In this case, $f_{1k} = 0$ for k not equal to 2 and 5 for $k = 2$ and $F_j = 5$. Therefore $GE_{1,y'}(X) = 0$ for this second group.

It is known that the records in y' are more similar than the records in y with respect to the first attribute and this is verified by the fact that $GE_{1,y'}(X) < GE_{1,y}(X)$. Averaged over all groups and over all attributes to come up with the average group entropy, this measure can be used to determine the superior partition.

The relative group entropy can be calculated similar to that of the relative partition entropy. The maximum group entropy for a given group may be obtained with respect to each attribute j by equating the f_{jk}'s for all $k = 1..d_j$. This would give an indication of the group entropy for the worst group. The results could then be normalized as with the partition entropy to obtain the relative messure.

To sum up, for two partitions p and p', if the partition entropy of p is larger than that of p', this indicates that the number of groups in p is larger and their sizes vary less than in p'. If the average group entropy is smaller for p then this implies that the similarity of records in a group of p, on the average, is greater than that of a group in p'. The conclusion is that p is a better partition than p'.

To verify the results, we consider Yu and Chin's [30] partitioning method, known to be inferior to Glaser's hierarchical method. See Glaser [22].

3.5 Yu and Chin's Rectangular Partitioning Method

The partitioning algorithm is as follows:

1) Take the first two attributes and determine a grid with the first attributes = y axis and the second attribute = x axis.

2) Fill grid according to the values of the records in those two fields, i.e., the pair of values represents the coordinates of the grid into which the record is placed. Let size(i,j) = number of records in cell(i,j).

3) Partition the grid scanning from left to right, top to bottom. For every cell with (1 <= size(cell) <= threshold) do; look right 1 cell, look down 1 cell, look left 1 cell, look up 1 cell, look right 2 cells ... until enough cells are found such that size(old cell) + size(new cells) >= threshold.

4) Whenever a non empty cell is found, if it is to the right (under), the old cell, move everything in each cell in the column (row) of the old cell to the column (row) of the new cell.

Otherwise, if new cell is to the left (above), move contents of the column (row) of new cell to column (row) of old cell. Eliminate the column (row) from which records were moved by setting size(cell) = -1 for all cells in column (row) to be eliminated.

5) Scan grid again and mark each non empty cell. At this point a cell is non empty if size(cell) $\geq$ threshold. Every marked cell corresponds to a group. The collection of marked cells make up the partition.

6) Repeat the process until all attributes have been used, modifying the algorithm slightly: instead of associating an attribute with the y axis, the groups of the most recent partition are used, each group representing a row of the grid. The next attribute to be used becomes the x axis.

Data was collected for two different partitioning methods, Glaser's hierarchical and Yu and Chin's rectangular, varying the distribution of data values, the sizes of the attribute domains, the threshold, and the number of records.

3.6 Results

Results for M = 5, Domain Sizes = 2,4,5,5,5, Distribution = Uniform, Threshold = 2, and varying N.

Partition Entropy

N	Glaser's	Yu and Chin's
100	5.0053	4.3405
350	6.8385	5.9044
500	7.4521	6.3870

Average Group Entropy

N	Glaser's	Yu and Chin's
100	.4390	.7019
350	.3171	.4889
500	.2508	.4243

Results for M = 5, Domain Sizes = 2,4,5,5,5, Threshold = 2, N = 500, and varying Distribution.

Partition Entropy

Distribution	Glaser's	Yu and Chin's
Uniform	7.4521	6.3870
Normal	7.1554	5.6088
Exponential	4.0461	4.0461

Average Group Entropy

Distribution	Glaser's	Yu and Chin's
Uniform	.2508	.4243
Normal	.1551	.4361
Exponential	.5304	.5304

Results for M = 5, Domain Sizes = 2,4,5,5,5, Distribution = Uniform, N = 500, and varying Threshold.

Partition Entropy

Threshold	Glaser's	Yu and Chin's
2	7.4521	6.3870
3	6.9049	6.1155
5	6.0608	5.4851

Average Group Entropy

Threshold	Glaser's	Yu and Chin's
2	.2508	.4243
3	.3689	.5704
5	.5860	.8198

Results for M = 5, Threshold = 2, Distribution = Uniform, N = 500, and varying Domain Sizes.

Partition Entropy

Domain Sizes	Glaser's	Yu and Chin's
5,5,5,5,5	7.2511	6.8189
2,4,5,5,5	7.4521	6.3870
2,2,3,3,3	6.5167	5.8987

Average Group Entropy

Domain Sizes	Glaser's	Yu and Chin's
5,5,5,5,5	.4340	.5618
2,4,5,5,5	.2508	.4243
2,2,3,3,3	.0049	.1032

It can be seen from the results that the partition entropy is greater and the average group entropy is smaller for Glaser's partition in each case. From this we conclude that Glaser's method is better than Yu and Chin's with respect to the number of groups and size variation as well as to the similarity of records within a group. These results are consistent with our expectations, i.e., we know that Glaser's partition is superior to Yu and Chin's and we expect that the partition with the larger partition entropy and smaller average group entropy value would be the superior group.

CHAPTER 4

GOODNESS OF RESPONSE STRATEGY IN PARTITIONED DATABASES

4.1 Information Loss

The amount of information denied to a user as a result of the inference protection measures employed in a statistical database environment is the definition given to information loss by Schlorer [29]. This is something to be considered when evaluating any response strategy. A good response strategy should have little or no information loss. Information loss can be measured in terms of characteristic formulas. With every formula is associated a set of records. The records in the set are related to each other on some subset of attributes. These sets relay information to the user about the contents of the database. Restricting a query implies restricting the amount of information available to the user, i.e., information is lost.

4.2 Schlorer's Response Strategy

Schlorer's response strategy is based on two rules: query set restriction and query attribute restriction. Query set restriction ensures that the true response is returned only when the set of records satisfying a formula C is a union of 1 or more groups. Otherwise no response is given. In other words a query will not be responded to if there exists a group g such that $|g \cap X_C| \neq 0$ and $|g - X_C| \neq 0$, where X_C is the set of records satisfying C. Note that since $|g \cap \{\}| = 0$, queries with $X_C = \{\}$

are answered despite the risk of negative disclosure.

The justification for this rule is that queries with query sets of size 1 are restricted, since the partitioning strategy stipulates that all groups are at least as big as some threshold > 1. These particular queries are the ones which are singled out as possibly leading to compromise due to the threat of tracker attacks.

Query attribute restriction prevents an attribute not used in partitioning the database from being used in a formula C. For example, if Age, an attribute of a database, was not used to partition that database then a query such as "How many records satisfy Age = 15-29" would be restricted.

An elementary k-set is defined as the set of records corresponding to a characteristic formula which specifies values for k different attributes. For example, C = (Status = Student) corresponds to an elementary 1-set.

Given the above approach to responding, Schlorer [29] demonstrates that the amount of information loss is high. That is, according to his definition of information loss, the number of restricted queries whose formulas specifying elementary k-sets, for k = 0 .. M, is large. Here, M is the number of attributes used in the partitioning. To demonstrate this consider the following.

There exists a relationship between the number of elementary M-set queries whose query sets are of size one (which are themselves restricted) and the number of elementary 1-set queries which are restricted. An elementary 1-set is not necessarily empty and the information corresponding to these sets may not lead to compromise. The restriction of these queries therefore does not enhance the security of the system, however, due to the rigidity of the response strategy, information is lost unnecessarily.

One way of attempting to correct this situation is to increase the density of the database in such a way that there are fewer M-sets of size one. Given $S_M = \prod_{i=1}^{M} d_i$, d_i is the size of the i^{th} domain, N is the number of records in the database, s_M/N is inversely proportional to the density of the database. The density is inversely proportional to the number of elementary M-sets of size one. Therefore, if a small s_M/N, i.e., a denser database, can be achieved, this should lead to fewer elementary M-sets of size 1 and hence less information loss.

One way of decreasing S_M/N is to reduce the size of S_M; choose less attributes on which to partition. But this leads to still more information loss, not due to query restriction, but rather to the query formulation property that no attribute can be used in a query if it is not used in partitioning, even if the resulting query does not lead to compromise.

The other obvious way to decrease S_M/N is to increase N by introducing dummy records. As Schlorer points out, although it is possible to inject these dummy records in such a way that the error in the resulting average query response has zero bias, the same is not true for responses to frequency queries.

In other words, a significant amount of information loss is bound to be present when a user is operating under Schlorer's response strategy. The drawbacks are two fold; while information not contributing to compromise is inhibited, queries which could lead to compromise, i.e., queries with empty query sets are not. A user can obtain negative disclosure because the system does respond to these queries.

Three things should be included in a new response strategy: 1) true responses should not be given to queries of size 0 or 1 in order to prevent negative as well as positive compromise, 2) all queries should be responded to, whether the response is the true response or a perturbed one, and 3) all attributes should be allowed to be used in a characteristic formula.

Schlorer's response strategy for a partitioned database is based on two rules: 1) an answer to a query is given only when the query set size is a union of groups, and 2) only attributes used in the partitioning can be queried on. These rules cause a significant amount of information loss when the strategy is applied to a hierarchically partitioned database. This is because the only queries which will result in a query set being the union of groups are those which specify attributes for which a successful split occured for every son present at the time of the split. The chance of this happening is small and therefore the proportion of queries available to the user is also limited. Also, the hierarchical partitoning strategy does not guarantee that all attributes will be used in the partitioning process, so the number of attributes allowed on which to query could be small.

4.3 Extending Glaser's Response Strategy

Glaser's response strategy, outlined in Chapter 2, results in less information loss where only queries with set sizes of 0 are unanswerable. It is the aim here to establish an extension to Glaser's response strategy which will take into account queries with empty query sets. If a strategy can be found which provides a perturbed response to such queries while preserving the accuracy of the system, then the revised strategy will be completely free from information loss.

Two kinds of queries could result in an empty set of records satisfying it. One type is the invalid query. An invalid query is one which violates the database integrity rules. For example a query which asks for information about "males who are pregnant" in most applications would be considered an invalid query, and there would be no records satisfying it. A strategy is needed to distinguish between valid and invalid queries after it is found that the query set is empty. It is enough to check only those queries which are found to have empty query sets because if the query set is not empty then it follows that the query is valid. When an invalid query is detected, instead of supplying the user with a pertrubed statistic, as this could leave the user with serious doubts as to the reliability of the system, the system will respond with " query not meaningful ". This is equivalent to letting the user know that there are no records satisfying the query. Because a response is given, there is no information loss in the sense of query restriction. At the same time though, the security of the system is not jeopardised because a user should know the scope of the database and intuitively be able to recognize an invlaid query by himself. Information is not gained in terms of compromising sensitive information.

The second type of empty set query is one which is meaningful despite the fact that the query set size is 0. These are a subset of the valid queries. In order to eliminate information loss all together, valid queries with 0 set size should be responded to, but because $\sum_{i=1}^{r} c_i$, $\sum_{i=1}^{r} n_i$, and r all = 0, Glaser's usual response strategies used for non empty valid queries is not applicable. The next section gives some sample databases. For each, a list of integrity rules were formed to demonstrate that the ability exists to detect invalid queries. Section 4.3.2 assumes this ability to distinguish between invalid and valid queries and proceedes to present strategies for responding to valid queries with empty query sets.

4.3.1 Detecting Invalid Queries

The approach taken here is based on semantic integrity rule enforcement. A query is invalid if it breaks at least one of the integrity rules associated with the database. An integrity rule has the form: if expression 1 then expression 2. Any query which asks for the intersection of records satisfying expression 1 with those satisfying the compliment of expression 2 is invalid. A rule preventing a query asking for all pregnant males could be if Sex = Male then Status ≠ Pregnant.

The following examples list possible integrity rules for three different applications; a student database, a company personnel database, and a hospital database. The environments are described by the attributes and corresponding values.

STUDENT DATABASE

Attribute	Value
Status	Full Part Indep
Level	Ugrad Diploma Masters Phd
Year	1,2, ... ,5
Credits Earned	3,6,9, ... ,150
Credits Registered For	3,6, ... ,30
Faculty	Arts & Sci Eng Comm

Major	Math Comp Sci Elec Eng Build Sci Manag Account Finance
Sex	Male Female
Age	20,30, ... ,60
Citizenship	Cdn Landed Imm Other
Payroll	Yes No
Salary	4,000, 7,000, 10,000, ... 16,000, Null
Minor	Math Comp Sci Marketing Null

INTEGRITY RULES FOR STUDENT DATABASE

Rule 1 :	If Status = Part then Credits Registered For $\leq$ 6
Rule 2 :	if Status = Full then Credits Registered For $\geq$ 9
Rule 3 :	If Faculty = Arts & Sci then Major = Math
Rule 4 :	If Faculty = Eng then Major in (Comp Sci, Build Eng, Elec Eng)
Rule 5 :	If Faculty = Comm then Major in (Manag, Account, Finance)
Rule 6 :	If Major = Math then Minor $\neq$ Math
Rule 7 :	If Major = Comp Sci then Minor $\neq$ Comp Sci
Rule 8 :	If Citizenship = Other then Status = Full
Rule 9 :	If Payroll = No then Salary = Null
Rule 10 :	If Level $\neq$ Ugrad then Minor = Null

COMPANY PERSONNEL DATABASE

Attribute	Value
Department	Sales Computing Shipping/Receiving Accounts Personnel
Position	Manager Salesman Programmer Analyst Shipper Receiver Accountant Receptionist Tranee Operator Secretary
Personnel Category	Part/Temp Part/Perm Full/Temp Full/Perm
Years of Service	1 ... 50
Yearly Salary	7,000 ... 100,000
Commission	Null, 1,000, ... 50,000

INTEGRITY RULES FOR COMPANY PERSONNEL DATABASE

Rule 1 :	If Department = Sales then Position in (Manager, Salesman, Secretary)
Rule 2 :	If Department = Computing then Position in (Programmer, Analyst, Tranee, Operator)
Rule 3 :	If Department = Shipping/Receiving then Position in (Shipper, Receiver)
Rule 4 :	If Department = Accounts then Position in (Accountant, Secretary)
Rule 5 :	If Department = Personnel then Position in (Manager, Receptionist, Secretary)
Rule 6 :	If Position = Tranee then Years of Service < 1
Rule 7 :	If Position ≠ Salesman then Commission = Null
Rule 8 :	If Position = Manager then Salary ≥ 50,000
Rule 9 :	If Position = Tranee then Salary = 7,000

Rule 10 : If Personnel Catagory = (Part/Temp or Part/Perm) then Salary < 10,000
Rule 11 : If Position = Receptionist then Salary < 15,000
Rule 12 : If Position ≠ Manager then Salary < 50,000

HOSPITAL POPULATION DATABASE

Attribute	Value
Status	Doctor Patient Staff
Specifics	Cardiology Neurology Gynecology Cardiac Neural Psychiatric Urology Pediatrics Nurse Orderly Intern Intensive Care Recovery Maternity
Sex	Male Female
Age	0 ... 100

INTEGRITY RULES FOR HOSPITAL DATABASE

Rule 1 : If Status = Doctor then Specifics in (Cardiology, Neurology, Gynecology, Psychiatry, Pediatrics)
Rule 2 : If Status = Doctor then Age $\geq$ 25
Rule 3 : If Status = Patient then Specifics in (Cardiac, Neural, Psyciatric, Intensive Care, Maternity, Recovery)
Rule 4 : If Status = Staff then Specifics in (Nurse, Orderly, Intern, Receptionist, Operator, Lab Technichian)
Rule 5 : If Status = Staff then Age $\geq$ 18
Rule 6 : If Status = Patient and Specifics = Maternity or Gynecology then Sex = Female
Rule 7 : If Status = Patient and Specifics = Pediatrics then Age $\leq$ 13

4.3.2 Responding To Valid Empty Set Queries

The response strategies presented will be limited to responses to frequency queries and responses to average queries.

Given s = # of groups in the partition, and N/s = average group size, order the sizes of groups such that $n_i > n_{i-1}$, where n_i = size of i^{th} group. Choose smallest n_i such that $(s * n_i) >$ N, that is, choose the smallest group size greater than the average. Now let $n = n_i$ for this i. For all frequency queries with 0 set size, the perturbed frequency will be 1/(n*s) setting $\sum_{i=1}^{r} c_i$ and r to 1. Let C be any query whose query set size is exactly 1, and the record satisfying C is contained in a group of size n. Then, from the response strategy, Pfreq(C) = (1/n * 1/s). But this is also the perturbed frequency for any query of size 0. In this way 1/(s * n) will be the perturbed statistic for two different classes of queries. This duplication is enough to discourage the user from inferring that the true count of the formula C is 0 when the perturbed frequency Pfreq(C) is 1/(p * s).

The absolute error in this response is $|1/(n * s) - 0| = 1/(n * s) \sim 1/N$ which is small when N is large.

Responding to average queries is more difficult to do without changing the meaning of the query. The strategy is based on modifying the query using the nature of the attributes present in the query.

Query structure: The query to be modified has the structure Avg(C,D) where Avg = average, C = characteristic formula or set of records satisfying a characteristic formula which in this case is the empty set, and D is the data attribute over which the average will be calculated.

For example, the average salary of female PHD students would be written as Avg(Sex = Female and Level = PHD,Salary).

Furthermore, for the time being, operators present in C will be limited to the single operator "AND". Queries including "NOT" and "OR" operators will be considered later. In general, C can be represented as $\{A_1 = v_{j_1}$ AND $A_2 = v_{j_2}$ AND ... AND $A_k = v_{j_k}\}$. Note that from the integrity constraints, since C contains only the AND operator every A_i specified in C is specified only once.

Dependency matrix: Although any attribute may be specified in C, only a subset of the attributes of the system qualify as candidates for D. Call the attributes of this subset the data attributes.

The dependency matrix is formed by concatenating the attributes with the data attributes. A cell in the dependency matrix corresponds to an (attribute,data attribute) pair, (A,D) pair for short. Each cell can have one of three values, no, yes, or partial, depending on the relationship between the attribute and the data attribute involved. Dep(A,D) = no implies that for any possible record, the value in the D^{tn} field and the value in the A^{th} field will be independent for all values of A. Dep(A,D) = yes implies that the value in the D^{th} field of a record depends on the value in the A^{th} field. Dep(A,D) = partial implies that the value in the D^{th} field of a record determines a proper subset of the values of A which can occur in the A^{th} field of that record without changing the meaning of that record with respect to D.

Table 4.1 is an example of a dependency table for the Student Database where the data attributes are Credits Earned, Credits Registered For, Age, and Salary.

The information contained in the dependency table is vital when modifying a query in order to ensure that the new query is semantically similar to the original query.

Similarity operator: Consider A, an attribute specified in C such that v is the value of A specified and D is the data attribute. Let ~ be an operator such that if v_A is a value in the domain of A, $v_A \neq v$, then $v_A \sim v$ if C = (A = v) and C' = (A = v or A = v_A) describe the same relative environment with respect to D. i.e. we expect that the true answer to the query Avg(C,D) would be close or equal to the true answer to Avg(C',D) if both C and C' were not empty set queries. Call ~ the similarity operator.

Partition Function: For all pairs (A,D) such that dep(A,D) = partial, Parition(A,D) defines a partitioning of the values of A

Table 4.1 Dependency Table; Student Database

	Credits Earned	Credits Registered For	Age	Salary
Status	no	yes	no	yes
Level	no	yes	yes	yes
Year	yes	no	yes	yes
Credits Earned	-	no	yes	yes
Credits Registered For	no	-	no	no
Major	no	no	no	partial
Minor	no	no	no	no
Faculty	no	no	no	yes
Sex	no	no	no	no
Age	yes	no	-	no
Citizenship	no	yes	no	no
Payroll	no	no	no	yes
Salary	yes	no	no	-

into disjoint groups, $\{g_i\}$, such that if v_1 and v_2 are in the domain of A and are elements of the same group of partition(A,D) then $v_1 \sim v_2$ with respect to D. In other words the set of values making up the domain of A are partitioned into groups according to the $\sim$ operator whenever dep(A,D) = partial.

The following example of a partition list for the Student Database illustrates this. From Table 4.1, it is true that (Major,salary) = partial; this is the only (A,D) pair in the table with this property. Define Partition(Major,Salary).g_1 = {Math}, Partition(Major,Salary).g_2 = {Comp Sci, Elec Eng, Build Sci}, and Partition(Major,Salary).g_3 = {Manag, Account, Finance} where (Comp Sci ~ Elec Eng ~ Build Sci) and (Manag ~ Account ~ Finance) and $\{g_1,g_2,g_3\}$ partition attribute Major.

The interpretation of this is that the value in the salary field is partitally dependent on the value in the major field. The assumption here is that the salary of two students in the same faculty, say Eng, have a good chance of being equal despite the fact that their respective majors, say Comp Sci and Build Sci may differ.

Attribute set structure: Refering back to the structure of C, let $Z_C = \{A_1, A_2, \ldots, A_k\}$ be the set of all attributes specified in C. For each data attribute D, Z_C can be partitioned into three groups; Z_CI, the set of attributes A_i such that dep(A_i,D) = no, Z_CII, the set of attributes A_j such that dep(A_j,D) = partial, and Z_CIII, the set of attributes A_k such that dep(A_k,D) = yes. Note that $Z_C = Z_C\text{I} + Z_C\text{II} + Z_C\text{III}$.

Again the Student Database will be used to make this point clear. The attributes are catagorized first for the data attribute Salary and second for the data attribute Age.

Without considering any particular formula C, the attributes are catagorized as follows for data attribute = Salary. ZI type attributes = {Credits Registered For, Minor, Sex, Citizenship, Age}, ZII type attributes = {Major}, ZIII type attributes = {Status, Level, Year, Credits Earned, Faculty, Payroll}.

A different data attribute would specify a different breakdown of the attributes. For example, the classification with respect to Age is ZI type attributes = {Status, Credits Registered For, Major, Minor, Faculty, Sex, Citizenship, Payroll, Salary}, ZII type attributes = {}, and ZIII type attributes = {Level, Year, Credits Earned}. For some formula C and data attribute D $Z_C \cap ZI = Z_CI$, $Z_C \cap ZII = Z_CII$, and $Z_C \cap ZIII = Z_CIII$.

Query types: the set of possible queries of the form specified above can be catagorized into two classes; class1 = queries with $Z_CI \cup Z_CII \neq \{\}$ and class2 = queries with $Z_CI \cup Z_CII = \{\}$. Define "extend C to include v of A_i" as follows: if C = (... AND $A_i = v_{j_i}$ AND ...) then after extending C to include v of A_i we get C = (... AND ($A_i = v_{j_i}$ OR A_i = v) AND ...). Extending C to include every value of A_i is equivalent to dropping A_i from C.

For example, given C = (Year = 1 and Status = Indep) where A_i = Status then v_{j_i} = Indep, extending C to include v = Part yields C' = (Year = 1 AND (Status = Indep OR Part)). Extending C over all values of status gives C" = (Year = 1 AND (Status = Indep OR

Part OR Full)) = all year 1 students. Therefore C" is the same as (Year = 1).

Modifying class I queries: let Avg(C,D) be a class I query. $X_C = \{\}$ and $Z_C I \cup Z_C II \neq \{\}$. Find PAvg(C,D), the perturbed average, based on a set of k modified queries Avg(C_i',D) where the C and C_i' are semantically similar for D, for all i = 1 to k, k = $|Z_C I \cup Z_C II|$. C_i' is obtained from C by dropping A_i from C if A_i is in $Z_C I$ or extending C to include all v in Partition(A_i,D).g (where v_{j_i} is in Partition (A_i,D).g) when A_i is in $Z_C II$. PAvg(C,D) = $\sum_{i=1}^{k}$ Avg(C_i',D)/k. Let C = (Citizen = Can AND Major = Comp Sci, D = Salary). $Z_C I$ = {Citizen}, $Z_C II$ = {Major}. Avg(C_1',D) = average (Major = Comp Sci,Salary), Avg(C_2',D) = average (Citizen = Cdn AND (Major = Comp Sci OR Major = Elec Eng OR ... OR Major = Build Sci),Salary). PAvg(C,D) = [Avg(C_1',D) + Avg(C_2',D)]/2.

Modifying class 2 queries: now let Avg(C,D) be a class 2 query. That is $Z_C I \cup Z_C II = \{\}$. Modifying C by extending or dropping any attribute in Z_C will, in this case, affect the meaning of the query. Therefore, to average out the error over all the attributes in Z_C, the following scheme is used: drop each attribute in Z_C one at a time to obtain k different C_i' again assuming there are k attributes in Z_C. Avg(C_i',D) is calculated and PAvg(C,D) = $\sum_{i=1}^{k}$ Avg(C_i',d) / k as for class 1 queries. The difference between the perturbed response of class 1 queries and that of class 2 queries is that the class 1 response gives an indication of reality whereas the class 2 response is calculated in order to come up with a number based on the attributes present in the query, but does not necessarily give any information about reality.

If at any stage in calculating PAvg for either class of query, a C_i' is produced which is empty then the class of C_i' is determined and a series of C_i'' are produced according to the appropriate algorithm, until a PAvg(C_i',D) is found. This value is used in place of Avg(C_i',D).

An example of a class 1 query is Avg (Sex = Female AND Major = Comp Sci AND Status = Full,Salary). $Z_C = \{$Sex, Major, Status$\}$, $Z_CI = \{$Sex$\}$, $Z_CII = \{$Major$\}$, and $Z_CIII = \{$status$\}$. Since $\{Z_CI \cup Z_CII\} \neq \{\}$, C is a class I query.

The modification of Avg(C,D) is done as follows. C_1 = (Major = Comp Sci and Status = Full). Note that Sex is dropped from C. C_2 = (Sex = Female AND (Major = Comp Sci OR Elec Eng OR Build Sci) AND Status = Full). Note that Comp Sci ~ $\{$Elec Eng, Build Sci$\}$ because Partition(Major,Salary).g_2 = $\{$Comp Sci, Elec Eng, Build Sci$\}$. Therefore Avg(C,D) = (Avg(C_1,D) + Avg(C_2,D))/2.

An example of a class 2 query is Avg(C,D) where D = Age and C = (Level = Masters and Year = 2). $Z_C = \{$Level, Year$\}$, $Z_CI = \{\}$, $Z_CII = \{\}$, and $Z_CIII = \{$Level, Year$\}$. Since $\{Z_CI \cup Z_CII\} = \{\}$ this is a type II query.

The modified query is found as follows. C_1 = (Year = 2) and C_2 = (Level = Masters). Therefor Avg(C,D) = (Avg(C_1,D) +

Avg$(C_2$,D))/2.

This takes care of queries with C of the form ($A_1 = v_{j_1}$ AND ... AND $A_k = v_{j_k}$). The final issue to be considered is allowing for OR and NOT operators. This requires some normalization procedures to reduce C to its disjunctive normal form and replace all NOT operations with an appropriate sequence of OR operations. C in this form can be expressed as C_1 OR C_2 OR ... OR C_k where C_i, i = 1 to k is a conjuction of attribute values of the form considered initially. C is empty iff C_i is empty for all i = 1 to k. Modifying any C_i would be sufficient to force a non empty query, but for the sake of consistency each C_i must be modified. PAvg(C,D) = $\sum_{i=1}^{k}$ PAvg(C_i,D)/k.

Analysis: the time to find Z_CI, Z_CII, Z_CIII is constant for each element of Z_C. To find C' the worst case happens when Z_CII = Z_C. This is because although modfying using members of Z_CI or Z_CIII requires constant time, modifying using members of ZII does not. For all A in Z_CII, Partition(A,D) must be accessed and then the group g found such that v is in g where v is the value specified by A in C. The worst case for finding g happens when every value of A is checked before the group which contains v is found. This implies a sequential search. If this is the case for all A in Z_C then the work to find C' would be = $|Z_C|$ * maxdomain + $|Z_C|$ * work to calculate PAvg(C',D) by Glaser's response strategy.

4.4 Probability That an Empty Query is Valid

The extra work required to respond to valid empty queries can be justified if their probability of occurrence is small. This probability can be calculated for a given database as follows.

Define a cell as an M-tuple, where the ith entry in the M-tuple is some value from the domain of the ith attribute. Given d_i, the size of the domain of the ith attribute, there are $\prod_{i=1}^{M} d_i$ unique M-tuples or cells. Every record in the database specifies exactly one value for each attribute and therefore can be associated with a single cell. Consider a cell non empty if at least one record is associated with it and empty otherwise.

Every formula defines a union of one or more cells. For example, let Z_C = set of attributes specified by formula C. If C is of the form $A_1 = v_{i_1}$ AND $A_2 = v_{i_2}$ ... AND $A_M = v_{i_M}$ then call C an elementary formula with $|Z_C| = M$. Corresponding to this elementary query is the M-tuple $(v_{i_1}, v_{i_2}, \ldots v_{i_M})$, i.e. C specifies one cell.

Not all formula have $|Z_C| = M$. The number of cells specified by any formula C is $(\prod_{i=1}^{M} d_i / \prod_{A_j \varepsilon Z_C} d_j) * \prod_{A_j \varepsilon Z_C} |V_{C,j}|$ where $V_{C,j}$ is the set of values of the jth attribute OR'ed in C. If C has only the "AND" operator then the number of cells specified by C = $(\prod_{i=1}^{M} d_i / \prod_{A_j \varepsilon Z_C} d_j)$ since $|V_{C,j}| = 1$ for all A_j in Z_C.

Let p be the number of cells specified by C, p_1 the number of valid cells, and p_2 the number of invalid cells. Then $p = p_1 + p_2$. Prob[C = empty | C is valid] = (Prob[cell = empty | cell is valid])^{p_1} * (Prob[cell = empty | cell is invalid]) ^{p_2}. It is known that Prob[cell = empty | cell is invalid] = 1 because by definition, no records can satisfy an invalid M-tuple and therefore Prob[C=empty | C is valid] = (Prob[cell = empty | cell is valid]) ^{p_1}. It is now necessary to find Prob[cell = empty | cell is valid] and p_1.

4.4.1 Prob[cell = empty | cell Is valid]

Let σ = total number of cells, i.e. $\sigma = \prod_{i=1}^{M} a_i$. $\sigma 1$ = total number of valid cells, $\sigma 2$ = total number of invalid cells; $\sigma = \sigma 1 + \sigma 2$.

Prob[cell = empty | cell is valid] = # empty valid cells / # valid cells. Let $f(\sigma 1)$ = # of empty valid cells. $f(\sigma 1)$ = # valid cells - # non empty valid cells. If δ = # non empty valid cells, then δ can be found using the algorithm

```
1. $\delta$ := 0;
2. For all records $r_i$, i=1..N do
       if $r_i \neq r_j$ for all j = 1 to i-1
           then $\delta = \delta + 1$.
```

the notation r_i refers to the M-tuple representing record i. Therefore $1 \leq \delta \leq N$ and δ depends on both the density and the distribution of values in the database.

To find $\sigma 1$, first find $\sigma 2$; the number of invalid cells, using the principle of inclusion on the set of integrity rules. With every integrity rule I, there corresponds an invalid formula Q which covers all possible violations of that rule. Let E be a function which takes any formula and returns the set of cells associated with that formula.

$$\sigma 2 = \sum_{i=1}^{r} |E(Q_i)| - \sum_{1 \leq i_1 < i_2 \leq r} |E(Q_{i_1} \text{ AND } Q_{i_2})| + \ldots (-1)^{r+1} |E(Q_1 \text{ AND } Q_2 \text{ AND } \ldots \text{ AND } Q_r)|$$

where r = the number of integrity rules. As was noted earlier,

$$|E(Q)| = (\prod_{i=1}^{M} d_i / \prod_{A_j \varepsilon Z_Q} d_j) * \prod_{A_j \varepsilon Z_Q} |V_{Q,j}|.$$

To find $|E(Q_i \text{ AND } Q_j)|$, note that if A_k is in Z_{Q_i} and A_k is in Z_{Q_j}, A_k being the k^th attribute, and $\{V_{Q_i,k} \cap V_{Q_j,k}\} = \{\}$ then $E(Q_i \text{ AND } Q_j) = \{\}$. If $\{V_{Q_i,k} \cap V_{Q_j,k}\} \neq \{\}$ then $|V_{Q_i} \cap Q_j,k| = |V_{Q_i,k} \cap V_{Q_j,k}|$

The same is true for the intersection of more then two sets of cells.

From σ and $\sigma 2$, $\sigma 1$ is found to be $\sigma - \sigma 2$. From this, $f(\sigma 1) = \sigma 1 - \delta$ can be determined and Prob[cell = empty| cell is valid] = $f(\sigma 1)/\sigma 1$ is solved. It remains to find p_1.

First p_2 will be found by the princple of inclusion and exclusion similar to the way that $\sigma 2$ was found. For any query Q corresponding to an integrity rule I, Q AND C specifies a subset of the cells that C alone specifies. More specifically, Q AND C specifies a subset of the p_2 invalid cells that C specifies. If

E(C) = the set of invalid cells specified by C, then E(C) = Union E(C AND Q_i) for i = 1 to r. Therefore $p_2 = |E(C)| = \sum_{i=1}^{r} |E(C \text{ AND } Q_i)| - \sum_{1 \le i_1 < i_2 \le r} |E(C \text{ AND } Q_{i_1} \text{ AND } Q_{i_2})| \ldots (-1)^{r+1} |E(C \text{ AND } Q_1 \text{ AND } Q_2 \text{ AND } \ldots \text{ AND } Q_r)|$. Hence, $p_1 = p - p_2$.

The probability that C is empty given C is valid is $((\sigma 1 - \delta)/\sigma 1)^{p_1}$.

For example, let C = (A_1 = v_1 OR v_3 OR v_5) AND (A_2 = v_2 OR v_3) AND (A_3 = v_2)), M = 5, $d_1 = 6$, $d_2 = 3$, $d_3 = 5$, $d_4 = 2$, $d_5 = 2$. $Z_C = (A_1, A_2, A_3)$, $V_{C,1} = (v_1, v_3, v_5)$, $V_{C,2} = (v_2, v_3)$, $V_{C,3} = (v_2)$. Therefore $|E(C)| = (\prod_{i=1}^{5} d_i / \prod_{j=1,2,3} d_j) * \prod_{j=1,2,3} V_{C,j}$ = [(6*3*5*2*2)/(6*3*5)] * (3*2*1) = 24. Let Q_1 = (A_1 = v_1 OR v_3), Q_2 = ((A_2 = v_3) AND (A_1 = v_4)), and Q_3 = ((A_1 = v_1 OR v_4) AND (A_3 = v_3)). Then $|E(Q_1 \text{ AND } Q_2)| = 0$ since $V_{Q_1,1} = \{v_1, v_3\}$ and $V_{Q_2,1} = \{v_4\}$ and $|V_{Q_1,1} \cap V_{Q_2,1}| = 0$. Similarily $|E(Q_1 \text{ AND } Q_3)| = (\prod_{i=1}^{5} d_i / \prod_{j=1,3} d_j) * V_{Q_1 \text{ AND } Q_3, j}$. Note that $V_{Q_1 \text{ AND } Q_3, 1} = \{v_1\}$. Therefore $|E(Q_1 \text{ AND } Q_3)|$ = ((6*3*5*2*2)/(6*5)) = 12.

Consider the following subset of the student database with a subset of rules and corresponding queries.

DATABASE

#	Attribute	#	Value	domain size
1	Status	1	Full	3
		2	Part	
		3	Indep	
2	Credits Registered For	1	3	10
		2	6	
		...	...	
		10	30	
3	Major	1	Math	7
		2	Comp Sci	
		3	Elec Eng	
		4	Build Sci	
		5	Manag	
		6	Account	
		7	Finance	
4	Faculty	1	Arts&Sci	3
		2	Eng	
		3	Comm	
5	Sex	1	Male	2
		2	Female	
6	Citizenship	1	Can	3
		2	Landed Im	
		3	Other	

RULES

Rule 1 : If status = Part then Credits Registered For $\leq$ 6.

Q_1 : (Status = Part) AND (Credits Registered For = 9 OR 12 OR ... OR 30).

$Z_{Q_1} = \{A_1, A_2\}$

$V_{Q_1,1} = \{v_2\}$

$V_{Q_1,2} = \{v_3, v_4, \ldots\ v_{10}\}$

Therefore $|E(Q_1)| = (7*3*2*3)(8)$

Rule 2 : If Status = Full then Credits Registered For $\geq$ 9

Q_2 : (Status = Full) AND (Credits Registered For = 3 OR 6)

$Z_{Q_2} = \{A_1, A_2\}$

$V_{Q_2,1} = \{v_1\}$

$V_{Q_2,2} = \{v_1, v_2\}$

Therefore $|E(Q_2)| = (7*3*2*3)(2)$

Rule 3 : If Faculty = Eng then Major in (Comp Sci, Elec Eng, Build Sci)

Q_3 : (Faculty = Eng) AND (Major = Math OR Manag OR Finance OR Account)

$Z_{Q_3} = \{A_4, A_3\}$

$V_{Q_3,3} = \{v_1, v_5, v_6, v_7\}$

$V_{Q_3,4} = \{v_2\}$

Therefore $|E(Q_2)| = (3*10*2*3)(4)$

Rule 4 : If Faculty = Arts & Sci then Major = Math

Q_4 : (Faculty = Arts & Sci) AND (Major = Comp Sci OR Elec Eng OR Build Sci OR Manag OR Account OR Finance)

$Z_{Q_4} = \{A_3, A_4\}$

$V_{Q_4,3} = \{v_2, v_3, v_4, v_5, v_6, v_7\}$

$V_{Q_4,4} = \{v_1\}$

Therefore $|E(Q_4)| = (3*10*2*3)(6)$

Rule 5 : If Faculty = Comm then Major = (Manage or Finance or Account

Q_5 : (Faculty = Comm) and (Major = Math or Comp Sci or Elec Eng or Build Sci)

$Z_{Q_5} = |\{A_3, A_4\}$

$V_{Q_5,3} = \{v_3\}$

$V_{Q_5,4} = \{v_1, v_2, v_3, v_4\}$

Therefore $|E(Q_5)|$ (3*10*2*3)(4)

Calculating $|E(Q_i \text{ AND } Q_j)|$ we get $E(Q_1 \text{ AND } Q_2) = \{\}$ since $A_2 \epsilon Z_{Q_1}$ and $A_2 \epsilon Z_{Q_2}$ and $(V_{Q_1,2} \cap V_{Q_2,2}) = \{\}$. For $E(Q_1 \text{ AND } Q_3)$ we see that Q_1 AND Q_3 = (Status = Part) AND (Credits Registered For = 9 OR 12 OR ... OR 30) AND (Faculty = Eng) AND (Major = Math OR Manag OR Finance OR Account). $Z_{Q_1 \text{ AND } Q_3} = \{A_1, A_2, A_3, A_4\}$. $V_{Q_1 \text{ AND } Q_3,1} = \{v_2\}$, $V_{Q_1 \text{ AND } Q_3,2} = \{v_3, v_4 \ldots v_{10}\}$, $V_{Q_1 \text{ AND } Q_3,3} = \{v_1, v_5, v_6, v_7\}$, and $V_{Q_1 \text{ AND } Q_3,4} = \{v_2\}$. Therefore $|E(Q_1 \text{ AND } Q_3)| = (2*3)(8*4) = 192$.

In this way all sets $E(Q_i$ AND Q_j AND ...) can be found yielding $\sigma 2$ and $\sigma 1 = \prod_{i=1}^{M} d_i - \sigma 2$. To find p_2 take the intersection of each Q_i and C to find p_2, the number of invalid cells specified by C. If C = (Sex = Female) AND (Faculty = Eng) then C AND Q_1 = (Sex = Female) AND (Faculty = Eng) AND (Status = Part) AND (Credits Registered For = 9 OR 12 OR ... OR 30). $Z_{Q_1} = \{A_1, A_2\}$, $Z_C = \{A_4, A_5\}$ so $Z_{Q_1 \text{ AND } C} = \{A_1, A_2, A_4, A_5\}$. $V_{Q_1 \text{ AND } C,1} = \{v_2\}$, $V_{Q_1 \text{ AND } C,2} = \{v_3 \ldots v_{10}\}$, $V_{Q_1 \text{ AND } C,4} = \{v_2\}$, and $V_{Q_1 \text{ AND } C,5} = \{v_1\}$. Therefore the number of cells specified by Q_1 AND C = $(\prod_{i=1}^{M} d_i \; / \prod_{j=1,2,4,5} d_j) * V_{Q_1 \text{ AND } C,2} =$

(3*10*7*3*2*3)/(3*10*3*2) * 8.

Repeat this to find $|E(Q_i \text{ AND } C)|$ for Q_i AND C for all i integrity rules and find $E(Q_i \text{ AND } Q_j \text{ AND } C)$ for all i,j such that $1 \leq i < j \leq r$ and so on. From this Φ_2 is obtained and p_1 follows immediately.

From table(4.2) below, the behavior of this probability can be observed. It decreases as the uniformity of the distribution of records increases and as the density of the database increases.

Table 4.2 Behavior of ρ = Prob[C=empty | C=valid] for Increasing p_1 and Varying Parameters; Distribution, N, and σ.

M	σ	$\sigma 1$	N	δ	Distribution	p_1	ρ
5	1024	560	300	226	Uniform	1	0.60
						2	0.36
						3,4	0.21
						5-560	0.00
					Normal	1-10	0.99
						11-22	0.89
						23-35	0.78
						36-50	0.68
						51-68	0.58
						69-90	0.48
						91-119	0.38
						120-162	0.27
						163-243	0.17
						244-560	0.00
					Exponential	1-10	0.99
						11-22	0.89
						23-35	0.78
						36-50	0.68

						51-68	0.58
						69-90	0.48
						91-119	0.38
						120-162	0.27
						163-243	0.17
						244-560	0.00
5	1024	560	100	89	Uniform	1	0.84
						2	0.71
						3,4	0.59
						5,6	0.42
						7-9	0.30
						10-14	0.18
						15-560	0.00
			300			1	0.60
						2	0.36
						3,4	0.21
						5	0.08
						6-560	0.00
			500			1	0.42
						2	0.17
						3	0.07
						4-560	0.00
5	1024	560	300	226	Uniform	1	0.60
						2	0.36
						3,4	0.21
						5-560	0.00
3	125	68		66		1	0.03
						2-68	0.00
3	24	13		13		1-13	0.00

4.4.2 Consistency of responses

The following example will point out the type of inconsistencies to be expected from such a response strategy. Given Q = Avg(Sex = Female AND Level = PHD AND Faculty = Eng,Salary), where C = (Sex = Female AND Level = PHD AND Faculty = ENg) specifies no records, Q must be 0. Also known is the fact that C_1 = (Sex = Female AND Level = PHD AND Faculty = Eng AND Status = Fulltime), C_2 = (Sex = Female AND Level = PHD AND Faculty = Eng AND Status = Parttime), and C_3 = (Sex = Female AND Level = PHD AND Faculty = Eng AND Status = Indep) all specify no records and Freq(Sex = Female AND Level = PHD AND Faculty = Eng) must be equal to Freq(C_1) + Freq(C_2) + Freq(C3). But since all the characteristic formulas specify 0 records, the perturbed frequency will be 1/s*n for each of them. This leaves the situation where 1/s*n = 1/s*n + 1/s*n + 1/s*n, hence an inconsistent situation. Similarily the perturbed average can be seen to be just as inconsistent.

What are the dangers of using such an obviously inconsistent system? Since in our example the user can know that the frequency of (Sex = Female AND Level = PHD AND Faculty = Eng), 1/s*n is the perturbed statistic, could be 0, 1/N, 2/N, then the response to an average query is based on a small population and conclusions drawn from it should not be considered as accurate. A valid user probably would not use the average in this case since he knows it is based on a small population. The accuracy of the data base is not necessarily violated by the information because if there are no records satisfying Female and PHD and Eng, it cannot be concluded that the salary of these non existent records is lower or higher on the average then those of Male and PHD and Eng.

The integrity of the database suffers in small measure in this situation due to the inconsistencies mentioned above.

However, this should not affect the usefulness of the system because the user must be aware of the fact that empty queries are responded to as 1/s*n for frequency queries and a response based upon the semantics of the database and query for average queries. Then these inconsistencies should come as no surprize. On the other hand, the purpose of perturbing the statistics at all, is for the reason that in the event a user is trying to compromise the system, these inconsistencies appearing for sensitive queries (whose set size is 0) would be enough to block his efforts.

In the case of frequency queries, a valid empty set query can be answered as follows: Pfreq(C) = 1/sn where n is the size of the smallest group bigger than the average group of size N/s. If the partition is a "good" partition, the size of the groups should not vary too much from the value of N/s. Therefore, most frequency queries of size 1 will responded to with 1/sn, preventing compromise. A user on receiving a response for pfreq = 1/sn will not know if the true frequency is 0 or 1/N. The accuracy of the response, all though not exact, is sufficient for statistical summaries.

CHAPTER 5

PROBABILISTIC TRACKER ATTACKS

5.1 Introduction

Given an environment where true responses are given to queries and query set sizes are restricted to a certain range, Schlorer [29] and Denning [14] developed the idea of trackers as a means of compromising sensitive information. Denning [16] successfully eliminates the threat of straightforward tracker attacks by perturbing the reponses to queries in her random sampling approach to securing a database. This section reviews the notion of trackers and presents the theory behind a possible method of using them to break Denning's random sampling system. The method is based on range reduction techniques introduced by Alagar [4]. This strategy narrows down the true answer to a range, based on the perturbed response, and then systematically queries the system in such a way as to reduce this range to the correct answer. This is done with some determinable degree of probability. Once it is established that the true size of a query set can be determined, trackers can be applied in the usual way. Although the results will be less than 100% accurate, they may be as much as 90 - 95 % accurate.

Recall the notation for random sampling from Chapter 2: X_C is the set of records satisfying C, n_C is the size of X_C and X_C^* and n_C^* are, respectively, the records in and size of the sample taken from X_C. The sampling criterion is determined by the value of f(C,i), C is the characteristic formula, and i a record εX_C. If f(C,i) = 1 then i is kept for the sample and p is the probability of this happening. The perturbed statistics are calculated from the random samples; the perturbed frequency = $\frac{n_C^*}{pN}$, the perturbed average = $\frac{1}{n_C^*} * \sum_{i \varepsilon X_C^*} v_{ij}$ where v_{ij} is the value in the j^{th} field of record i.

5.2 Trackers

Trackers were developed as a tool to be used in compromising systems using small query set restriction as a means of securing its information. Let t be the threshold stipulating the smallest allowable query and N-t the largest. A general tracker is a formula T whose query set size, n_T is in the range [2t,N-2t]. T is used to find the query set size of C, an unanswerable query, and possibly to find more about individuals satisfying the characteristics described by C then is allowed. Given that the function Count, which returns the size of the query set specified, is available to the user, and $n_C < t$ for this particular C, then queries of the form Count(C+T), Count(C+ $\bar{T}$), Count(T), and Count($\bar{T}$), will all be answerable. Count(C) of course would be restricted but could be calculated from the formula

Count(C) = Count(T + C) + Count($\bar{T}$ + C) - [Count(T) + Count($\bar{T}$)]

which is the same as

$$\text{Count}(C + T) + \text{Count}(C + \bar{T}) - N.$$

Because random sampling does not give true answers to queries, the tracker attack breaks down when applied to a system protected by random sampling. It would seem that the maximum information a user could obtain would be n^*_{C+T} and $n^*_{C+\bar{T}}$. (see section 5.3). But Count(C) $\neq n^*_{C+T} + n^*_{C+\bar{T}} - N$. Therefore, for trackers to be effective, n_{C+T} and $n_{C+\bar{T}}$ must be obtained from n^*_{C+T} and $n^*_{C+\bar{T}}$.

It is the aim of the next 2 sections to show that it is possible to obtain n_Q from n^*_Q for some Q, with some degree of probability, but it is not yet known whether or not with a dedicated computer system n_Q could be obtained for all possible Q.

5.3 Probabilistic Ranges

It is assumed in Denning [16] that the user knows both p, the sampling probability and N, the total number of records in the system. From this information and the perturbed response to a frequency query, a range [a,b] can be calculated such that the size of the query set lies within [a,b] with some predetermined probability.

The first step in calculating [a,b] is to transform the perturbed response Pfreq(C) into the size of the random sample n^*_C. This can be done since it is known that Pfreq $= \frac{n^*_C}{pN}$. Therefore p*N*Pfreq(C) $= n^*_C$.

Suppose the range for the true answer is to be found with 95% probability. [a,b] is obtained in two steps.

Firstly, $a = n_C^*$. This is because it is true that the size of the sample taken from the query set must be at most as big as the query set itself. Therefore n_C is $\geq n_C^*$.

A first approximation to [a,b] could be to choose b as N because with 100% probability, n_C is a member of $[n_C^*,N]$. This first approximation is not very useful since $[n_C^*,N]$ becomes large quickly as n_C^* becomes smaller than N. The next approximation results from minimizing the upper bound of [a,b].

A discussion of the behvior of n_C and n_C^* will help in developing a method of reducing b from N to a number closer to a, bearing in mind that as b decreases, the probability that n_C is in [a,b] also decreases.

Given a query with a true count of $n_C = X$, make the following assertions

1. $\text{Prob}[n_C^* > X] = 0$,
2. $\sum_{k=0}^{X} \text{Prob}[n_C^* = k] = 1$, that is $\text{Prob}[n_C^* \in [0..k]] = 1$,
3. $\text{Prob}[n_C^* = k \mid n_C = X] = \binom{X}{k} p^k (1-p)^{X-k}$.

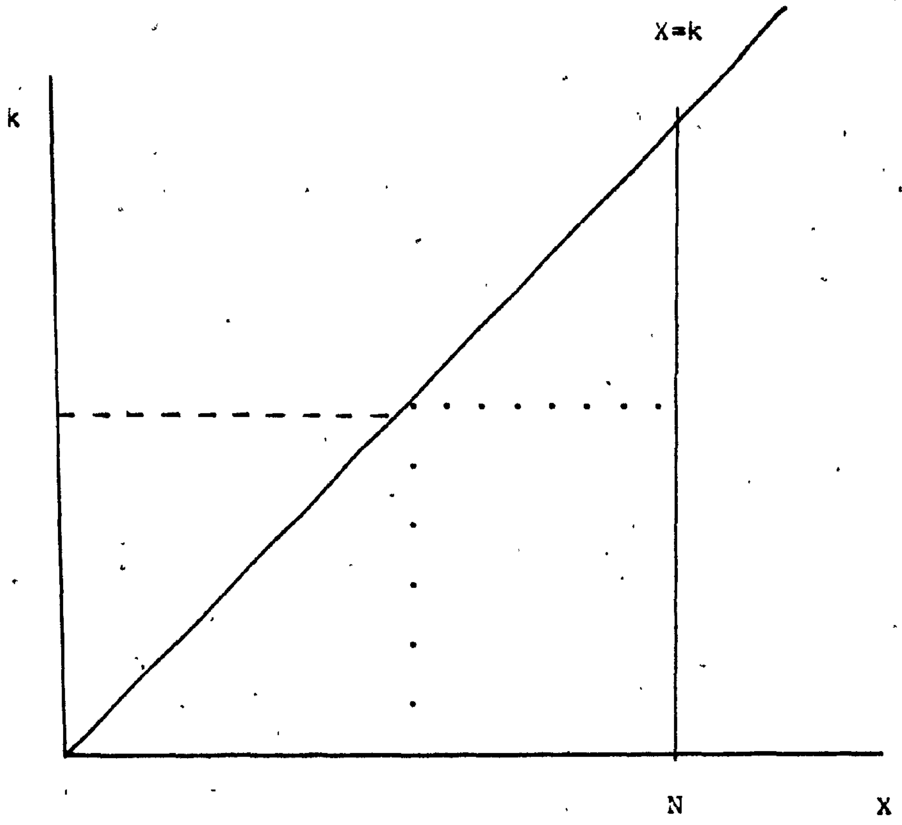
Given a query whose random sample is of size k, i.e. $n_C^* = k$, assert that

4. $\text{Prob}[n_C < k] = 0$,
5. $\sum_{X=k}^{N} \text{Prob}[n_C = X] = 1$, that is $\text{Prob}[n_C \in [k..N]] = 1$.

A graphic representation can be found in figure (5.1) where the X axis represents the value of n_C and the k axis represents the value of n_C^*. Interpret the graph to mean for any X and any k, (X,k) is = Prob[$n_C^* = k \mid n_C = X$] which in words means the probability that the sample n_C^* taken from X_C is k when $n_C = X$. This probability can be computed from the formula given in 3 above. Note that the line X = k splits the plane into two regions. Points on or below X = k are non zero. Points above X = k must be zero. From 1 and 2 above, if $n_C = X$ then n_C^* must be between 0 and X. A sample is never larger than the size of the set from which it was chosen. Also, from 4 and 5, if $n_C^* = k$, then it is known that n_C must be between k and N by a similar reasoning. For example point(5,3) lies below X = k indicating that Prob[$n_C^* = 3 \mid n_C = 5$] $\neq 0$. This agrees with our initial assertions that the size of the sample must be ϵ [0,5] given the size of X_C is 5, and the size of the original set must be ϵ [3,N] given the size of the sample is 3. Point(3,5) lies above X = k indicating that Prob[$n_C^* = 5 \mid n_C = 3$] = 0, also in line with the nature of the system. That is, the size of the sample can never be more than the size of the set, and the size of the original set can never be smaller than the sample.

The task here is to find Prob[$n_C = x \mid n_C^* = k$], the probability that $n_C = X$ given that the size of the sample taken from X_C is k. So fix k and the horizontal sample space for k becomes the points (X,k) for X = 0..N. The probability associated with one point out of this space is

Figure 5.1 Prob$[n_C = X \mid n_C^* = k]$



$$\sum_{k=0}^{X} wt(X,k) = 1$$

$$\sum_{X=k}^{N} wt(X,k) = 1$$

$$\sum_{X=0}^{k-1} wt(X,k) = 0$$

$$\text{Prob}[n_C = X \mid n_C^* = k] = wt(X,k) / \sum_{X=k}^{N} wt(X,k)$$

$\text{weight}(X,k)/\sum_{X=0}^{N} \text{weight}(X,k)$. But $\text{weight}(X,k) = 0$ for $X = 0..k-1$.
Therefore $\text{weight}(X,k)/\sum_{X=k}^{N} \text{weight}(X,k)$ is =
$\text{Prob}[n_C^* = k \mid n_C = X] \,/\, \sum_{X=k}^{N} \text{Prob}[n_C^* = k \mid n_C = X]$
which is $\text{Prob}[n_C = X \mid n_C^* = k]$.

Finally, given a 95% certainty for the range of the true response, find b such that $\sum_{X=k}^{b} \text{Prob}[n_C = X \mid n_C^* = k] \geq .95$, $b \leq N$. Now it can be said with at least 95% certainty, n_C is in $[a,b]$.

5.4 Properties of Ranges

Because range manipulation plays an important role in estimating the size of a query set, this section lists some of the properties of ranges along with examples to illustrate them. See Alagar[4].

Properties:

(i) Addition

$[a,b]+[c,d] = [a+c,b+d]$,
if $x \in [a,b]$ and $y \in [c,d]$ then $x+y \in [a,b]+[c,d]$.
Ex. $[2,5]+[7,10] = [9,15]$,
$3+9 = 12 \in [9,15]$

(ii) Subtraction

$[a,b] - [c,d] = [a-d,b-c]$,
if $x \in [a,b]$ and $y \in [c,d]$ then $x-y \in [a,b] - [c,d]$
Ex. $[7,10] - [2,5] = [2,8]$,
$8-2 = 6 \in [2,8]$

(iii) Intersection

$[a,b] \cap [c,d] = [\max(a,c), \min(b,d)]$

Ex. $[2,10] \cap [5,30] = [5,10]$

(iv) Reduction

if $x \in [a,b]$ and $x \in [c,d]$

then $x \in [a,b] \cap [c,d]$

Ex. $9 \in [7,10] \cap [2,20]$

(v) One element range

$a = [a,a]$

Ex. $5 = [5,5]$

5.5 Formula Partitioning

If C is the formula for which the query set size is sought, and $[a_C,b_C]$ is the corresponding range obtained for the size of X_C, then it will be necessary to partition C into a number of formulas to carry out the reduction process on $[a_C,b_C]$. If C is a formula and $C_1, C_2, \ldots C_k$ is a partiton of C, then $\{C_i\}$, $i=1..k$, has the following properties:

(1) union$\{X_{C_i}\}$ for $i=1..k = X_C$ and

(2) intersection$\{X_{C_i}\} = \{\}$,

where X_{C_i} is the query set of C_i. A partition of C can be obtained by intersecting C with every value of some attribute. For example, if C = (Marital Status = Single) then C_1 = (Marital Status = Single AND Sex = Male) and C_2 = (Marital Status = Single AND Sex = Female) form a partition of C. The intersection of single males with single females is obviously {} just as the union of all single males and all single females is all single persons in the database. Another partition would be C_1 = (Marital Status = Single AND (Age = 0-14 OR 15-29)), C_2 = (Marital Status = Single AND (Age = 30-44 OR 45-59)), and C_3 = (Marital Status = Single AND (Age = 60 -74 OR 75-89)).

In general, if $\{a_1, a_2, \ldots, a_k$ is a partition of the values of attribue A and Q_1 = OR'ing of all values in a_1, Q_2 = OR'ing of all values in a_2, ... Q_k = OR'ing of all values in a_k then
C_1 = (C AND Q_1), C_2 = (C AND Q_2), ... C_k = (C AND Q_k)
which forms a partition of C. The values of an attribute can be partitioned in many ways and each way gives a different partition for C. To push the point even further, once C has been partitioned into say $\{C_i\}$, i=1..k, each of the k C_i's can be partitioned on another attribute.

The point of this exercise will become clear in the next section on range reduction in the random sampling environment.

5.6 Range Reduction

Let $[a_C, b_C]$ be the range found for the true count of C and let p_C = probability that $n_C \in [a_C, b_C]$. C can be partitioned into C_1, C_2, ... C_k and ranges $[a_{C_1}, b_{C_1}]$ ... $[a_{C_k}, b_{C_k}]$ found such that $n_{C_1} \in [a_{C_1}, b_{C_1}]$, $n_{C_2} \in [a_{C_2}, b_{C_2}]$, ... and $n_{C_k} \in [a_{C_k}, b_{C_k}]$, all with p_C * 100% probability. It is true that n_C = $n_{C_1} + n_{C_2} + \ldots + n_{C_k}$ and therefore from property(i) $n_C \in \sum_{i=1}^{k} [a_{C_i}, b_{C_i}]$ with $(p_C)^k$ * 100 % probability. From property(iv), since $n_C \in \sum_{i=1}^{k} [a_{C_i}, b_{C_i}]$ with $(p_C)^k$ * 100 % probability and $n_C \in$

$[a_C, b_C]$ with p_C * 100 % probability, this implies $n_C \in \sum_{i=1}^{k} [a_{C_i}, b_{C_i}] \cap [a_C, b_C]$ with $(p_C)^{k+1}$ * 100 % probability. If the intersection results in a one element range, this element is the true count with probability $(p_C)^{k+1}$. If the intersection fails to isolate one number, then C can be partitioned differently and the process repeated. Or C_1, ... C_k can each be partitioned in an attempt to reduce $[a_{C_i}, b_{C_i}]$, for i=1..k, their respective ranges, down to one element ranges. Obviously, better approximations occur when successive k's are small, as the error propagation is less. It might not be possible to obtain a one element range intersection in every case, but the following results show that it is at least possible in some cases.

5.7 Successfull Probabilistic Range Reductions

This section demonstrates probabilistic range reduction for four formulas. The size of the random sample taken was obtained in each case from a psuedo database containing 500 records, with 5 attributes each having domain size of 4. Remember that in a formula, a number of the form i0j means the jth value of the ith atribute is being specified. For simplicity, * denotes intersection and + denotes union.

Example 1. For C = (101 * 401), $n_C^* = 31$. With 95% probability, n_C is found to be $\in [31,34]$. Let C_1 = (101 * 401) * (201 + 203 + 204) and C_2 = (101 * 401) * (202), $n_{C_1}^* = 21$ and $n^*_{C_2} = 7$. With 95% probability, we have found $n_{C_1} \in [21,23]$ and $n_{C_2} \in [7,8]$. Since $n_C = n_{C_1} + n_{C_2}$, $n_C \in [21,23] + [7,8] = [28,31]$. If $n_C \in [31,34]$ and $n_C \in [28,31]$ then $n_C \in [31,34] \cap [28,31] =$

[31,31] with $(.95)^3$ * 100% which = 87% probability. Since [31,31] = 31, the true count of (101 * 401) has been found with 86% probability.

Example 2. For C = (101 * 402) * (201 + 202 + 203), n^*_C = 20. With 95% probability, $n_C \in [20,22]$. Let C_1 = (101 * 402) * (201 + 202 + 203) * (301 + 303) and C_2 = (101 * 402) * (201 + 202 + 203) * (302 + 304), $n^*_{C_1}$ = 11 and $n^*_{C_2}$ = 11. With 95% probability, n_{C_1} was found to be $\in [11,13]$ and $n_{C_2} \in [11,13]$. Since $n_C = n_{C_1} + n_{C_2}$, $n_C \in [11,13] + [11,13] = [22,25]$. If $n_C \in [22,26]$ and $n_C \in [20,22]$ then $n_C \in [22,26] \cap [20,22] = [22,22] = 22$. Hence with $(.95)^3$ * 100% which = 87% probability, the true count of (101 * 402) + (201 + 202 + 203) is 22.

Example 3. For C = (101 * 403) * (201 + 203 + 204), n^*_C = 28. With 95% probability $n_C \in [28,31]$. Let C_1 = (101 * 403) * (201 + 203 + 204) * 303 and C_2 =(101 * 403) * (201 + 203 + 204) * (301 + 302 + 304), $n^*_{C_1}$ = 4 and $n^*_{C_2}$ = 21. With 95% probability we have found $n_{C_1} \in [4,5]$ and $n_{C_2} \in [21,23]$. Since $n_C = n_{C_1} + n_{C_2}$, $n_C \in [4,5] + [21,23] = [25,28]$. If $n_C \in [25,28]$ and $n_C \in [28,31]$ then $n_C \in [25,28] \cap [28,31] = [28,28] = 28$. Hence with $(.95)^3$ * 100% = 87% probability, the true count is 28.

Example 4. For C = (101 * 404) * (301 + 302 + 303), n^*_C = 26. With 95% probability, $n_C \in [26,29]$. Let C_1 = (101 * 404) * (301 + 302 + 303) * 202 and C_2 = (101 * 404) * (301 + 302 + 303) * (201 + 203 + 204), $n^*_{C_1}$ = 7 and $n^*_{C_2}$ = 18. With 95% probability we have found $n_{C_1} \in [7,8]$ and $n_{C_2} \in [18,20]$. Since $n_C = n_{C_1} + n_{C_2}$,

$n_C \in [28,28] = 28$. Hence with $(.95)^2 * 100\%$ which = 90% probability, the true count is 28.

5.8 Consequences of Probabilistic Range Reduction

Take for granted, for the moment, that true counts of queries could be estimated with a large degree of probability when querying a random sampled database. If these counts were used in trackers in the usual way, then the database could be compromised in the same way as a database protected by small query set restriction, albeit only to the degree of probability that the the true counts were estimated, a degree of probability < 100%.

The examples of total range reductions given above are enough to spark interest in the potential of compromising in this way. For these consequences to be a serious threat to the security of random sampled statistical databases though, both the ease of range reduction (the probability that range reduction could be achieved for any given formula) would have to be found feasible and the hit ratio (the frquency of range reductions which actually returned the true count) would have to be found experimentally as high as the theoretical hit rate of $(p_c)^{k+1} * 100\%$.

CHAPTER 6

CONCLUSION

In Chapter 2, the criticisms of partitioning as a means of securing a statistical database were cited. They included the absence of some measure by which a particular partition of a database could be rated. Without such a measure, partitioning has the drawbacks that querying on the ill-formed groups would lead to inaccurate responses, rendering the database useless as an information tool. This criticism was overcome with the development of two techniques for measuring the appropriate entropies of a partition indicating the relative number of groups in the partition and their size variation as well as the similarity of the records within a group. The partition and average group entropy values could be calculated for the theoretically best possible partition for any given application. If a number of actual partitions are available for that application, comparing their entropies to the one calculated for the best partition would aid in choosing the best partition from the ones available.

Another criticism of partitioning was based on the amount of information loss inherent in the response strategy employed. Although Glaser's response strategy fared better than that of Schlorer's, it came up short with respect to queries with empty query sets. Strategies for responding to these queries for average and frequency statistics when the semantic integrity of the database was not violated, and a means for recognizing those queries which did violate this integrity were presented.

Glaser [22] pointed out that a database can be partitioned efficiently so that there is accuracy of statistics and security of information. In addition he indicated how it can cope with a dynamic database. However there were the few gaps noted above. The results of this thesis fill the gaps and establish partitioning as a sound and safe technique for securing a statistical database.

Random sampling fares well as a security measure in that the work involved in obtaining the samples is minimal, the statistics produced are accurate and it can certainly support a changing environment. Unfortunately, random sampling suffers slightly in the areas of information loss and compromise. The information loss is limited to information associated with queries whose query sets are small, and an attempt to overcome this results in useless statistics due to the nature of the sampling probability.

It was shown in Chapter 5 how an attempt at compromising a database under random sampling can be made. The basic idea is to obtain a confidence interval for the number of records in the query set of some query with high probability and then use the range reduction techniques of Alagar [4] to obtain the exact number. As far as we know this is the first and only known attempt at compromising the random sampling method. Our results, though not exhaustive, have clearly shown that in some cases the true count of a query was founa . There is potential for further stuay in this area in aetermining now much compromise is possible and how easy it is to achieve it.

BIBLIOGRAPHY

[1] Achugbue, J.O. and Chin, F.Y. The Effectiveness of Output Modification by Rounding for Protection of Statistical Databases. Information, Vol.17, No.3, Mar.1979, p.209-218.

[2] Alagar, V.S. Complexity of Compromising Statistical Databases, Dept. Of Computer Science, Concordia University, Montreal, Canada, Nov.1980.

[3] Alagar, V.S., Blanchard, B., and Glaser, D. Effective Inference Control Mechanisms for Securing Statistical Databases. AFIPS Joint Computer Conference Proceedings, Vol.50, 1981.

[4] Alagar, V.S. Range Response: An Output Modification Technique and an Analysis of its Effectiveness for Security of Statistical Databases. Dept. Of Computer Science, Concordia University, Montreal, Canada, July 1983.

[5] Campbell, D.T., et al. Confidentiality-preserving Modes of Access to Files and to Interfile Exchange for Useful Statistical Analysis. Eval. Q. Vol.1, No.2, May 1977, p.269-299.

[6] Chin, F.Y. and Ozsoyoglu, G. Security in Partitioned Dynamic Statistical Databases. Proc. IEEE Compsac, 1979, p.594-601.

[7] Chin, F.Y. and Ozsoyoglu, G. Statistical Database Design. ACM TODS, Vol.6, No.1, Mar.1981, p.113-139.

[8] Chin, F.Y. and Ozsoyoglu, G. Auditing and Inference Control in Statistical Databases. IEEE Trans. Software Engineering, Vol.SE-8, No.6, Nov.1982, p.574-582.

[9] Cox, L.H. and Ernst, L.R. Controlled Rounding. US Bureau of the Census, Washington, D.C., Jan.1981.

[10] Dalenius, T. and Reiss, S.P. Data-Swapping--A Technique for Disclosure Control. Dept. Of Computer Science, Brown University, Providence, R.I., 1978.

[11] Dalenius, T. A Simple Procedure for Controlled Rounding. Statistisk Tidskrift, Vol.3, 1981. P.202-208.

[12] Davida, G.I., Linton, D.J., Szelag, C.R., and Wells, D.L. Database Security. IEEE Trans. On Software Engineering, SE-4, No.4, Nov.1978, p.531-533.

[13] Denning, D.E. and Denning, P.J. Data Security. Comput. Surv. Vol.11, No.3, Sept.1979, p.227-249.

[14] Denning, D.E., Denning, P.J., and Schwartz, M.D. The Tracker: A Threat to Statistical Database Security. ACM TODS, Vol.4, No.1, Mar.1979, p.76-96.

[15] Denning, D.E. and Schlorer, J. A Fast Procedure for Finding a Tracker in a Statistical Database. ACM TODS, Vol.5, No.1, Mar.1980, p.88-102.

[16] Denning, D.E. Secure Statistical Databases with Random Sample Queries. ACM TODS, Vol.5, No.3, Sept.1980, p.291-315.

[17] Denning, D.E. Cryptograpny and Data Security, Addison-Wesley, Reading, Mass., 1982.

[18] Denning, D.E. And Schlorer, J. Inference Controls for

Statistical Databases. Computer, July 1983, p.69-82.

[19] Fellegi, I.P. On the Question of Statistical Confidentiality. J. Amer. Statist. Assoc., Vol.67, No.37, Mar.1972, p.7-18.

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

[21] Fellegi, I.P. Controlled Random Rounding. Survey Methodology, Vol.1, No.2, 1975, p.123-133.

[22] Glaser, D.M. Partitioning A Database to Provide Security Against Statistical Inference. Masters thesis, Department of Computer Science, Concordia University, Montreal, Jan.1981.

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

[24] Nargundkar, M.S. and Saveland, W. Random Rounding to Prevent Statistical Disclosures. Proc. Amer. Stat. Assoc. Soc. Stats. Sec., 1972, p.382-385.

[25] Records, Computers and the Rights of Citizens. By the US Dept. Of Health, Education, and Welfare. N.p.: The Colonial Press Inc., 1973.

[26] Schlorer, J. Confidentiality of statistical Records: A Threat Monitoring Scheme for On Line Dialogue. Meth. Of Inf. In Medicine, Vol.15, No.1, Jan.1976, p.36-40.

[27] Schlorer, J. Disclosure from Statistical Databases:

Quantitative Aspects of Trackers. ACM TODS Vol.5, No.4, Dec.1980, p.467-492.

[28] Schlorer, J. Security of Statistical Databases: Ranges and Trackers, Klinische Dokumentation, Universitat Ulm, W. Germany, Nov.1981.

[29] Schlorer, J. Information Loss in Partitioned Statistical Databases, Klinische Dokumentation, Universitat Ulm, W. Germany, 1983.

[30] Yu, C.I. and Chin F.Y. A Study on the Protection of Statistical Databases. ACM SIGMOD Conference on Management of Data, Toronto, Ontario, Aug.1977, p.169-181.