LBL-17413 Rev.
c. 2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

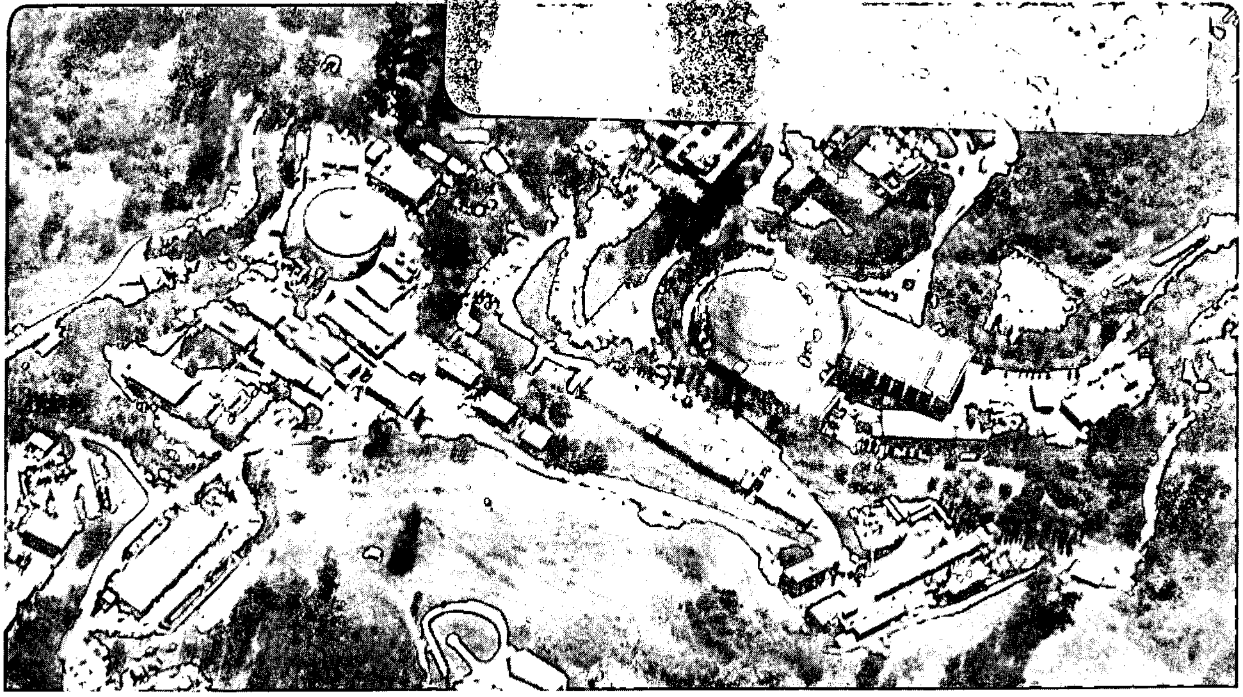
Computing Division

To be presented at the Second International Congress and Exhibition on Computer Security (IFIP/Sec'84), Toronto, Ontario, Canada, September 10-12, 1984; and to be published in the Proceedings

LOWER LEVEL INFERENCE CONTROL IN STATISTICAL DATABASE SYSTEMS

D.L. Lipton and H.K.T. Wong

May 1984



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBL-17413 Rev.
c. 2

LBL-17413

LOWER LEVEL INFERENCE CONTROL IN STATISTICAL DATABASE SYSTEMS

David L. Lipton
Harry K.T. Wong

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94704, U.S.A.

San Francisco State University
San Francisco, California 94132, U.S.A.

ABSTRACT

An inference is the process of transforming unclassified data values into confidential information. Previous research has studied the use of statistical aggregates to deduce individual data values.

Other types of inference are possible. Obscure correlations may be possible from an "expert" knowledge of the population. Inferences may be formed from "obvious" facts about the world. "Expert" users of a database management system may infer data values from variables in the system performance.

As a counter-measure, system management must consider these variables and correlations while formulating system security policy.

Research supported by U.S. DOE under contract No. DE-AC03-76SF00098

1. MOTIVATION

An inference is a process which transforms unclassified data values into protected data values.

Historically, the study of the inference problem in statistical database management systems, has been limited to threats which use statistical aggregates to identify individual record values. As a simple example, consider the following problem. A population database consists of four individual records.

Figure 1
Example of a Small Statistical Database

NAME	SEX	PROFESSION	SALARY
Dalton	F	EE	36
Jones	M	Programmer	20
Rankin	M	EE	18
Smith	F	Programmer	26

Let the salary attribute column be suppressed.

Release the following aggregate statistics:

SUM (Salary | Sex = all, Profession = all) = 100
MEAN (Salary | Sex = Male) = 19
MEAN (Salary | Profession = Programmer) = 23

By linear programming, Dalton's salary must be in the range [16,46]. Aggregates on additional attributes may be used to reduce this range further. A comprehensive survey of statistical inference methods and counter-measures is presented in [7].

This paper will identify several other types of inferences which may be formed in statistical database systems. Users who possess some preknowledge of characteristics of a population, may know that a strong correlation exists between the value of a released attribute and the value of a suppressed attribute. [22] Similarly, such correlations (or FUNCTIONAL DEPENDENCIES) may be apparent from "common knowledge". A third class of threats are possible from "expert" users of a database system, who form inferences from performance variations at the user interface.

Because these inferences require some preknowledge, they may be formed by a class of "expert" users at a lower level than that which is possible by the naive user of the database system.

The most important counter-measures to these threats are to expand the definition of "DATA OBJECTS" which are logically known to the database system. Database designers should become aware that uncontrolled access to these classes of data objects, is a security risk.

Section 2 will provide a theory for the lower level inference problem. Section 3 will illustrate applications of this theory to threats from unlimited access to database documentation in a menu-driven system. Section 4 will discuss applications to threats which are caused by unrecognized variable behavior at the user interface level. Section 5 will suggest some design principles for database management systems, to prevent user-written programs from generating inferences by covert information flow.

2. PARTITIONING A DATABASE BY SECURITY CLASS

This section will introduce a theory of how data objects may be assigned to PARTITIONS (or mutually exclusive groups), as a function of security class. FUNCTIONAL DEPENDENCIES (or correlations) between objects of different groups may exist. [4] One type of functional dependency (the EQUIVALENCE PATH) may allow users to infer the existence and value of objects in a partition to which access privileges have not been granted. [22] This section will present a series of transformations to optimize the partitioning scheme. The inference control problem will be reduced to the more manageable problem of regulating access to these newly-identified classes of data objects.

Consider a simplified READ-ONLY database. Let there be only two security classes: RELEASED and SUPPRESSED. [30] Each ATOMIC (or logically indivisible) data object is assigned to exactly one of these security classes. The assignment may be a function of both the general data object definition and the value of a specific instance. [29]

Let k_i be a data object which is recorded (or "logically known") in the data object dictionary and in the security subsystem files.

Let u_i be a data object which is not so recorded (or "logically unknown").

NOTE: The data objects k_i and u_i need not be logically indivisible. They may be vectors, matrices, trees, or networks of data objects whose meaning is semantically valid to the database system.

Let FD_i be the description of a functional dependency which maps data object k_i to a unique data object k_j.

$$FD_i\text{:}\ k_i \Rightarrow k_j$$

Let $\Rightarrow$ be a functional dependency which is known to the system security manager.

Let ---> be a functional dependency which is unknown to the system security manager.

Let the functions released(k_i) and suppressed(k_i) represent the partition in which a data object k_i is currently located.

The following series of transformations will eliminate all EQUIVALENCE PATHS (or instances where the value of a suppressed data object may be determined from released data objects). Additional transformations will illustrate ways to minimize DATA LOSS (or non-confidential data which must be suppressed to protect confidential data).

Suppose that the unknown data object u_i is in the released partition of the data object space. Suppose that a functional dependency exists from u_i to the suppressed data object k_j. This unsafe case may be represented as:
(i) released(u_i) ---> suppressed(k_j)

If the data object u_i is identified and cataloged in the files of the access control mechanism as k_i, then the following unsafe case is created:
(ii) released(k_i) ---> suppressed(k_j)

If an unknown functional dependency ---> is identified by the system security manager, then the unsafe case
(iii) released(k_i) $\Rightarrow$ suppressed(k_j)
is generated.

At this point, the space of data objects in the database may be repartitioned as either the safe case
(iv) released(k_i) $\Rightarrow$ released(k_j)
or as the safe case
(v) suppressed(k_i) $\Rightarrow$ suppressed(k_j).

Thus, the data object space has been repartitioned such that no EQUIVALENCE PATHS remain.

The following example will illustrate this process. It will also informally introduce some inference techniques which will be discussed in later sections.

Consider the record definition for a small personnel database:
< NAME, SOC_SEC_NO, MARITAL_STAT, DWELLING_TYP, CAR_VALUE, OCCUPATION, SALARY >

A specific application requires an EXTERNAL VIEW of this record in which the following attributes must be suppressed: NAME, SOC_SEC_NO, DWELLING_TYP, and SALARY. The system security manager must release as much information as possible without compromising the security of any suppressed data.

By statistical analysis of similar populations, it is discovered that a strong correlation exists as follows:

CAR_VALUE ---> SALARY

However, this analysis reveals that the "intuitively obvious" correlations:

OCCUPATION ---> SALARY
OCCUPATION ---> CAR_VALUE

are both much weaker for similar populations. Therefore, only CAR_VALUE need be suppressed at the attribute definition level.

It is also discovered that a strong correlation exists for

(MARITAL_STAT = single) ---> (DWELLING_TYP = apartment)

However, there are no dependencies observed between other values for MARITAL_STAT with any specific value for the suppressed attribute DWELLING_TYP. Therefore, suppress all attribute instance values for (MARITAL_STAT = single).

From "common knowledge", a user lists four possible values in the attribute domain for MARITAL_STAT:

MARITAL_STAT $\in$ {single, married, divorced, widowed}

By observing that no instances of (MARITAL_STAT = single) appear, the user may conclude that many of the suppressed instances have this value. As a counter-measure, the released attribute value with the lowest frequency must also be suppressed. Such a measure prevents inference and minimizes data loss. [26]

Until this point, it has been assumed that functional dependencies exist only between atomic data objects. It has also been assumed that any suppression may be performed if it is logically necessary. Two exceptions to this case will be presented.

1. If the WORK FACTOR required for a user to compute the INFERENCE $\Rightarrow$ in expression (iii) is of exponential complexity, then no repartitioning need occur. As an example, consider the case

(vi) released($< k_1, \ldots, k_n >$) $\Rightarrow$ suppressed(FD_1)

where FD_1 may either be an encryption algorithm or an authentication test. In this latter example, the released vector of data objects, may be a request for authentication and an enciphered record of the user's response. As further suppressions would be impossible, the major counter-measure is to increase the complexity of the inference. [19] [11]

2. If the released data object k_i is composed of several functionally independent data objects which are logically known to the database system:

$$k_i = < k_{i_1}, \ldots, k_{i_n} >$$

then system security policy need only suppress a "sufficient" number of $k_{i_l} \in k_i$, such that the functional dependency $\Rightarrow$ may not be computed. Symbolically, this repartitioning transforms the unsafe case

(vii) released($<k_1, \ldots, k_i, k_{i+1}, \ldots, k_n>$) $\Rightarrow$ suppressed(k_{n+1})

to the safe case

$$\text{(viii) } \max(i) \left\{ \begin{array}{c} [\text{released}(<k_1,\ldots,k_i>) \text{ and suppressed}(<k_{i+1},\ldots,k_n>)] \Rightarrow \text{suppressed}(k_{n+1}) \\ \text{and} \\ \text{not } [\text{released}(<k_1,\ldots,k_i>) \Rightarrow \text{suppressed}(k_{n+1})] \end{array} \right\}$$

for some $0 \leq i < n$. The problem of maximizing the number i of disclosed variables has been shown to be NP-Complete. [31]

Without loss of generality, the class of cases where the dependent data object is a vector of attributes without any functional dependencies between them:

(ix) $k_1 \Rightarrow < k_2, \ldots, k_n >$

may be treated as a separate case for each attribute k_i where $2 \leq i \leq n$. (Note that additional constraints should be introduced if any dependencies exist between elements in the dependent vector.)

The inverse of case (iv),

(x) suppressed(k_i) $\Rightarrow$ released(k_j)

is safe if the functional dependency $\Rightarrow$ is not reversible.

Thus, these transformations have reduced the inference threat to an access control problem. To insure that an access control subsystem is adequate to prevent security violations, the following assumptions must be valid:

1. All data objects in a database must be listed in the data object dictionary and in the access control subsystem file.
2. All functional dependencies between data objects must be known to system security management.
3. No suppressed data object is functionally dependent on a released data object. (All EQUIVALENCE PATHS have been eliminated.) [22]

Subsequent sections will suggest methods for identifying and cataloging unknown data objects u_i and unknown functional dependencies --->. Optimal suppression mechanisms will be presented to minimize the amount of unclassified data which has been falsely suppressed.

The theory of secure partitioning is sufficiently general to transform a wide range of known inference problems into access control problems. Some examples are presented to further illustrate its flexibility.

3. DOCUMENTATION

This section will present a survey of several types of inferences which may be formed from documentation in a menu-driven statistical database. This class of inferences is a function of data definitions and of the specific population represented in the database. Therefore, no security counter-measures may be implemented by the vendor of the database management system product. [17]

Counter-measures require that system security managers become "experts" on the population represented. This may require statistical analysis of databases of similar populations. System management may use expert systems to find new functional dependencies which are implied by record definitions. Additionally, system management should monitor any outside knowledge which specific users may have about populations or individual records.

Currently, there is a trend in the design of large statistical database systems to provide the user with as much documentation as possible about the population represented. Although such documentation (or METADATA) may be required for user-friendly on-line support, it also introduces a security problem. The secure design principle of LEAST PRIVILEGE requires that users be restricted to access requests within their NEED-TO-KNOW requirements. [25]

As an example, consider the SUBJECT statistical database system which was designed for public presentation of data from the 1980 U.S. Census. [2] This is a menu-driven system in which the user must specify the following type of access path to read a statistical aggregate value:
FILE_CATEGORY, FILE_NAME, ATTRIBUTE_DOMAIN_1, DOMAIN_VALUE_of_ATTRIB_DOM_1,. . .
ATTRIBUTE_DOMAIN_n, DOMAIN_VALUE_of_ATTRIB_DOM_n, AGGREGATE_FUNCTION_NAME
Extensive narrative documentation is provided for every node at every level. Several access paths may be possible to descend through the hierarchy to any node.

At each node level, it is possible in SUBJECT to read a narrative text file which contains information about all subtrees beneath a given node.

There is a browse function in SUBJECT which allows a user to randomly move up and down through levels of a menu.

Unlimited access to database documentation has been shown to create several new types of inference threats:

1. TYPE-R INFERENCES - "Obvious" functional dependencies between objects in different populations, or between attributes in a single record instance. [3]

EXAMPLE: There may be a functional dependency between the major businesses in an area and the most probable occupations of the area's population.

2. TYPE-S INFERENCES - Functional dependencies based upon knowledge of the hierarchical structure of the database. [3] This is a superset of the JOIN DEPENDENCIES in a relational database context. [5]

EXAMPLE:
COUNT(Male Programmers) = COUNT(Male Programmers over 30)
User Preknowledge: John Smith is a male programmer
Therefore, John Smith is over 30.

3. EQUIVALENCE PATHS - Users may have "expert" knowledge of a specific population. This may include knowledge of functional dependencies which exist in real data; but were not known to the formulators of system security policy. [22]

EXAMPLE: Pine Bluff, Arkansas has won the National Bowling Conference 25 times in the last 30 years. Therefore, an abnormally large percentage of the population has been operated on for dislocated shoulders. This inference may be formed without any knowledge of record instances of the form:

< NAME, ADDRESS $\in$ Pine_Bluff,Ark, NO_SHOULDER_OPERNS > 0 >

4. Users may form new statistical aggregate values by further analysis on individual records of the database system[1].

Several counter-measures to these threats are possible. Documentation for the database should be partitioned by NEED-TO-KNOW requirements. System security management should become aware of new functional dependencies which exist in real data. Statistical analysis and expert systems should be used to discover these new relationships.

Two types of documentation objects will be studied in this paper: NARRATIVE TEXT FILES and MENUS OF CHILD NODES. Appropriate security mechanisms will be presented for each type.

Documentation data objects for the database should be grouped into COMPARTMENTS by the access control subsystem. [17] These compartments should be based upon the users' NEED-TO-KNOW. [6] Within each compartment, a security clearance level may be assigned for the minimum read access privilege. As an additional constraint, an explicit access privilege grant may be required. In the latter case, the security class of the user and the data object become merely integrity constraints to determine if a grant should be issued. [18]

NARRATIVE DOCUMENTATION FILES should be segmented. Each partition should not discuss more than one child node in the menu. This prevents inferences which are based on the knowledge of nodes which are suppressed for a given user.

Compartments in unstructured text need not be separate paragraphs. Small, contiguous strings of text in a file may have different security values. Consider the security-oriented text processor proposed in the following example. [17]

This example will describe a text processor which allows database designers to prevent access to classified clauses in a sentence. A separate version of the sentence need not be stored for each security clearance level.

Assume that when a new computer account is issued, the user is assigned a vector of maximum read access levels for each NEED-TO-KNOW compartment:

< army_max_read_level, navy_max_read_level, air_force_max_read_level >

Each element in this vector is a non-negative integer. [6] [18]

The following code listing represents a text file "big_pine_2". The text of this file is one highly classified sentence from a Defense Department database:

"The combat forces deployed in the operation included 5,182 men of the 101st airborne rangers, the battleship New Jersey, and six A-7 reconnaissance aircraft."

The assembler-like language of this text processor, sanitizes clauses of the sentence by the user's clearance level. The result is a syntactically correct natural language text from which no inference of the suppressed clauses may be formed.

To understand the programming language used in this example, a brief introduction is provided.

The BEGIN and END Operators

Each block of text must be framed by a pair of operators <BEGIN,END>. The operand for each is text_type=text_name. To simplify this example, the only two text_types which will be used are "FILE" and "SENTENCE".

The COMPARTMENT Operator

Each clause within a sentence is assigned a COMPARTMENT (or topic designation) regarding the type of information it contains. The operands may be literals or logical expressions of literals. In this example, literals include "army", "navy", and "air_force". The operand "ANY" is the logical INCLUSIVE-OR of all literal operand values.

The SEC_LEVEL Operator

Each clause within a sentence must be assigned a minimum read clearance level for the compartments designated. If the user's security class is lower than this level, then the clause is suppressed. The operand of SEC_LEVEL is a non-negative integer. The highest values represent more sensitive information.

The TEXT Operator

The operand of TEXT is a literal string of printable ASCII characters which is surrounded by double quotes.

The CONCATENATION Operator

Suppressed text should appear invisible to lower level users. This operator inserts natural language conjunctions in text as appropriate for syntactic correctness. Examples of such conjunctions are "and", "or", comma(,), and semi-colon(;).

The Null or Continuation Operator (*)

When operands or comments use extra lines, the asterisk indicates this to the assembler.

Comment Fields

The semi-colon(;) must be the leftmost character of the comment field on every line.

Thus, the text file "big_pine_2" would be represented in this language as follows.

```
LABEL       OPERATOR         OPERAND       COMMENTS

big_pine_2  BEGIN            FILE="big_pine_2"
sent_1      BEGIN            SENTENCE="sent_1"
clause_1    COMPARTMENT      ANY
*           SEC_LEVEL        1             ;Allow read access for any user
*           *                  *           ;who has a read access maximum
*           *                  *           ;clearance level of at least 1
*           *                  *           ;in any NEED-TO-KNOW compartment
*           TEXT             "The combat forces deployed
*           *                 in the operation included"
clause_2    COMPARTMENT      army
*           SEC_LEVEL        2
*           TEXT             "5,182 men of"
clause_3    COMPARTMENT      army
*           SEC_LEVEL        1             ;SEC_LEVEL=2 > SEC_LEVEL=1
*           TEXT             "the 101st airborne rangers"
*           CONCATENATION    ",", "and"    ;link clauses to form
*           *                  *           ;coherent natural language text
clause_4    COMPARTMENT      ANY
*           SEC_LEVEL        1
*           TEXT             "the battleship New Jersey"
*           CONCATENATION    ",", "and"
clause_5    COMPARTMENT      air_force
*           SEC_LEVEL        4
*           TEXT             "six"
clause_6    COMPARTMENT      air_force
*           SEC_LEVEL        3
*           TEXT             "A-7"
clause_7    COMPARTMENT      air_force
*           SEC_LEVEL        2
*           TEXT             "reconnaissance aircraft"
*           END              SENTENCE="sent_1"
*           END              FILE="big_pine_2"
```

If user$_i$ wishes to read this file, the owner or administrator of "big_pine_2" must issue the command

PERMIT READ ACCESS ON big_pine_2 TO user$_i$

to update the access control subsystem files. [28], [18], [20] At run-time, the user types the command

PRINT big_pine_2

The following three cases illustrate the system's response as a function of the user's security class.

1. A user whose security vector is < 1, 3, 3, > will receive the text string:

"The combat forces deployed in the operation included the 101st airborne rangers, the battleship New Jersey, and A-7 reconnaissance aircraft."

2. A user whose clearance vector is < 0, 0, 1 > will receive:

"The combat forces deployed in the operation included the battleship New Jersey."

3. A user whose clearance vector is < 0, 0, 0 > will receive the message

"REQUEST DENIED".

This simple example of a security-oriented text processor, illustrates the power of a SANITIZER which is implemented below the level of expert systems. 50 distinct security class vectors are recognized for the file "big_pine_2". 12 possible combinations of clauses may be generated at run-time. These values are clearly a combinatorial function of the file length. Thus, the efficiency of not storing all gradients of sanitized text is realized for larger files. [17]

In a menu-driven, hierarchical database system, MENU ENTRIES should also be selectively suppressed by the user's NEED-TO-KNOW.

The following HIERARCHY OF ACCESS FUNCTIONS to an arbitrary menu entry, ranks access requests by increasing security clearance level. [17]

1. Suppress information about the existence of this node in its parent's narrative documentation file.
2. Suppress printing of this node in a menu of its parent's children.
3. Print this node, but suppress its narrative documentation file.
4. Allow only queries which do not disaggregate this node.
5. Allow printing of the node's menu of children.
6. Allow queries which disaggregate the node by reading aggregates about proper subsets of its children.

Several types of MENU ENTRY NODES are used in the SUBJECT database management system: groups of files, file names, lists of attributes in a file record, lists of values for an attribute domain, and aggregate statistics for a given value of an attribute. [2] Because this hierarchy of access functions is sufficiently general, it may be used for any of these node types.

Linear access hierarchies have been shown to be less realistic representations of the world, than partially ordered lattices of vectors of access privileges. [7] In most cases, users may not be implicitly trusted with all of the access privileges which are ranked below the highest function which they have been granted. However, the total ordering described above represents grants of successively greater knowledge about a subset of a given database. [17]

Another inference control technique, is the PARTIAL SUPPRESSION OF MENU ENTRIES. SUPPRESSION and AGGREGATION techniques may be successful if they do not force illogical ranges or illogical data distributions. [17] To motivate the need for a variety of node suppression mechanisms, figure 2 illustrates some classifications of attribute domains taken from Wiederhold. [27]

The following six menu entry suppression techniques may be implemented for appropriate types of attribute domains.

1. SUPPRESS ANY NON-ENUMERABLE CATEGORY NODE. This technique is possible for a domain of unrelated category names, if a count aggregate may not be logically formed at the parent node level. As an example, consider a domain which is a collection of files about logically unrelated topics. [17] (This domain is NON-ENUMERABLE because the user is unable to infer a complete list of all elements in the domain, if presented with any subset of the domain element menu list.)

2. SUPPRESS LOW FREQUENCY NODES. This is possible for a domain of elments which are NON-ENUMERABLE and cannot be ranked in an absolute linear ordering by magnitude (such as a list of surnames). Each menu entry in the list represents a set of individual records which have the given property. The QUERY SET SIZE THRESHOLD RULE is used to suppress groups which contain less than k elements or less that L% of the total number Σ of elements in the database. [8] [33] [34] A formal statement is:

```
IF $3 \le MAX(k, L\Sigma) \le group\_size \le MIN(\Sigma-k, (1-L)\Sigma) \le \Sigma-3$
THEN process a query about this group
ELSE {inference violation} suppress this group;
```

where k is the MINIMUM GROUP SIZE about which information may be disclosed,
L is the minimum fraction of database records per query set in the range $0 \le L < 0.5$,
Σ is the NUMBER OF INDIVIDUAL RECORDS in the database, or some local subset thereof.
(EXAMPLE: Σ may be the population of the United States, of a state, of a city, or of a census tract.)

The upper bound for GROUP-SIZE prevents inferences about small, excluded subsets.

The set of records in GROUP-SIZE may either be homogeneous, or may be the disjunction of all released sets.

The absolute bounds of 3 and $\Sigma - 3$ have been chosen because:
If $k = 0$, then a "NEGATIVE DISCLOSURE" will occur. (The knowledge that a set is empty may be useful to an imputer.) [32]
If $k = 1$, then an individual record is uniquely identified. [26]
If $k = 2$, then a user who has preknowledge of one of the individual records, may ask queries which reveal additional information about the other individual. [7]

Thus, the number k is fixed by public law or system policy as an arbitrary value which is believed to be the maximum number of records about which an individual may possess knowledge.

Figure 2
TYPES OF ATTRIBUTE DOMAINS
after [Wiederhold-1977]

SORTABILITY	MEASURABILITY	VALUE/ CHOICES	RANGE/ BOUNDING	SAMPLE DOMAINS	SUPPRESSION MECHANISM FOR DISAGGREGATE SUBSET
ranked	metric	continuous linear	known-finite	human weight	rolling up (or combining attribute values together)
ranked	metric	continuous linear scale	not assignable	net personal wealth {+ or -}	1.rolling up 2.suppression of extreme values
ranked	metric	integer	known-finite	human age	rolling up
ranked	metric	integer	upper bound not assignable	number of books read	1.rolling up 2.suppression of extreme upper values
ranked	ordinal	-------	-------	friendliness	1.rolling up 2.suppression of entire attribute
unranked	nominal	non-enumerable	logical dis-aggregation impossible	file (category)	suppression of any value possible
unranked	nominal	non-enumerable	logical dis-aggregation possible	surname	suppression of low frequency values
unranked	nominal	enumerable	-------	hair color	not possible; must suppress entire attribute
unranked	existential	-------	-------	sex	not possible; must suppress entire attribute

3. SUPPRESS NODES AT END(S) OF THE SORT. Possible for an ordered domain if the end(s) of the range are not "commonly known". [17]

However, the major limitation of partial suppression techniques, is that a user may infer the existence and magnitude of a suppressed node if

$\text{COUNT}_{\text{INDIVID\_RECS}}(\text{PARENT\_NODE}) - \text{COUNT}_{\text{INDIVID\_RECS}}(\text{RELEASED\_CHILDREN}) > 0$

If the description of the suppressed node is obvious in such a case, then one of the following mechanisms is appropriate:

4. FALSE SUPPRESSION. If only one menu entry node violates the QUERY SET SIZE THRESHOLD RULE, then suppress the two lowest frequency nodes. [26]

5. ROLLING UP (or INCREASING PARTITION GRANULARITY). In this technique, several similar domain elements are DISJOINED (or INCLUSIVELY ORed) into one menu entry. [24]
EXAMPLE: Compress a list of the 50 states of the U.S. into the following list of states and regional aggregates of states to suppress small aggregates for less populous states.
state = {California, Northwest, Southwest, Texas, Rocky Mountains, Mississippi Valley, New York, East Coast}

6. PROHIBIT DISAGGREGATION. If unreasonable disjunctions of elementary sets imply low elementary set values, then suppress the entry menu. Allow only queries which do not disaggregate the menu's parent node.
EXAMPLES:
hair_color = {[black, red, bald], [brown], [blond]}
human_age = {[0-5], [6-80], [81-100], [100+]}

At a higher level of control, system security managers should identify new functional dependencies which exist in real data. Such correlations may not be verified by the INTEGRITY ANALYSIS mechanism when new records are inserted into a file.

Some general techniques for finding these functional dependencies or EQUIVALENCE PATHS include:

1. System security managers should analyze databases of similar populations to find such dependencies. [17]
2. System managers should recognize "obvious" functional dependencies when partitioning database objects by security class. [3]
3. Users should be denied access to subsets of the database about which they have some prior knowledge. [3]
EXAMPLE: A user may not read census data about the census tract in which he lives.

To reduce the complexity of finding new functional dependencies in real data: [17]

1. The system security management may randomly select a small subset of the record instance population of a database. Correlations found in this subset may be assumed

valid for the entire database.

2. Accept correlations within a given confidence interval. Although a relationship in real data may not be valid for all record instances, it may be useful to an imputer.

Chin and Ozsoyoglu [3] have suggested that a theorem-prover be used to find inferences which are based on "well-known" facts. If feasible, expert systems should be developed and implemented as a required database design tool.

In conclusion, threats from uncontrolled read access to database documentation are dependent on the specific population represented. This class of inference control mechanisms cannot be implemented by the vendors of a database management system for all installations. Similarly, only measures to control the "obvious" functional dependencies may be implemented at the record definition level. The characteristics of each local population must be evaluated individually. [17]

4. USER INTERFACE

The terminal interface provides several variables which may assist "expert" users to form inferences about real data and about security subsystem data.

This category of threats has traditionally been associated with the COVERT INFORMATION FLOW problem. [16] Trojan Horse procedures and other user-written routines may cause the flow of data or analogs of data to a user interface. However, this section will present a unified approach to some unintentional design weaknesses in statistical database systems, which cause covert flow. Section 5 will present some database design principles to block the implementation of such user-written procedures at run-time.

The most sensitive data value that may be inferred through the user interface, is a bound for the size of a set of homogeneous records. A user may know that several queries were rejected because the group-size parameters were outside of the range imposed by the QUERY SET SIZE THRESHOLD RULE. [17] This information allows the user to form a GENERAL TRACKER to isolate additional "small" subsets of records. [10]

The values of other sensitive parameters may also be inferred through the user interface. These include legitimate system user names, and verifications of the existence of a data object.

Therefore, inference control at the user interface, should attempt to conceal the QUERY FAILURE POINT. [17] As the query is being processed, it may fail for any of the following reasons:

1. Invalid user name at log-in
2. Invalid authentication parameters submitted at log-in
3. Syntax error in the query language statement
4. Invalid data object name referenced
5. Invalid access function referenced
6. Access privilege request has not been granted to this user
7. Invalid authentication parameters submitted for this access request
8. Flow policy violation
9. Statistical aggregate requested has not been pre-computed in a partitioned database
10. Statistical inference control violation detected

To reduce system processing costs, a user should know that an on-line terminal session was unsuccessful because of an improper log-in protocol sequence. (However, no failure reason need be given for a batched query submitted by an operator.) Similarly, on-line documentation should be provided by the database management system vendor, to assist in the formation of syntactically correct query statements.

An interactive user terminal interface provides four variables which allow "expert" users to form inferences:

1. On-line turnaround time [23]
2. Accounting statistics [16]
3. The error message received [27]
4. The failure point of an interactive protocol sequence [17]

The PERTURBATION and SUPPRESSION of these variables may be controlled through the front-end of a database management system.

The user may gain valuable information from the ON-LINE TURN-AROUND TIME. Morris and Thompson [23] observed this phenomenon when the log-in protocol

< user_name, literal_password >

was submitted to an early version of the UNIX operating system. The turn-around time was much faster for an invalid user name than for a valid user name and an invalid literal password.

To delay (or PERTURB) the on-line turn-around time, the system may execute a procedure of the form:

```
delay := $10^6$ * (expected_turnaround_time - f(query_failure_point));
j := 0;
FOR i := 1 TO delay DO
  j := j + 1/i;
```

where f is a monotonically increasing function of the FAILURE_REASON_SEQUENCE_NUMBER, and need not be linear.

ACCOUNTING DATA may be used to read analogs of privileged information which are intentionally caused by COVERT FLOW. [16] However, several other types of inferences may be obtained from performance statistics:

1. Query failure point
2. Magnitude of a data object
3. The size of a set of records in a secondary storage data structure

As a counter-measure, end users who do not pay for computing resources, should either receive perturbed accounting statistics, or should not receive any such parameters.

A single error message "REQUEST DENIED" should be issued for all query failure reasons. Otherwise, a user may determine if the failure was caused by a non-existent data object, a non-existent access privilege grant, or a statistical inference violation[2]. [27]

Therefore, the following interactive sequence should be completed for all users for all queries. [17] Appropriate time delays should be induced between system prompts. This may be construed as suppression of the query failure point.

System: User name? [*Log-in only*]
User: My_name. [*Log-in only*]
System: Literal password? [*Log-in only*]
User: My_password. [*Log-in only*]
System: Response $\in$ {*"WELCOME", "REQUEST DENIED"*} [*Log-in only*]

System: Query text?
User: My_query.
System: Transform $<a_1, \ldots, a_n>$
User: $f_{user}(a_1, \ldots, a_n,$ *current_time*)
System: Query_answer $\in$ {*Information_Requested, "REQUEST DENIED"*}

The cost of inference controls at the user interface is justified by the PRINCIPLE OF INVISIBILITY: "Inaccessible data objects should be indistinguishable from non-existent data objects." [12]

Therefore, all four interface variables should behave in a time-independent and query-independent manner. The performance of the user-interface should be monitored and perturbed by a security subsystem at this level. Vendors of database management systems should produce products which provide protection at this level. [17],[18]

5. THE CONFINEMENT PROBLEM

COVERT FLOW is a generic term for information which flows from a privileged state through channels which are "not intended for information transfer". [16] The transmission may be of actual data values or of analogs. The processes of converting data to analogs and analogs to data, have been referred to as INFERENCES. [7] This section will briefly suggest ways in which improved software engineering practices may reduce the threat of covert flow in database management systems.

Techniques for covert flow by analog include reading the program status word, printing a variable number of carriage returns, and varying the system load as a boolean sequence over time. A Trojan Horse procedure may substitute analogs of protected values in place of accounting statistics. [16] These transmission channels should be regarded as data objects. Access to them should be controlled by system security policy.

The accepted counter-measures to covert flow, include program certification at compile-time, auditing, suppression of accounting data, increasing the band-width of the data paths, and periodic validation of the object module image in main memory.

In a database management system, user programs may attempt to generate analogs by unauthorized access to files or to functional units of the DBMS. To reduce the opportunities for this type of intruder, the following design principles should be implemented: [17]
1. Require that a main driver procedure access all DBMS functional units at a maximum distance of one subroutine call. This decreases the probability that implicit EXECUTE

access grants will cause procedures to execute in an inappropriate sequence[3].
2. Require that, when possible, absolute branch addresses should be substituted for subroutine call and return instructions. This measure also reduces the threat of implicit EXECUTE access grants to untrustworthy procedures.
3. Database functional units should be required to SCRUB (or set to zero) all registers, variable storage blocks in main memory, and temporary disk files, when control is relinquished. This sanitization technique prevents covert flow through STORAGE CHANNELS. [13], [16]

The ultimate solution to the confinement problem, is a dedicated database machine. [21] However, the secure database kernel is a reasonable alternative for a multi-user system. [18]

6. CONCLUSIONS

An inference is the process of transforming unclassified data values into confidential data values. Most previous research in inference control has studied the use of statistical aggregates to deduce individual records.

However, several other types of inference are also possible. Unknown functional dependencies may be apparent to users who have "expert" knowledge about the characteristics of a population. Some correlations between attributes may be concluded from "commonly-known" facts about the world. Database security managers have ignored these "obvious" relationships. Similarly, within a single record structure, different populations may have extremely different characteristics. To counter this threat, security managers should use random sampling of databases of similar populations, as well as expert systems. [17]

"Expert" users of the DATABASE SYSTEM may form inferences from the variable performance of the user interface. Users may observe on-line turn-around time, accounting statistics, the error message received, and the point at which an interactive protocol aborts. One may obtain information such as the frequency of attribute values, and a negative verification of the existence of a data object.

At the back-end of a database system, covert flow of inferences may be induced by Trojan Horse procedures and other user-written procedures. As a counter-measure, improved software engineering practices will reduce opportunities to bypass functional units of the database system.

Most current implementations of security subsystems in database management systems, are weak and inflexible. [20] The original authorization control mechanisms have frequently been down-graded or eliminated to improve time and space complexity. [14], [15], [20]

The term "DATA OBJECT" has been narrowly defined to include only real data records in a database management system. [5]

The security of the DATABASE and the security of the DATABASE MANAGEMENT SYSTEM must be recognized as separate but related problems. [17]

Thus by broadening definitions and re-evaluating the cost of protection against the cost of compromise, database designers may reduce the threats caused by lower level inferences. [17]

ACKNOWLEDGEMENTS

The authors wish to thank Zbigniew Michalewicz for a draft of his paper [22] which provided the original motivation for studying lower level inference control. Arie Shoshani was helpful in suggesting improvements for the final revision. Virginia Sventek provided assistance with graphics in the preparation of the final revision.

FOOTNOTES

1. However, individual records are usually not physically present in statistical database systems such as SUBJECT. Aggregates have been precomputed, and the individual record tapes have been destroyed. [10] Thus, algorithms for inference by performing relational algebraic operations on individual records (such as proposed by Denning, Denning, and Schwartz [9]) are not possible. Furthermore, most users do not possess the extensive preknowledge of individuals in a population. Thus, it is difficult to form inferences about arbitrary small subsets of individual records. [8]

2. However, if a query fails because of a syntax error or a system crash, then the user should know the exact reason.

3. This principle reduces the flexibility of C.J.Date's [5] model of three layers of design independence in a commercial database system. Logically, Date implies that EXTERNAL VIEWS call the CONCEPTUAL LEVEL as a subroutine. Similarly, the CONCEPTUAL LEVEL calls the PHYSICAL LEVEL. The implementation of a driver at one level, requires a programming group which reports directly to the project manager. This unit of a database design project would serve as an interface for communication between the EXTERNAL, CONCEPTUAL, and PHYSICAL programming teams. Brooks [1] notes that the communications paths between functional units in a system, will resemble the communication paths between individuals in a programming project.

BIBLIOGRAPHY

[1] Brooks, F., The Mythical Man-Month (Second Edition) (Addison-Wesley, Redding, Massachusetts, 1982).

[2] Chan, P. and Shoshani, A., SUBJECT: A Directory Driven System for Large Statistical Databases, in: Wong, H.K.T. (ed.) An LBL Perspective on Statistical Database Management (Lawrence Berkeley Laboratory, University of California, Berkeley, California, 1982).

[3] Chin, F.Y. and Ozsoyoglu, G., Statistical Database Design, ACM Trans. on Database Systems 6(1) (March 1981) 113-139.

[4] Codd, E.F., Further Normalization of the Database Relational Model, in: Courant Computer Science Symposia Series 6, Database Systems (Prentice-Hall, Englewood Cliffs, New Jersey, 1972).

[5] Date, C.J., An Introduction to Database Systems (Second and Third Editions) (Addison-Wesley, Redding, Massachusetts, 1977 and 1981).

[6] Denning, D.E., A Lattice Model for Secure Information Flow, Communications of the ACM, 19(5) (May 1976).

[7] Denning, D.E., Cryptography and Data Security, (Addison-Wesley, Redding, Massachusetts, 1982).

[8] Denning, D.E., A Security Model for the Statistical Database Problem, in: Hammond, R. and McCarthy, J.T. (eds.), Proceedings of the Second International Workshop on Statistical Database Management (Lawrence Berkeley Laboratory, University of California, Berkeley, California, 1983).

[9] Denning, D.E., Denning, P.J. and Schwartz, M.D., The Tracker: A Threat to Statistical Database Security, ACM Trans. on Database Systems 4(1) (March 1979).

[10] Denning, D.E. and Schlorer, J., A Fast Procedure for Finding a Tracker in a Statistical Database, ACM Trans. on Database Systems, 5(1) (March 1980).

[11] Earnest, L., Private communication to Hoffman, L., in: Hoffman, L., Computers and Privacy: A Survey, ACM Computing Surveys, 1(2), (June 1969) 92.

[12] Fernandez, E., Summers, R.C. and Woods, C., Database Security and Integrity, (Addison-Wesley, Redding, Massachusetts, 1981).

[13] Gold, B., Linde, R., Schaeffer, M. and Scheid, J., Final Report - Periods Processing vs KVM/370, Technical Report, System Development Corporation (May 1977), referenced in: Jones, A.K., Protection Mechanisms and the Enforcement of Security Policy, in: Bayer, R., Graham, R.M. and Seegmuller, G., Operating Systems: An Advanced Course, (Springer-Verlag, New York, 1979).

[14] Graham, G.S. and Denning, P.J., Protection: Principles and Practices, in: AFIPS Conference Proceedings, Spring Joint Computer Conference (AFIPS Press, Arlington, Virginia, 1972).

[15] Lampson, B.W., Protection, in: Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems (Department of Computer Science, Princeton University, Princeton, New Jersey, 1971). Reprinted in ACM Operating System Review, 8(1), (January 1974).

[16] Lampson, B.W., A Note on the Confinement Problem, Communications of the ACM, 16(10) (October 1973) 613-615.

[17] Lipton, D.L. and Wong, H.K.T., Lower Level Inferences: Security Threats from Objects Logically Unknown to a Statistical Database, Unpublished Technical Report in Progress, Lawrence Berkeley Laboratory, University of California, Berkeley, California (1984).

[18] Lipton, D.L. and Wong, H.K.T., A Secure Kernel for a Database System Using Flow Control for Grant Integrity, Unpublished Technical Report in Progress, Lawrence Berkeley Laboratory, University of California; Berkeley, California (1984).

[19] Lipton, D.L. and Wong, H.K.T., Authentication by Keyless Hash Transformation, Paper submitted to the 1984 Conference of the International Association for Cryptological Research (CRYPTO'84) for consideration of presentation, Lawrence Berkeley Laboratory, University of California, Berkeley, California (May 1984).

[20] Lipton, D.L. and Wong, H.K.T., Security of Database Management Systems: A Feature Analysis of Several Commercially Available Products, Unpublished Technical Report in Progress, Lawrence Berkeley Laboratory, University of California, Berkeley, California (1984).

[21] Lipton, D.L. and Wong, H.K.T., A Database Perspective on Computer Security, Unpublished Technical Report in Progress, Lawrence Berkeley Laboratory, University of California, Berkeley, California (1984).

[22] Michalewicz, Z., Functional Dependencies in Statistical Databases, Technical Report, Department of Computer Science, Victoria University of Wellington, New Zealand (1983).

[23] Morris, R. and Thompson, K., Password Security: A Case History, Technical Report, Bell Laboratories, Murray Hill, New Jersey (April 1978).

[24] Olsson, L., Protection of Output and Stored Data in Statistical Databases, in: ADB-Information 4 (Statistica Centralbyran, Stockholm, Sweden, 1975), referenced in: Denning, D.E., Cryptography and Data Security, (Addison-Wesley, Redding, Massachusetts, 1982).

[25] Saltzer, J.M. and Schroeder, M.D., The Protection of Information in Computer Systems, Proceedings of the IEEE, 63(9) (September 1975) 1278-1308.

[26] Schlorer, J., Identification and Retrieval of Personal Records from a Statistical Data Bank, Methods of Information in Medicine, 14(1) (1975).

[27] United States Department of Labor, Employment and Training Administration; Computing Division, Lawrence Berkeley Laboratory, University of California; and National Technical Information Service, U.S. Department of Commerce, Report 3: Social Indicators for Planning and Evaluation, 1980 Census of Population, Technical Report LBL-15850, Lawrence Berkeley Laboratory, University of California, Berkeley, California (April 1982).

[28] Wiederhold, G., Database Design (McGraw-Hill, New York, 1977).

[29] Woodfill, J., Segal, P., Ranstrom, J., Meyer, M. and Allman, A., INGRES Version 7 Reference Manual, Memorandum No. UCB/ERL M81/61, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, California (August 27, 1981).

[30] Yu, C.T. and Chin, F.Y., A Study on the Protection of Statistical Databases, in: Proceedings of the ACM SIGMOD International Conference on the Management of Data (Association for Computing Machinery, New York, 1977).

[31] Chin, F.Y. and Ozsoyoglu, G., Auditing and Inference Control in Statistical Databases, Technical Report, University of California, San Diego, California (December 1980), referenced in: Denning, D.E., Cryptography and Data Security (Addison-Wesley, Redding, Massachusetts, 1982).

[32] Dalenius, T., Toward a Methodology for Statistical Disclosure Control, Statistisk Tidskrift, 15 (1977) 429-444, referenced in Denning, D.E., Cryptography and Data Security (Addison-Wesley, Redding, Massachusetts, 1982).

[33] Hoffman, L.J. and Miller, W.F., Getting a Personal Dossier from a Statistical Data Bank, Datamation 16(5) (May 1970) 74-75.

[34] Cox, L.H., Suppression Methodology and Statistical Disclosure Control, Journal of the American Statistical Association, 75(330) (June 1980) 377-385.