INADEQUACIES OF INFERENCE CONTROL IN STATISTICAL DATABASES

Achilles Venetoulis

Technical Report No. 26
January 1988

Laboratory for Computational Statistics

Department of Statistics
Stanford University
Inadequacies of Inference Control in Statistical Databases

by

Achilles Venetoulias

Department of Statistics
Stanford University
Stanford, California

The increasing misuse of computers and the enhanced threat to personal privacy through access to data banks have emphasized the problem of database security and stimulated research in the development of technical safeguards for data. The goal is to protect a database from misuse. We examine a particular instance of this problem: How to employ inference control mechanisms in order to protect statistical databases from the release of confidential information.

We identify those characteristics of a statistical database which are necessary for the definition of the problem and explain the general principles of penetration techniques. We present and critically compare the different methods of inference control by discussing the reasons for their respective inadequacies. In certain cases, we propose alternatives and potential improvements to the existing methods. We conclude that data transformations appear to be the most promising approach for ensuring the uncompromisability of a database.

Keywords: Statistical Databases, Queries, Security, Compromise, Inference Control, Data Suppression, Data Transformations.
Inadequacies of Inference Controls in Statistical Databases

Achilles Venetoulis

1. Introduction

The increasing misuse of computers and the enhanced threat to personal privacy through access to data banks have both emphasized the problem of database security and stimulated much interest in the development of technical safeguards for data. The goal of such research is to protect the database from misuse. There are as many aspects to the problem of misuse as there are precise definitions of the term “misuse”. Two well-known instances of the problem are undesired modifications of the database and release of confidential information from the database. These examples do not exhaust the range of possible instances of misuse but they prove useful as an introduction to the problem.

Generally, database security is either external or internal. External security encompasses operations such as personnel screening and limiting access to the computer system which are outside the main computing system. Internal security refers to operations within the computing system. Although some security mechanisms like user authentication, security monitoring and password management lie at the interface between the system and the users, we will not treat external security directly. We will concentrate on one kind of internal security control, known as inference control, as it is applied to statistical databases.

There are four interrelated but distinct types of internal security safeguards. These controls regulate the access to stored objects, the flow of information from one stored object to another, the encryption of confidential data and the inference of confidential data in statistical databases. We will consider in detail only the last issue. The reader should note that the problem of retrieving privileged information from a database is not limited to statistical databases. Recently, there have been attempts, based on methods of logical inference, at treating the problem in any database. In particular, Sicherman, et al. [SiJR83] develop a number of safety criteria concerning answers to potentially dangerous queries (i.e., queries whose answers
reveal "secrets" from the database) and establish a sufficiency criterion for the protection of privileged information. Their approach, however, differs from the approaches employed to attack the problem of security in statistical databases and the results appear impractical. We will focus, therefore, on the traditional methods used to protect statistical databases.

The issue of inference of confidential data was first examined for statistical databases. See Hoffman and Miller [HoMi70] and Haq [Haqm74, Haqm75]. Since the early treatments of the database security problem, it has been noted that the issues are quite similar to those appearing in the design of operating systems. In fact, many of the commonly used internal security controls for the access to stored objects or the flow of information are common to the design of a database management system that tries to enforce security and to the design of a sound operating system. Such controls involve concepts like proper user identification, object dependent access limitations, capacity addressing for access controls, and security classes for flow controls. Despite this similarity, the operating system solutions do not suffice to solve the database security problem. The inadequacy of the operating system solutions for this context can be displayed by trivial examples of the inference control issue. In general, we understand much more about controlling access to individual records than controlling inferences. The distinction lies with the fact that access control mechanisms make no interpretation of the content of a database, have no notion of a history of previously released information, and allow for an easy breakdown of complex operations. For example, a query in a database requires a decision as to whether the particular query should be permitted. This decision cannot be based on fragments of the query [DoJL79]. To widen the gap between security control mechanisms in databases and these in operating systems we note that we are concerned with users that are performing read access and hence, access controls that allow read but not write or alter access are not applicable. In these terms, the need for protection of the database from undesired modifications can be, at least partially, satisfied by either a combination of access and flow controls or by encryption methods. The protection of confidential information on the other hand cannot be insured by such mechanisms, and different tools have been developed to solve
the problem. We now turn to this issue.

2. Statistical Databases

There is no clear cut definition of a statistical database, largely because the different models of statistical databases in the literature are founded on "locally" convenient and useful definitions. Each particular model corresponds to a definition of a statistical database, which actually renders the model useful and allows for an analysis of certain aspects of the problem. A statistical database is a collection of records from which queries concerning certain subsets of the records may be answered. Usually the queries are of a statistical nature pertaining to subsets of the database that are of special interest to the user. We assume that a statistical database actually contains all the relevant data of a whole population. Under this assumption, the queries can be thought of as questions regarding the statistical properties of important subpopulations. There has been some discussion in the literature concerning the accuracy or the necessity of such an assumption. It is argued that realistically databases contain only the data for subpopulations. Furthermore, different schemes have been suggested as to how subpopulation identification information should be incorporated into the database and/or into the database treatment by the database management system. However, the whole population assumption not only implies the discussion but seems not to affect the generality of the problem at least as far as inference control is concerned. In fact, adopting the rather flexible statistical viewpoint that the user is only concerned with the contents of the database alleviates even the semantic complications of the argument.

The above definition of a statistical database is broad and seemingly vague. In principle, any collection of quantitative information residing in a computer system is a candidate statistical database. As soon as the body of information is interrogated and statistically analyzed, either in total or by sampling of subsetting, it becomes a statistical database [NDSS83]. Determining the exact characteristics or features of a statistical database is collateral to this paper. Fortunately, for the problem of database security the concept of a statistical database as pre-
viously described suffices. The necessary modifications to this concept will be mentioned as encountered in the model discussion. The idea conveyed by this definition is basically that statistical databases are databases which are used primarily for statistical purposes.

In the relational model suppose that we have a database, $D$, containing information on a population that consists of $n$ individuals. One of the variables recorded for each individual is variable $V$ (equivalent to attribute $V$) that takes on values $v_i$; let us denote by $v_i$ the value of $V$ for individual $i$. Then statistical queries, referring to $V$, can range from smooth statistics like $\sum_{i \in S} v_i^m$ (where $S$ is some subset of $\{1, 2, \ldots, n\}$ and $m$ is an integer) to discontinuous ones like $\max_{i \in S}(v_i)$, $\min_{i \in S}(v_i)$ or $\text{median}_{i \in S}(v_i)$. Observe that the first type of query, $\sum_{i \in S} v_i^m$, reduces to counting the elements in $S$ when $m = 0$ (the so called count queries ask for $\text{count}(S) = |S| = \sum_{i \in S} 1$) and can easily produce the sample moments when combined with $|S|$ (a sample $r$-moment is defined to be $n^{-1} \sum_{i=1}^{n} X_i^r$ for any population of $n$ individuals). These kinds of queries are two examples but they represent the vast majority of the commonly used queries. Note that many frequently computed statistics can be obtained from the answers to the above and similar queries.

3. The Problem

When statistical evaluations are done on a file that contains sensitive information, the question of privacy protection arises. The confidentiality dilemma involves providing statistically useful summary information while protecting the privacy of the individuals. Suitable mechanisms for protecting information may depend on the logical data model. Current research revolves around what is actually obtainable within the summary information criteria constraints and what methods provide security mechanisms in a multiuser environment.

Generally, it is the aggregate of the data and not the actual data for each individual object that is important for statistical databases. Though this principle is not strictly true, we assume that this is always the case. Often statistical analysis calls for a special treatment of outliers, the observations that do not conform to the general patterns found in the population. In such
instances, exact identification of cases contributes to a better understanding of the problem and results in more accurate statistical inferences.

We concentrate on the purpose and nature of statistical databases and we suppose:

1) that statistical databases contain some privileged information or confidential data which should remain unknown to the users; and,

2) that statistical databases must supply statistical summaries without violating this confidentiality.

We also assume that the users possess some preliminary knowledge, often referred to as pre-knowledge or prior knowledge. This assumption is natural and realistic since users must have some information about the database for it to be of any interest to them. For all practical purposes, the users will have at least an idea of the integrity constraints of the databases. More specific assumptions about prior user knowledge play an important role in determining the results obtainable from any approach. They are determined by the particular model and viewpoint and vary from approach to approach.

We would ideally like to allow for a rich user–database interface by giving the users the capability of extracting from the database as much information as possible, while at the same time not revealing “secrets” from the database (a “secret” is any part of the sensitive data). However, the restriction of keeping database secrets concealed proves to be a fundamental consideration in the database system design which is difficult to strictly enforce.

Controlling the information obtainable from a database has proven to be a particularly difficult problem. Statistical summaries contain vestiges of original information, hence the database questioner may be able to deduce the original data by processing the summaries appropriately. Seemingly innocuous queries may prove informatively valuable when strategically placed in an intelligently composed series. Correlating the answers and combining them with prior knowledge may enable an intruder to obtain pieces of confidential data. This occurrence is known as a compromise.
A compromise can be either personal (i.e., extraction of confidential information about a specific individual) or general (extraction of confidential information about the whole database) and either positive (confirmation of an assertion) or negative (confirmation of the negation of an assertion). Dobkin, et al. [DoJL79] summarize the conceptual scheme of a compromise occurrence as follows: There is a set of data elements in the database called the UNKNOWN set that user $U$ is not permitted to know. User $U$ somehow, perhaps as a result of previous queries, knows a set of data elements called the KNOWN set, some elements of which may not be explicitly recorded in the database. User $U$ asks a sequence of queries enlarging the set of KNOWN data elements. A compromise occurs when the KNOWN and the UNKNOWN set intersect.

Thus, our goal is to protect the confidential data from unauthorized disclosure by limiting the use of the database so that no sequence of (statistical) queries is sufficient to deduce confidential or private information. Theoretically, it involves three hypothetical individuals: the statistician, whose interest is to obtain aggregate statistics from the database; the database manager who wishes to secure the confidentiality of some sensitive data; and the database designer who needs to satisfy both their needs. Obviously, the risk of compromise occurrence depends on the method of release of statistical information.

Also, the degree of security, or conversely, the extent to which a database can be compromised, is a function of the specific model adopted, where it is understood that the concept of the model encompasses parameters like the user's prior knowledge, the size and the nature of the database, the form of the queries and the protection mechanisms implemented on the system. Thus, security is a relative issue; the objective of inference controls is to make the cost of compromise unacceptably high. Interestingly, the cost of retrieval of privileged information is customarily measured in terms of number of queries needed to achieve a compromise of the database. The dependence of this cost on the complexity of the queries asked and the user's ability to reuse and/or store intermediate results is neglected in the literature. However, because most of the widely used statistical queries are simple in nature this is a minor complication.
The following simple example, adopted from DeDe79, illustrates some of the concepts mentioned. We have a database that describes the contributions of some individuals to the political party of their preference:

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Occupation</th>
<th>Contribution ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antigone</td>
<td>F</td>
<td>Student</td>
<td>300</td>
</tr>
<tr>
<td>Hector</td>
<td>M</td>
<td>Student</td>
<td>50</td>
</tr>
<tr>
<td>Plato</td>
<td>M</td>
<td>Student</td>
<td>500</td>
</tr>
<tr>
<td>Euclid</td>
<td>M</td>
<td>Professor</td>
<td>1000</td>
</tr>
<tr>
<td>Noether</td>
<td>F</td>
<td>Professor</td>
<td>100</td>
</tr>
<tr>
<td>Kolmogorov</td>
<td>M</td>
<td>Professor</td>
<td>10</td>
</tr>
<tr>
<td>Iphigenia</td>
<td>F</td>
<td>Administrator</td>
<td>25</td>
</tr>
<tr>
<td>Roxanne</td>
<td>F</td>
<td>Administrator</td>
<td>150</td>
</tr>
</tbody>
</table>

To protect the confidentiality of the contributions, the system only responds to queries that involve at least two but no more than six individuals. Suppose a questioner knows from some external source that the characteristic formula

\[ F = \text{STUDENT AND FEMALE} \]

uniquely identifies Antigone. The questioner wishes to confirm this fact and additionally, to find out Antigone's contribution. To that end, the questioner poses the questions:

- \( Q_1 \): How many persons are STUDENTS?
- \( Q_2 \): How many persons are STUDENTS and MALE?
- \( Q_3 \): What was the total contribution of all STUDENTS?
- \( Q_4 \): What was the total contribution of all MALE STUDENTS?
The answers received are respectively: $A_1 = 3$, $A_2 = 2$, $A_3 = 850$ and $A_4 = 550$. Answers $A_1$ and $A_2$ confirm the questioner's prior knowledge. The formula $F$ identifies some individual (Antigone). The last two answers allow for an easy compromise of Antigone's contribution. The questioner needs only to subtract $A_4$ from $A_3$ to deduce the desired contribution. Note that the system does not object to any questions, because they all meet the stated requirement. Compromise has been achieved, although direct questioning about Antigone is prohibited.

We can define a database to be *compromisable* by a certain type of queries, iff after a sequence of such queries by the user there is a disclosure from the database. This definition is useful in that it formalizes the concept of a secure database and is a composite of existing definitions. A database is *secure* (under a protection scheme applied to it) from a certain type of queries iff it is not so compromisable (under the protection scheme). The definition emphasizes the need for further study in the field. Compromisability is surely connected to some form of inadequacy and yet, the above definition can lead to peculiar situations. It allows a compromisable database to be more acceptable than a non-compromisable database, if the number of possible compromises is small on the average, and thus tolerable. If, on the other end of the scale, a database is secure but allows estimation with small error of one of its crucial elements (say one attribute for the whole database) we regard it as unacceptable for security purposes [KaUl77]. This paradoxical consequence of the definition has lead to the concept of statistical security in a statistical database, which is analyzed later.

Before we begin the discussion of each particular security model, a final point warrants attention. Proof that a database is compromisable amounts to exhibiting a method for constructing a sequence of queries whose responses imply the previously unknown value of a data element in the database. To prove that a database is secure, we must show that no such sequence exists. The sharp contrast between these two logical directions is a preview of what follows. The fact that proving security is a more complex task than proving compromisability indicates that we might actually be in more trouble than initially expected. This turns out to be the case! Research conclusively demonstrates that security in a database is a rarer phenomenon
than we desire or imagine. In the usual cases, despite many elaborate protection schemes compromisibility is the rule rather than the exception.

4. Snooping

Suppose that the user already knows the values \(c_1, c_2, \ldots, c_j\) (which may be items in the database) and asks queries \(Q_1, \ldots, Q_k\) receiving responses \(A_1, \ldots, A_k\). Each query is of some allowable form and corresponds to some query set, which is the subset of the database used to answer the particular query. Assuming that the user does not possess any additional knowledge about the database, the penetration inference can be expressed as follows:

\[
\begin{align*}
B_1 & \implies I = I_1, \quad V = f_1(A_1, \ldots, A_k, c_1, \ldots, c_j) \\
B_2 & \implies I = I_2, \quad V = f_2(A_1, \ldots, A_k, c_1, \ldots, c_j) \\
\vdots & \quad \quad \quad \quad \quad \vdots \\
\vdots & \quad \quad \quad \quad \quad \vdots \\
B_\ell & \implies I = I_\ell, \quad V = f_\ell(A_1, \ldots, A_k, c_1, \ldots, c_j)
\end{align*}
\]

Here \(B_1, \ldots, B_\ell\) are Boolean conditions, exactly one of which must be true depending on the responses \(A_1, \ldots, A_k\). Suppose \(B_i\) is true — then \(I_i\) (which is some constant) identifies an individual in the database and \(f_i\) gives the (previously unknown) value associated with that individual, thus compromising the database.

This is a general model for compromise attempts of any statistical database. Two special cases appear to contain all published methods for penetrating the security of a statistical database. They are as follows:

Case 1: \(\ell = 1; \) \(f_1\) is a linear combination of terms which involve only responses to average and sum queries.

Case 2: Each \(B_i\) can be written either as \((p = q) \wedge B'_i\) or as \((p \neq q) \wedge B'_i\), where at least one of \(p\) and \(q\) is a response to a count or percentile query. In this case, for each condition,
the Boolean conditions are used to detect equality of responses and the known overlap between response sets is used to determine the value of \( I \).

The above inferential scheme, taken from *Beck80*, illustrates theoretically the basic principles of compromise techniques. The intruder either isolates the desired piece of confidential information by appropriate exclusions and inclusions in the queries or solves a system of linear equations with the desired data elements as the unknowns. Most known security penetration attempts are variants of this idea. Dissappointingly, the simplicity of the compromise techniques is not shared by protection methods.

When the responses from the system are distorted, the above principle expands naturally. Suppose that the variation applied to a response is dependent on some attribute \( V \) of the query \( Q \) (or of the answer). The user may ask a series of queries \( Q_1, \ldots, Q_k \) which change the attribute \( V \) while having a known effect on the true answer. When the variations applied for \( Q_1, \ldots, Q_k \) are independent, simple averaging allows an arbitrary degree of accuracy in the estimation of the target element. When the variations are correlated, because \( V \) reflects in some way the information contained in the true answer, a different series of queries \( Q'_1, \ldots, Q'_k \) with known answers and attributes \( V'_1, \ldots, V'_k \) that are related to \( V \) (the \( Q'_i \) are related to \( Q \)) allows a correlation estimation which in turn yields information on the variation in the response to \( Q \).

There is not only a tremendous abundance of snooping tools but they also seem to be applicable under presumably prohibitive conditions. Ironically, most of the human recourses expended on research in this area have significantly contributed to the variety and versatility of such tools. The hope remains that by better understanding compromise techniques we will eventually succeed in developing adequate systems to counter them.

Most simple examples of database compromise are key dependent. That is, they depend on specification of the individual object to be compromised by a key in the database. The query set can be determined either by a list of keys of the database or by a characteristic formula. In the first case, the query set is formulated by matching a list of keys (provided by the query)
with the whole database and the query is then called key-specified. In the second case, the characteristic formula (which is an arbitrary logical formula) is used to construct the query set and the query is called characteristic-specified. In these terms, our observation can be restated as: Most simple database disclosures occur for key-specified queries.

This partially true observation has lead to comparisons of the research in both directions. Historically, key-specified queries and their security have been treated first. Not only were the results not promising but this type of queries was found to be the least interesting and not significant. Statistical databases do not usually produce data on particular individual records. To use the keys effectively one needs to know the exact correspondence between keys and identifying characteristics. Under different conditions it may be harder or easier to achieve compromise with one type of queries or another. In the majority of cases, compromise is easier with key-specified queries. Consequently, most recent work focuses on characteristic-specified queries.

With the exception of the first section, the next few sections discuss proposed inference controls. They are divided into two broad categories, data suppression and data transformations. Data suppression methods include an operating systems approach, limitations on the query types and random sampling. They are techniques that restrict the information released. Data transformations include data distortion, data rounding and data swaps. These are techniques that modify information before it is released. In the literature, the terms data transformations and data swaps are often used to describe particular data distortion methods. We make this further distinction.

5. The Kam–Ullman Model

Kam and Ullman [KaUl77] have presented and examined the security of key-specified queries. Each database record is completely specified by a \( k \)-bit key and a database is viewed abstractly as a function from these \( k \)-bit strings to the integers. An \( s \)-query is a string of bits that contains exactly \( s \) 0's and 1's; clearly, \( 0 \leq s \leq k \) and the remaining \( k - s \) bits of the query
are nonimportant. The $k - s$ matched keys form a cube of dimension $k - s$ in the Boolean $k$-cube. Responses are also restricted to sums. Observe that this is equivalent to a model with averages as its basic operation only when the database consists of exactly one record for each key (in this case the size of the query sets is easily found to be $2^{k-s}$ records). Also, observe that this model carries severe restrictions. The sum of the information over the query set and the cardinality of this same set cannot be given out simultaneously [Chin78]. The assumption of exactly one record per key is stringent and limiting (the brief discussion of the sparsity issue in the paper does not remove this limitation). The resulting theorem is that if $s \leq k$ no database is compromisable by $s$-queries given that the range of the database values is not restricted. This last condition proves crucial. Once the range of the database values is known (recall that a database is viewed as a mapping to $\mathbb{Z}$), compromise is possible by $(k - 1)$-queries.

Yao has strengthened the noncompromisability result by confirming a conjecture stated by Kam and Ullman [Yaoa79]. Compromise in the original paper was exclusively the inference of a single database element. Yao established that there are no revealable secrets in the database, which assume the form of rational constraints on less than $2^{k-1}$ database elements. The simplicity of this result (Kam and Ullman provide a lengthy proof of a special case) and the elegance of its proof make it an attractive theorem, although mainly of theoretical interest.

Along the same lines, Chin offers a significantly improved version of the model (avoiding some of the above limitations and considering characteristic-specified queries) and shows that confirming the existence of any particular record in the database is equivalent to deducing the existence of all records. Prior knowledge of one record suffices for full compromise! [Chin78].

6. Physical Separation

Physical separation of the privileged information from the nonprivileged information is the most obvious and conceptually simple protection scheme for a database. This is, however, a tool borrowed from security methods in operating systems and its effectiveness is doubtful (as we have already seen, such methods are inadequate for the purposes of database security).
The principle is simple in that "the privileged data cannot be accessed at all without proper authorization or the privileged information is encrypted in such a way that only authorized users can decode and read it" [Reis79]. But we pay a high price for this simplicity. When the privileged information is encrypted, decoding procedures must be supplied to the authorized users. Such procedures obviously involve development of specialized software, which takes its toll in costs and complication of the user-system interface. More significantly, separation violates principles of efficient storage (at a minimum, unless an elaborate indexing system is implemented, headers and database keys must be repeated in both storing units). Finally, the greatest drawback of this confidentiality protection scheme lies with the fact that it is too severe for a large many applications. Occasionally, data about individual objects is privileged but statistical information concerning a large sample of such objects is not privileged and should be accessible to all users [Reis79]. In such instances, the suggested scheme is not only inappropriate but also hinders the use of the database.

7. Partitioning of the Database

A different method for protection of sensitive information in a database is introduced in YuCh77. The rationale behind this proposal is intuitively appealing, fundamentally sound and rather simple. As we have seen, a variety of compromise techniques are based on the combinatorial principles of inclusion and exclusion. The intruder's goal is to isolate a certain record in the database and deduce a piece of confidential information from that record. To eliminate such record isolation attempts, Yu and Chin suggest that the database be stored in groups, each one of which contains at least some predetermined number of records. Queries may be then applied to any one group or sets of groups but never to subsets of records within a particular group. (A variant of this proposal is the "microaggregation" scheme where groups of individual records are aggregated into synthetic "average individuals" and statistics are computed for the synthetic individuals rather than the real ones. This model proves extremely useful in the unweighted means approach to the analysis of variance of nonorthogonal designs, provided the sizes of groups meet certain conditions).
Clearly, this mechanism points at the heart of the matter. Questioners can only obtain information about a whole group, and are never able to break down this information in order to make inferences about individual records. This control is efficient in insuring the confidentiality of individual records. Moreover, it appears promising that an extension of this idea (i.e., an appropriate group selection) may efficiently protect the confidentiality of any general database secrets (i.e., secrets of the database about more than a single individual record). To the best of our knowledge, however, such an extension has yet to surface.

Nevertheless, this is not a practical safeguard. Even in the case in which the grouping of the records in the database arises naturally (say, according to an unambiguous classification of the records in a number of statistical subpopulations), this approach may severely obscure useful statistical information in the database. Generally, the legitimate free flow of statistical summaries can be inhibited by excessively large groups or by ill-considered groupings. Often the goal of statistical analysis is precisely the identification of existing subpopulations within the given population (cluster analysis, recursive partitioning and discriminant analysis are examples of statistical techniques devised for this purpose). Any forced grouping of the data may either render the subsequent statistical analysis impossible or its results unreliable. Additionally, the bookkeeping needed for the implementation of this scheme is extensive. Insertions, updates, and deletions will result in forming, revising, and reforming groupings which can be a costly procedure. Note that the latter objection becomes legitimate only for dynamic database applications. Other proposed security mechanisms may suffer a similar inadequacy which is never raised because static models are employed. On balance, the impracticality of the technique overshadows its robustness, yielding it infeasible and rarely implemented.

8. Query Limitation

The tracker, a concept invented by Schlörer is the most famous security penetrating device. Due to its nature, it is a devastating snooping tool that circumvents the most intuitive protection schemes, limitations on the size of the query sets.
The original work by Schlörer [Schl75] demonstrates how databases can be easily compromised by characteristic-specified queries even if some queries are not answerable because their query sets (or the complements thereof) are too small. "The questioner divides his preknowledge of a given individual record into parts, which are then reassembled into a special characteristic formula called a tracker. From the responses of a few answerable queries involving the tracker, the questioner may determine whether or not the given individual has a characteristic previously unknown to the questioner" [DeDS79]. In other words, whenever the system responds to queries with corresponding query set cardinalities in the range of \([k, n - k]\) (where \(n \geq 2k\) is the number of records in the database and \(k\) is the minimum cardinality of a query set for the query to be answerable), the tracker can be employed to achieve compromise provided there is some prior knowledge of a given record. The example in Section 3 above is an example of a tracker. The following theorem formalizes this concept.

Theorem. Let \(F = A \land B\) be a formula that identifies individual record \(R\) of the database and assume that \(T = A \land \bar{B}\) is a tracker for record \(R\). We calculate with three answerable queries:

\[
\text{Count}(F) = \text{Count}(A) - \text{Count}(T)
\]

\[
\text{Count}(F \land \rho) = \text{Count}(T \lor (A \land \rho)) - \text{Count}(T)
\]

If \(\text{Count}(F \land \rho) = 0\), \(R\) does not have characteristic \(\rho\). If \(\text{Count}(F \land \rho) = \text{Count}(F)\), \(R\) does have characteristic \(\rho\). If \(\text{Count}(F) = 1\), arbitrary statistics about \(R\) can be computed from \(q(F) = q(A) - q(T)\) (\(q\) represents a query).

The theorem and its proof are found in DeDS79. We call this kind of tracker the individual tracker. The idea's dependence on prior knowledge about the particular individual target record is of little consolation. This dependence is removed by Denning, et al. [DeDS79], by introduction of the general tracker. Furthermore, their paper generalized Schlörer's work in a different direction. They showed the tracker to be applicable not only to count queries but to any statistical queries.

The general tracker works for any system that responds to query sets with cardinalities between
$2k$ and $n-2k$. Clearly, $k$ cannot exceed $n/4$ for the existence of a general tracker to be possible; when $k = n/4$, the cardinality of the query set must be $n/2$ for a general tracker to exist. A theorem analogous to the one stated above can be proved. The advantage of the general tracker is that it suffices to answer all compromise questions. One does not have to construct an individual tracker for each record of interest.

It is easy to construct examples of compromise where the general tracker is not needed. Its definition is a sufficient condition for compromise, but is stronger than necessary. Databases, even when secure from the general tracker, may be susceptible to an individual tracker attack and may still be compromisable. The double tracker (requiring $k \leq n/3$) [DeDS79] and the union tracker [Schl80] are two different forms of trackers. The necessary and sufficient conditions for the existence of these trackers are derived in the corresponding references.

Any database with at least $2k + 1$ distinguishable classes of records has a general tracker. The probability that a general tracker exists in a database with $n$ records converges rapidly to 1 as $n$ goes to $\infty$. In this context, for many systems (including System R and INGRES) the intractability of the problem of a tracker construction appears to be the only hope for a positive step towards security. It is, however, a false hope. Under the unrealistic assumption that the whole database can be inspected, DeDS79 supplies an $O(n^2)$ algorithm for constructing a general tracker. Under the assumption that the user knows only the domains of the attributes and has no prior knowledge regarding the contents of the database, a better algorithm is given in DeDS80. Provided that the maximum number of identical records is not too large, this algorithm produces a tracker in $O(\log_2 S)$, where $S$ is the number of distinct records possible. Strictly speaking, the procedure depends on knowledge of $k$ and $n-2k$. This is not a limiting assumption because protective measures (such as $k$) that must remain concealed themselves are of dubious value [Schl80]. Even this minor obstacle can be overcome. The paper suggests a modified version of the algorithm that finds a tracker by estimating $k$ and $n-2k$.

In sum, limiting the class of answerable queries on the basis of the cardinality of the query sets is of little use. Although the existence of a tracker in a database must be coupled with enough
information about an individual record in order to lead to compromise of the particular record, trackers constitute a real threat to database security. The rigorous mathematical formalization of the tracker appears in Schl80: Even a random search finds a tracker. Although no upper bound is derived for the complexity of this random search, experimental results indicate that the solution may be devastating. In probabilistic terms, trackers exist with probability close to one for almost all databases. Note that the quick success of the random search prevents monitoring. Even if the system is monitoring queries to detect a compromise attempt, it is highly unlikely that it will be alarmed by the few queries needed to produce a tracker.

Controls that limit the overlap between query sets can be effective as a protection from trackers. They are, however, infeasible and extremely expensive to implement, because every new query must be compared to all previous ones to check whether illegal overlap takes place. They can also be overly restrictive for applications which require comparisons of a subgroup to the whole population. But our main reservation is their vulnerability to combinatorial attacks.

Considering key–specified sum queries of fixed length, Dobkin, et al. [DoJL79] introduce a measure of security for a database \( D \), namely the minimum number of queries needed in order to achieve compromise. Formally, once we have determined the set of allowed queries on the system, \( Q \), and the set of allowed query sequences \( \bar{Q} \), we can define the security problem as the triplet \((D, D_0, \bar{Q})\). \( D_0 \) stands for the amount of prior user knowledge. Note that we need to specify the set of permissible sequences \( \bar{Q} \), because \( \bar{Q} \neq 2^Q \). The overlap control makes some sequences illegal and thus, dissallowed [Reis79]. Then, we denote by \( S(n, k, r, \ell) \) the security measure. Here, \( n \) is the size of the database in records, \( k \) is the number of elements involved in the sum requested by a query, \( r \) is the allowed overlap, \( \ell \) indicates that \( \ell \) data elements belong to the user's body of prior knowledge, and \( S(n, k, r, \ell) \) represents the minimum number of queries needed to compromise the \((\ell + 1)\)th data element.

A lower bound for \( S(n, k, r, \ell) \) is given in Reis79:

\[
S(n, k, r, \ell) \geq \frac{2k - (\ell + 1)}{r}
\]
and an even tighter bound for $S(n, k, 1, \ell)$ is derived in Leeeu79:

$$S(n, k, 1, \ell) \geq 2k - \ell, \quad \ell \geq 2.$$ 

These bounds are based on the non-triviality assumption: $k > \ell + 1$. It is desirable but difficult to actually prove of disprove an optimality property for this last bound. $S(n, k, r, \ell)$ is intuitively attractive because it is monotonic in $n, k, r$. Observation shows that

$$S(n, k, 1, 0) = 2k - \ell$$

and that

$$S(n, k, 1, 1) = 2k - 2.$$ 

In fact,

$$2k - 2 \geq S(n, k, 1, \ell) \geq 2k - \ell.$$ 

for $\ell \geq 2$ and $k > \ell + 1$ follows readily from our discussion. However, it is surprising that $S(n, k, 1, 2) = 2k - 2$. This means that restricting the users to an overlap of one record makes compromise with two previously known elements as difficult as compromise with one previously known element, although it is easier to achieve compromise when we know one element beforehand than when we do not know any. Because the exact behavior of $S(n, k, 1, \ell)$ remains unknown, it is unwise to attempt any explanations. Reiss notes that $S(n, k, 1, \ell)$ depends on purely number theoretic connections between $k$ and $\ell$ [Leeeu79]. This could be a possible explanation for this strange phenomenon, since 2 appears to be a special case in all of number theory. The only known fact is that, with fixed overlap, we can actually achieve the above lower bound for infinitely many $k$, provided that at least one database element is known.

An upper bound for the security measure can also be obtained under suitable conditions. The result is that when $n \geq k^2 - k + 1$, compromise can be achieved in linear time without prior information, even with the overlap limited to one record. When $n \leq k^2 + k$ compromise is impossible. A similar conclusion can be drawn for key-specified selection queries, that is queries requesting the median or any order statistic of fixed size subsets of the database, even when
the system can return an arbitrary value of the query set and not necessarily the requested statistic. The only necessary assumption is that no two individual records have the same data value [Denn78].

Unfortunately, combinatorial techniques are not the only methods for subverting a security scheme that enforces overlap limitations. Linear queries with arbitrary weights on a system that only allows for overlap of size one can be used to penetrate the security mechanism. For fixed size key-specified queries with the weights unknown, knowledge of one element in the database leads to full compromise. Assessing a user's prior knowledge is difficult. In fact, most users do know something about the database and the necessary condition for security, no prior knowledge, is rare. Moreover, it has been shown that partitioning the database elements into classes and applying linear queries that contain one term from each class (here, each weight corresponds to one class) is not safe either. It suffices to know all the weights and one data element in all but one class or one element from each class and all but one of the weights in order to fully compromise the database. The only secure state is when the questioner knows all the weights, one data element from all but two classes and does not know any elements from the remaining two classes. In this case, the two "unknown" classes are secure. Of course, the "known" classes are still fully compromisable [ScDD79].

We have explained why limitations on overlaps and the cardinalities of query sets do not obstruct the intruder. The careful reader might object that the arguments revolve around fixed size query sets. Although true, this objection does not invalidate our discussion. W. de Jonge [Jong83] presents a simple technique that accomplishes compromise for systems that limit the size of their query sets and answer queries about means on sets of variable size. In this context, de Jonge provides a natural extension to the concept of the tracker. We can, therefore, conclude that query limitations are essentially dubious protective measures. They may impair the utilization of the database, while they do not offer reliable security.
9. Distortion and Statistical Security

Most important among the different protection schemes in the category of data distortion are those that introduce some random perturbation to the data in order to enforce the confidentiality constraint. The striking result that essentially unifies all of these approaches is that statistically we can achieve arbitrarily satisfactory levels of security. As long as the database system distorts the data in a statistically acceptable way (e.g., the answers to the user’s queries remain unbiased estimates of the true responses), any security mechanism can be penetrated. But penetration requires an exponential number of queries (determined at will by the proper authority) and it is thus, infeasible. Of course, one questions the significance of such a claim. The precision of the concept of statistical compromise and security remains an issue as does the assessment of the value of such techniques. If, after all, we have managed to obtain arbitrary degrees of protection (according to our desire in any given situation) we have arrived at the definitive solution to the problem. We address these questions below, by a parallel analysis of two pertinent papers.

Our simplified model will be the following: In a database $D$, with $n$ individual records, “random” perturbations (chosen appropriately) with a certain variance are added to each element every time the element is used in a query computation. The perturbations are taken to be independent and identically distributed. We suppose that a user attempts to achieve compromise of $D$, by trying to extract the value $y_i$ of some confidential data element in $D$.

Beck80 defines $D$ as compromisable if the user is capable of obtaining an estimate $\hat{y}_i$ of $y_i$ such that $\sigma(\hat{y}_i) < f(y_i)$, where $\sigma(\hat{y}_i)$ stands for the standard deviation of the estimate $\hat{y}_i$ and $f$ is a function of the true value $y_i$ (and other database parameters). This seems to be the most reasonable definition of statistical compromise. It also appears in disguise in TrWY84 and it is undoubtedly sensible. The trick is to use Chevychev’s inequality to bound either the variance of the user’s estimator [Beck80] or the probability that the error of the estimator will be within a certain range (say smaller than $\epsilon$) [TrWY84]. We then control the degree of difficulty for any compromise attempt by tightening the bound. In theory, statisticians
would like small errors in the quantities they compute while database managers would like large individual perturbations. A suitable choice for the variance of the random perturbations makes compromise extremely expensive, resulting, however, in an increase in the variance of the computed statistics.

From a purely statistical viewpoint, this approach stirs tremendous controversy. It demonstrates the applicability of statistical methods and provides a valuable and powerful tool that leads to mathematically sound solutions of a difficult problem. On the other hand, it seems to contradict the very goal of statistics.

To substantiate this view we provide a simple but very interesting generalization in two directions of the method adopted in TrWY84. The authors are content with treating the problems only for sum queries and their approach is based on a Chebychev bound for the probability that the error is smaller than some constant. In their notation, let \( q_c(d) \) be a query for a subset \( c \) of \( \{1, 2, \ldots, n\} \) with \( d = (d_1, \ldots, d_n) \). Perturbing \( d \) by \( e = (\epsilon_1, \ldots, \epsilon_n) \), the system returns \( q_c(d') \) with \( d' = d + e \). The pivoting probability is \( P\left\{ \frac{1}{|c|}|q_c(d') - q_c(d)| < \epsilon \right\} \). We can use an asymptotic expansion for the error term \( |q_c(d) - q_c(d')| \), so that we can allow queries to be any smooth statistics and not only sums. Restricting ourselves to sums, however, it is likely that sharper results may be obtained by using a central limit theorem approximation to the term

\[
\frac{1}{|c|}|q_c(d') - q_c(d)| = \frac{1}{|c|} \sum_{i \in c} \epsilon_i
\]

in order to get an actual estimate of the probability under construction. This way we free ourselves from the useful but limiting Chebychev bound.

These are the gratifying aspects of the approach. But, given that statistics seeks to filter noise out of real data, the idea of distortion seems contrary to the goal. It is not clear that such a violation of first principles is tolerable. The disappointing aspect of the approach is that eventually the database will be used for statistical purposes, and intentional obscuring of data will thus have adverse consequences.
The objections to the technique of data distortion are straightforward. The most important objection is the introduction of unnecessary errors in the calculations. Sometimes these errors can be extremely large even if there is no real threat of compromise [TrWY84]. Note that the additional constraint that these artificial perturbations remain small at all times (so that our calculations are not seriously affected) backfires. An intruder can achieve statistical compromise of the database by estimating any one of its elements with a small (negligible) error. To further complicate matters, note that perturbing final statistics (after the actual calculations) seems the right way of perturbing the answers. It insures that errors will not produce some kind of catastrophic effect on the result by being propagated in the computations. However, such perturbations do not necessarily guarantee adequate security [Beck80, DeDe79] and thus, one must resort to individual perturbations which in turn can lead to numerically disastrous results. This error inoculation control has adverse consequences if the actual data is supposed to be available for authorized users. The objection that identically distributed perturbations do not provide confidentiality of outlying observations can be easily overcome by introducing perturbations dependent on the data values. This can be achieved either by appropriately choosing the function $f$ [Beck80] or by making the perturbations multiplicative rather than additive [TrWY84]. Of course, distortion schemes based on perturbations independent of the queries are equivalent to schemes without distortion. Simple estimation techniques will yield correct estimates of the noise intended to mask the responses (either directly or by estimating the autocorrelation of the noise) and compromise results. True randomness in the introduced perturbations may result in inconsistent statistics and may reduce the user–system interface to a procedure equivalent to experimenting because the same query may be answered differently at different times. Pseudorandomness is, therefore, preferred. Lastly, it remains unclear how to extend data perturbations to categorical (nonnumeric) variables. In sum, the inadequacy of data distortion oriented inference controls stems from the inaccuracy in results that cannot be justified by the provided guarantee of security!
10. Random Sampling

Random sampling was first introduced formally as a security mechanism [Denn80]. At that time it was hoped it would become the most effective security scheme available to database designers. The idea consists of applying queries to random samples from the database and not to the entire database. These random samples are chosen by a suitable mechanism implemented by the database management system. This random selection of query sets, a process which is beyond the questioner’s control, potentially deprives intruders of their capacity to structure queries in such a way that the sequence of queries will eventually attack a desired target. The 1960 U.S. Census is a handy example of application of random sampling. It was distributed on tape as a random sample of one record in one thousand. Each sample contained no name and specified location only by size of city in one of nine geographic regions. Therefore, compromise occurs with probability smaller than 1/1000.

The motivation for random sampling is well-grounded. Most inference controls are subverted by a single basic principle of compromise: Control of the composition of each query set leads to the isolation of a single value by intersecting query sets. Random sampling essentially denies the user control of the composition of the query sets.

The proposal of Denn80 substantially improves the Census Bureau procedure with the following modification: As the query system locates records satisfying a given characteristic formula $C$ (i.e., forms the query set) it applies a selection function $f(C, R)$ to each record $R$ in the query set; $f$ determines whether $R$ is used in the sample (a parameter $p$ specifies the sampling probability of a record selection) and this selection process produces a sampled query set $X^*_c = \{ R \in X_c \mid f(C, R) = 1 \}$ ($X_c$ is the query set). The statistic returned to the user is calculated from $X^*_c$ [Denn80]. Provided each sample contains a large proportion of the records in the query set, this procedure insures accurately and timely statistics, while it keeps the implementation costs low (by using only a part of the database).

Some subtle issues are too detailed for the purposes of the present discussion. For example, the sampling probability of a record selection $p$ must depend on the size of the query set, a large
leads to a relatively routine compromise, and the choice of $f$ is important. The attractive result is that the relative errors in the statistics returned (frequencies and averages) decrease as the square root of the query set size, while simultaneously the complexity of the compromise effort increases linearly in the cardinality of the query set.

Nevertheless, there are two major limitations. First, the method cannot be used for small to medium size databases for the obvious reason that the statistics computed will be statistically insignificant because of the small sample sizes. To tailor the method to such databases, it has been suggested that random sampling be combined with a minimum query size control. Second, a computer may be an efficient tool in circumventing the difficulty of the large number of queries needed to achieve compromise. A systematic query generation may very well lead to compromise, especially since alternate formulations of the same query may yield different answers. The system must be sophisticated enough to reduce all queries to normal form in order to neutralize this anomaly. However, Denning writes in [Denn80]: “Unfortunately, the problem of reducing a formula to a normal form is intractable; even if an efficient algorithm could be found, there are other methods for removing the sampling errors”. For these reasons, this alternative has been ignored as a possible inference control, despite its appealing combination of a low risk of components and small relative errors in the desired statistics.

11. Data Transformations

Lastly, we turn out attention to the most promising approach to the problem. A basic question in statistical database security is to determine whether or not a given statistical output will permit deduction of the original database contents. In a way, the approach of data transformations reverses the problem. It addresses the question of determining the necessary and sufficient conditions under which a statistical database $D$ can be transformed to another database $D'$, while preserving the statistics of interest [Schl81]. To paraphrase, suppose we are given some statistical output $O$ from a database $D$. We wish to find out if it is possible to construct a different database $D'$ that will produce an identical output $O$. If so, we would like to know
what conditions on \( D \) make this transformation \((D \text{ to } D')\) possible and whether there is a general algorithm that will produce \( D' \) from \( D \), given \( O \).

By construction, release of \( D' \) yields accurate statistics and a high degree of confidentiality. "The risk (of compromise) is not zero. Given sufficient supplementary knowledge and vast resources, it may theoretically be possible to determine \( D \) from \( D'' \)" [Schl81]. But there may be many more than one candidate for \( D \) and the intruder will have to choose the right one, an unlikely event. For protective purposes, some transformation does the job, whereas for an intrusion only the right inverse transformation works. In addition, partial transformations can be used for the same purposes. While the method avoids many of the problems existing in previous protection schemes its major drawback is that we have yet to produce an efficient algorithm that will carry out the task. It is strongly suspected that this is not a tractable problem. The conjecture is that constructing an exact transformation is an NP-complete problem. On the other hand, there is a suggestion for a heuristic nonoptimal transformation that might perform satisfactorily [Schl81]. We would also like to obtain simple criteria for transformability and this is still an open problem. The mathematical foundations to this approach of exact transformations have been set by Schl81.

As an alternative to exact transformations, Reis84 suggests approximate data swaps. The idea is the same, but the requirement of exactness of the transformation is relaxed. In Reiss' terms, "A data-swap is a transformation that maps the original data into a new database which exhibits the same statistics. We can parametrize this mapping by characterizing the part of the database that is being modified and the set of statistics that is being preserved". This is basically a formalized definition of the data transformation above. An approximate data-swap does not guarantee that the statistics will be preserved; it only insures that the statistics of the new database will be close to those of the old one.

The issue of "how close is really close" is not satisfactorily answered in the paper. The important advance is that eliminating the exactness requirement allows the satisfactory construction of ad hoc algorithms. Reiss, in particular, presents such an algorithm (with some variants)
that creates a new data matrix $D'$, from an old data matrix $D$, and approximately preserves the $2 \times 2$ frequency tables. He assumes that his data matrix consists of 0's and 1's, but this is only a simplifying assumption. The approach can be extended to any categorical database with a finite attribute domain and to frequency tables of any dimensions. It remains to be seen how to tailor this method for noncategorical databases.

The paper includes a simulation study that was performed in order to evaluate the method. The results are encouraging. The proposed concept of error in the approximations of the statistics is reasonable, but a satisfactory answer to how small an error should be considered acceptable is lacking. The theoretical results on the performance of the algorithm allow for as much as 50% error. This worst case behavior is exemplified by an easy construction. Naturally, this bound is disappointing, for 50% error might render the new database statistically worthless. The only really positive result is that simple frequency counts are preserved by the given algorithm when the algorithm is based on $2 \times 2$ frequency tables.

The basic idea of the algorithm is as follows: Given the original database $D$ (which is a matrix) and the $2 \times 2$ frequency tables, for all the combinations of variables, to be preserved, a new database (matrix) is constructed. The new matrix is constructed sequentially. There is a major problem. Except for the first column of the new matrix, the remaining elements are chosen by an ad hoc procedure that cannot be statistically justified. It involves a crude probabilistic estimate of $P\{X|X_1, \ldots, X_n\}$ by $\frac{1}{n} \sum_{i=1}^{n} P\{X|X_i\}$; only complete statistical independence allows for such a substitution, in which case the two quantities are trivially equal. Otherwise, some sort of a weak Markovian assumption is needed but the form of such an assumption is unclear and not provided in the paper. This discrepancy can be partially rectified by heuristic correction techniques suggested as variants to the main algorithm. The running example in the Appendix of the paper seems to be unaffected by the inconsistency in the algorithm, but a few examples are hardly justifications.

We propose two ways to improve the method. Neither is exact in its details, but they are directions for future work. One improvement is to estimate by the empirical distribution the
statistical quantities to be preserved. The estimation will be based on the original data. Then, using the bootstrap we could create a new database that would approximately conform to the statistical quantities but would be, in a way, random. The second improvement is directly geared towards the given algorithm. Joint conditional probabilities can be estimated from the empirical distribution. We can use these estimates to pick the elements in the new database in a more statistically correct fashion. That is, we can condition on all the remaining variables, as opposed to conditioning only on the previous variables. In the algorithm, when elements in column \( k \) are to be chosen, conditioning is done only on variables \( 1, 2, \ldots, k - 1 \). We suggest that we condition on variables \( 1, 2, \ldots, k - l, k + 1, \ldots, m \) (where \( m \) is the total number of variables) in order to choose column \( k \) in the new database. Also, we should use actual joint conditional probability estimates from the empirical distribution, not simple averages of single conditional probabilities.

In summary, we believe that data transformations either exact or approximate are the best method for database security in that they provide adequate statistical database security, while insuring accurate statistics.

12. Conclusions

We have attempted to take a comprehensive look at the problem of inference control and its suggested solutions. This paper is not, by any means, exhaustive. There has been an overwhelming amount of research and literature on the subject and it is impossible for any one paper to capture all the subtleties and complicated issues. Our approach has been to briefly explain why the methods of inference control are inadequate, and critically compare them. We conclude that data transformations are the right direction at least within the current state of research. Because of the difficulty of the approach there are not many papers on data transformations. The existing literature requires enhancement.

Some inherent problems with the approaches discussed are the following:

(i) All previous studies have considered static databases in order to simplify the problem.
(ii) The term “statistical database” has been used in substitution of “statistical file”; most authors actually mean the latter when they refer to statistical databases.

(iii) Users may be equipped with information other than what is explicitly in records.

(iv) The approaches have dealt with arbitrary query sets, which leads to an explosion in complexity. Statistics about particular query sets may be meaningless and as such may never be requested.

These observations suggest that a) a proper definition of statistical information must be found in order to reduce the magnitude of the problem, b) the problem should be investigated for dynamic databases, c) the possibility of enforcing security at the conceptual level of a database (rather than at the physical level) should be examined and d) the real user information should be used in order to obtain better safeguards. These considerations are explained in ChOz81, where the authors propose a statistical security database management system. This might prove to be the only alternative, if all the direct protection schemes fail. Unfortunately, the development of such a system is heavily dependent on the rudimentary, low level inference controls discussed here.

Acknowledgements

I would like to thank Professor Christos Papadimitriou for providing the motivation for this work, my friend Lee for her careful editing of the manuscript, and Karola for the \TeXing.
13. Bibliography


Olken, F. [1983]. How Baroque should a statistical database management system be? LBL-15765, Computer Science and Mathematics Department, Lawrence Berkeley Laboratory, University of California, Berkeley.


3. Tibshirani, R.J., Bootstrap Confidence Intervals, October 1984.