SECURITY OF DISTRIBUTED DATA SYSTEMS

by

STEVEN D. FINCH

BGS, University of Nebraska Omaha, 1976

A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

Approved by:

Major Professor

Table of Contents

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Background

One of the most significant aspects of the future of information processing systems is the increasingly complex problem of securing and controlling information in a distributed database system. With the growing proliferation of comparatively easy-to-use microcomputers, there has been a dramatic increase in the conditions that nuture undercontrol and heighten exposure to security risks. It is common knowledge that corporate losses resulting from the accidental and intentional misuse of computers are growing. Current estimates of losses from computer fraud in financial institutions alone range from $500 million to $1 billion. Such losses should shoot over $7 billion by 1986 [LEVE 82].

Of greater concern is the growing potential for single instances of massive losses as corporate resources become accessible to a rapidly growing number of dispersed office computer users. According to a recent report, the average computer larcenist nets $450,000 to $620,000 per incident, while the average bank robber nets around $3,200 [LEVE 82].

There is obviously a pressing need to implement more stringent corporate-wide computer security measures.

1.2 Report Overview

As databases become larger and proliferate throughout industry and government, we will see a steady growth in the amount of sensitive

information stored in a computer system. If we expect to allow sharing of that database, we must provide a secure system to prevent misuse, loss or manipulation. Unless a user is assured that his sensitive data is protected from unauthorized access, he will not allow it to be stored in the system. Furthermore, society will not tolerate violation of a citizen's privacy because of non-secured databases.

This report analyzes distributed database systems and discusses possible vulnerabilities in the areas of access control, inference access control, integrity of data and communications security. Emphasis is placed on those controls that can be implemented in software. Hardware, physical, environmental and procedural controls are mentioned when they are used to supplement software controls.

CHAPTER II

ACCESS CONTROL

2.1 Introduction

About 60 percent of the reported abuses to computer systems are thefts or embezzlements by a trusted employee who misuses his access to the computer. Technology has progressed to the point of providing reasonable protection from the outside penetrator. This chapter will therefore address the aspects of access control of data to authorized system users. In other words, if user "A" wishes to obtain a data element owned by user "B", permission must have been granted by user "B", even if the data is resident in the same database. Access controls must regulate the reading, changing and deletion of data and programs. These controls must also prevent the accidental or malicious disclosure, modification or destruction of data by errant or deceptive programs.

2.2 Password Access Systems

The most common method used historically in control of access to a computer system and its resources is by using passwords. There are three basic password schemes for user identity--by something he knows or memorizes, by something he carries or by a personal physical characteristic. Prevalent schemes include single or fixed passwords, changeable passwords, random passwords, functional passwords (categorize a user according to his security classification) and extended handshake [HOFF 73]. The method by which the operating system handles the comparison of the user-supplied password with the authorized password on

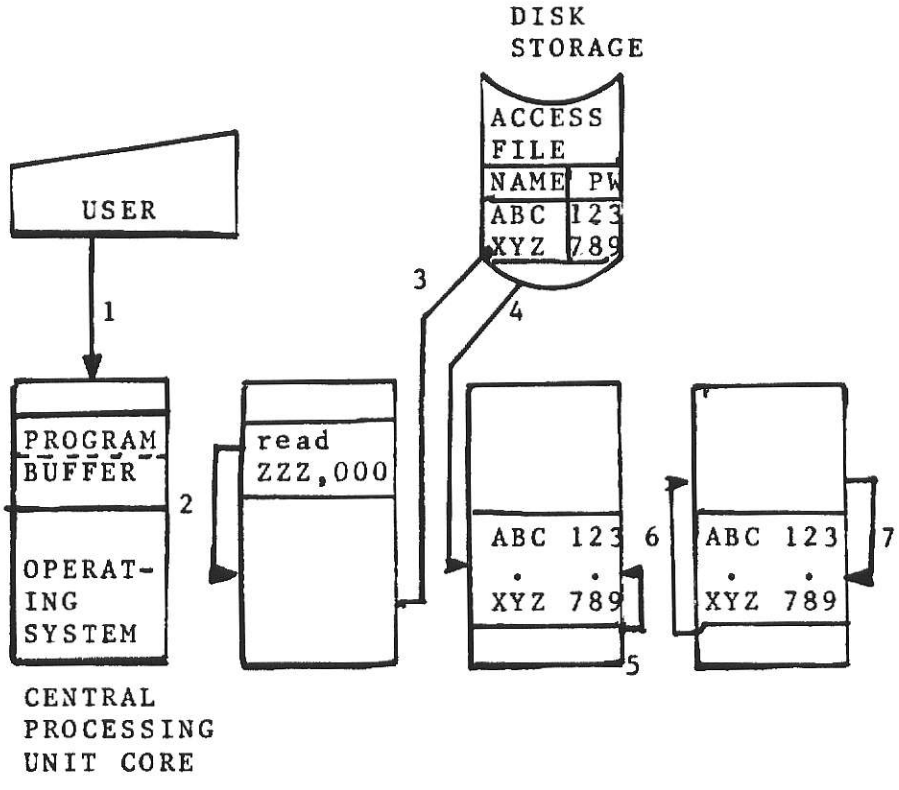
the authorization file is critical. Many systems write the authorization file into the user-allocated memory and then search this list for a matching password to a particular data set (data file). However, because the information remaining on the shared area (the user area) has not been overwritten, the current user now has access to the other users' passwords. This problem has been traditionally called implied sharing (Figure 2.1). Another type of implied sharing exists when registers that are accessible to user programs are used for the password comparison. Systems using this method do not expose the entire file to compromise; but if the registers are not overwritten or cleared, subsequent processes could read sensitive passwords. Many contemporary operating systems have resolved the implied sharing problem by performing the password search and comparison in memory accessible only by the system supervisor and/or by refreshing the memory used, by overwriting it with a set pattern of characters (ones or zeros or alternate ones and zeros).

There are many effective user identification means now commercially available to replace passwords in sensitive systems. These identification systems using handprint/fingerprint and voiceprint provide a higher level of security but at a cost prohibitive for most systems. Passwords are, therefore, by no means an outdated method of access control in many systems. They will provide the system reasonable security if properly handled externally by systems users and internally by the hardware and software.

In database management systems, it is sometimes necessary to protect selected data elements. Therefore, if we were to recount the passwords used to access that data element, we would need four passwords

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.



1. An unprivileged user requests a given amount of core. 2. The user requests to read file ZZZ and enters an arbitrary password 000. This file does not exist but would be the last file in the sort sequence. 3. The operating system must check the access file to see if he user has access to file ZZZ. 4. The system copies the access file into the users buffer area. 5. The operating systen searches for file ZZZ. 6. Failing to locate file ZZZ the system notifies the user and control is returned to the user. 7. The user reads his buffer obtaining file names and passwords.

FIGURE 2.1 Implied Sharing of Privileged Data

(Figure 2.2). If we now expand this database management system to a distributed system, we could envision a doubling, tripling, or quadrupling of the number of passwords needed. This would defeat the availability goals of our DDBMS and probably encourage the systems users to devise ways to thwart the security system.

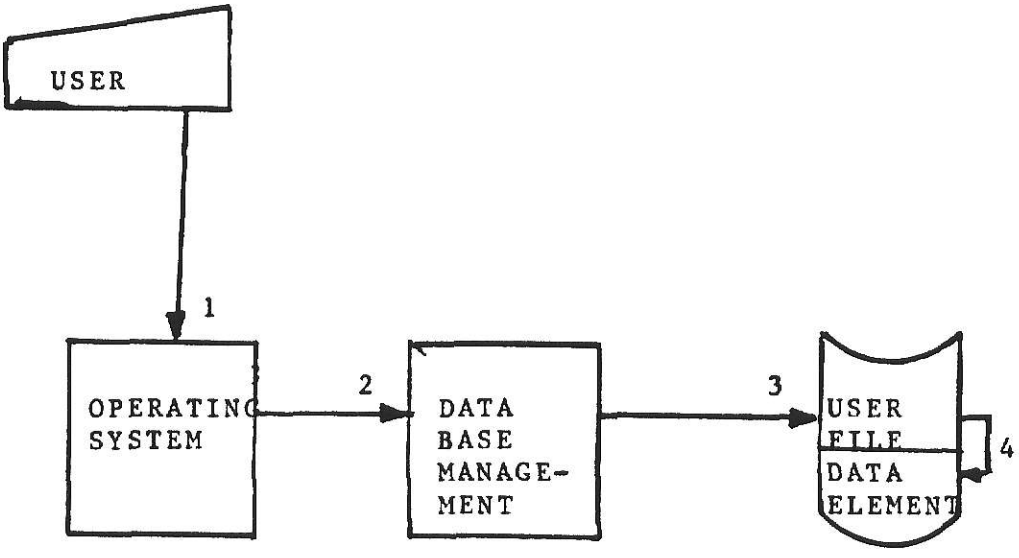
This gross proliferation of passwords would also greatly increase the probability that external disclosure would occur. Computer systems auditors and security evaluators have determined that one of the greatest vulnerabilities to penetration by an outsider is the availability of passwords that are not properly secured [ABBO 79]. Some of the common misuses of passwords include:

1. Passwords taped to the terminal
2. Passwords left in an unattended desk drawer
3. Passwords written on a paper in the purse or billfold
4. Passwords taped under the desk drawer
5. Passwords written inside the terminal users guide

These noted problems refer to systems where the user supplied only the password to enter the system. If we expand that by only three passwords, it is readily apparent that prevention of disclosure is difficult. If the passwords are randomly generated (i.e., PJ54C6U), one can see the difficulty in remembering three or more and applying them to the correct situation. The obvious solution is to develop an alternate scheme for user recognition and granting of access.

2.3 Transaction Processing Controls

The commands issued by the user of a transaction processing system are calls on a small library of transaction processes that perform specific operations, such as querying and updating on a database. In



1. At the system entry access point.
2. At the DBMS access point.
3. At the user file request.
4. At the data element level.

FIGURE 2.2 DBMS Access Control Points

such a system, the only authorized processes are the certified transactions. Therefore, it is possible to enforce the rules of access at the interface between man and machine. A user can identify a set of records by a characteristic formula 'C' which is a logical expression using the relational operators (=, >, <, etc.) and the boolean operators (AND, OR, NOT). These operators join terms which are indicators of values or compositions of relations. An example is:

C = "Army Officer and Graduate Student or (Status = Full time).

The transaction control program looks up a formula R specifying restrictions that apply to that given user. It then proceeds as if the user had actually presented the formula and R [DENN 79]. The concept of adding user restrictions, or privileges if you will, to the user request is common in database management systems.

When the system allows owners of records to revoke privileges that may have been passed around among users, it must be designed to revoke also any privilege that emanated from the revoked privilege. It is not difficult then to imagine the amount of communication that may be required in a distributed system.

In a distributed database system, a very critical but basic decision must be made concerning where the evaluation for access is made. If the evaluation is made at the node where the user entered the system, the transaction plus its appended authorization may be transmitted on the communications system. (For example, user A requests the number of items shipped to location ABC. This data is stored at a non-local node; the transaction might look like this: 'list,no, where item = widget and loc = ABC'). The local system would send this transaction to the node containing the data with the user I.D. and

authorization appended (i.e., 'A,OK'). So the entire transaction might appear as 'list,no, where item = widget and loc = ABC: A,OK;'. This would allow the perpetrator with a piggyback terminal (Figure 2.3) to monitor the authorization codes. This is potentially a very dangerous situation, but could be resolved by verifying access at the local node and then transmitting the transaction with an appended intermediate code that could be changed dynamically. This transaction would then be reevaluated at the node containing the data elements to be acted upon. The most casual reader will soon recognize the communications and synchronization required to maintain this type of access system. An alternate method would be to strip off the transactions that could be performed at the local node and grant authorization as appropriate. The remaining transactions would be communicated to the nodes containing the requested data for authorization. While this seems the simpler way to handle the problem, the piggyback terminal operator will soon realize that the transaction with an appended authorized I.D. will gain access to selected data elements. Then penetration is a simple matter of imitative deception. This method of handling access control could also make enforcement of inference access more difficult. This problem will be discussed further in Chapter III.

Unfortunately, transaction processing controls are traditionally implemented on a general purpose computer and are vulnerable to manipulation via the operating system. Most security flaws in existing systems are the consequences of design shortcuts taken to increase the efficiency of the operating system.

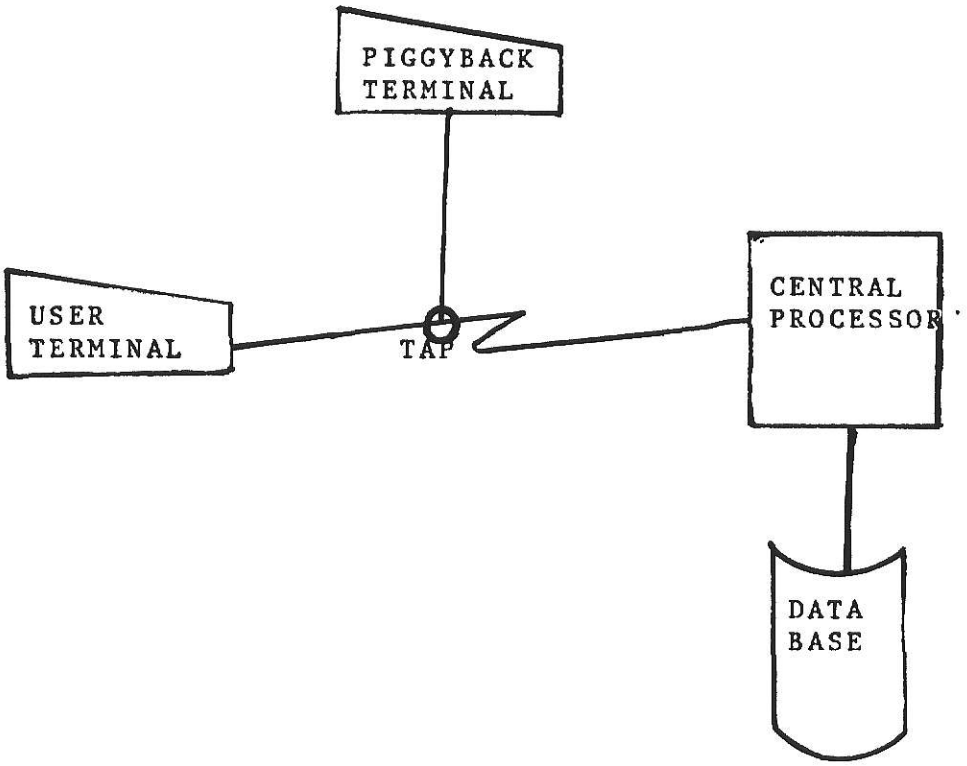


FIGURE 2.3 The Piggyback Terminal

2.4 SYSTEM R*

SYSTEM R* is the distributed version of IBM's SYSTEM R. Access control is via a modification of the transaction processing controls discussed in the previous section. Authorization rights must be refined so that different users have different access rights on the same database. IBM's designers quickly dismissed the use of passwords to accomplish this.

The mechanism used goes beyond the traditional password approaches. Its highlights are enumerated here:

1. Passwords are not used to control access rights to specific objects.

2. The system figures out when processing a user's access request whether the user is authorized to access the specified object.

3. Users access rights to specific objects can be dynamically modified.

4. To each object defined in the database is associated the identification of users who have specific access rights on this object.

5. Users can be allowed to grant and revoke access authorization on objects they are authorized to access.

6. The mechanism of access rights allocation is not based upon a rigid hierarchical method of access rights grant, rather control is distributed among authorized users.

The advantages of this mechanism include its flexibility, a high level of privacy, and ease of use. Another attractive aspect is the complete distribution of control over access rights. There does not need to be a centralized database administrator holding all power and granting access rights to specific users.

The authorization mechanism basically performs three kinds of operations:

1. Checking the ability of a SUBJECT to access an OBJECT
2. Granting access rights on an OBJECT to a SUBJECT
3. Revoking access rights on an OBJECT from a SUBJECT

One of the basic OBJECTS in SYSTEM R* are relations which are sets of n-tuples; a relation is conceptually represented in the system as a table. There is a special relation kept by the system, called the authorization table, which records the state of user's access rights of the relations. For each relation currently existing in the database, there exists at least one entry in the authorization table.

Privileges which may be exercised by each subject fall into three basic categories:

1. Privileges on relations: These are the normal database query authorizations (i.e., SELECT, INSERT, DELETE, UPDATE, EXPAND, INDEX).

2. Privileges on a program: These are defined as privileges granted to a program at compile time (i.e, RUN).

3. Special privileges: This encompasses special authorizations called resource authority and database administration (DBA) authority.

    a. Resource authority: This is required to create tables and acquire segments (logical storage space) because these operations use up space in the database.

    b. DBA authority: It is the highest level of authorization and gives its owner the capability to use any privilege on any relation and to run programs; DBA authority also includes resource authority. The only power restriction imposed to DBA consists in the interdiction for a DBA to grant or revoke privileges he uniquely holds due to DBA authority.

Privileges on relations and programs may be owned with or without grant option; a privilege obtained with grant option allows its holder to transmit this privilege to other subjects. Transmission of a privilege is performed by a GRANT statement. The GRANT statement has the form: GRANT <privileges> ON <object_name> to <subject_list> (WITH GRANT OPTION). The grant statement runs successfully if and only if the

grantor holds all the privileges he wants to deliver, with grant option. However, he may hold them either personally or he may belong to a set of users holding these privileges with the grant option. If the privilege is transmitted to the grantee without the grant option, the grantee is only allowed to use the privilege; he may not grant it to others. Any subject who has granted a privilege may later withdraw it by issuing a revoke command. The privileges held on the object will then be withdrawn unless the revokee had also acquired the privilege from another grantor. Conversely, a user is not allowed to revoke a privilege from a subject to whom he never granted it. The format of the REVOKE statement is:

REVOKE <privilege> ON <object_name> FROM <subject_list>.

SYSTEM R* provides the users with the capability for a more secure database because of the ease in granting and revoking privileges to objects. All objects in SYSTEM R* are protected when created and initially only the creator has access rights. Like other transaction processing systems, SYSTEM R* is subject to the vulnerabilities of the communications system and host operating system.

2.5 The Database Machine

The use of a backend processor or database machine properly implemented may greatly increase database security. There are a number of advantages resulting from the physical separation of the DBMS and the rest of the system. When the DBMS and the rest of the system are separate entities, it is possible to specialize each for their given task. The DBMS need not have a general purpose operating system; one

which is tailored to serve just the database management function will suffice. The host computer's general purpose operating system at each node need not support database management functions; it must only provide a mechanism to channel data to and from the DBMS. Thus, both components and the interface between them are simpler than would otherwise be necessary and so the difficulty of their construction is reduced somewhat.

The physical separation of the DBMS from the rest of the system should enhance database security. Since the DBMS and the rest of the system are connected by a single, well-structured link, assaults on the database via the host computer's operating system are virtually eliminated.

2.6 LADDER

LADDER (Language Access to Distributed Data with Error Recovery) is a distributed database management system developed at SRI for the Advanced Research Project Agency of the Department of Defense. The goal of the system is to provide the user easy access to information stored on multiple computers, under various database management systems. A graphical overview of LADDER is presented at Figure 2.4.

The system employs a natural language front-end called INLAND (Interactive Natural Language Access to Navy Data). While the primary function of this component is to provide the user with the ability to ask questions about the database in a natural language (i.e., English with an expansion to include Navy terminology), it also provides additional security within the system by isolating the user from

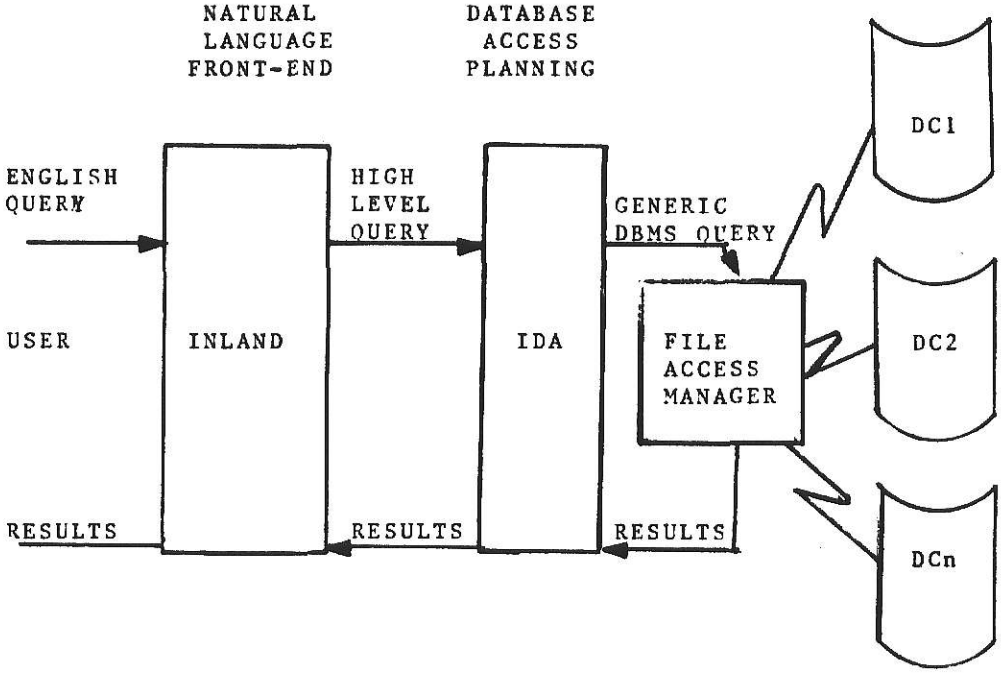


FIGURE 2.4 The LADDER System

database information such as location, and file and record structures. This is accomplished by the simple conversion of the user's query into a high-level query to the next component of the system called IDA.

IDA (Intelligent Data Access) translates the query received into a query program to be issued to the database management systems. To accomplish this, IDA must have access to a data dictionary or a description of the database structure. One query to IDA may generate several subqueries that may require access to multiple databases. IDA is not given the precise location of each file. In fact, IDA is led to believe that all the files exist on one computer. It, therefore, uses generic filenames and the next component must select the appropriate computer.

The generic DBMS query is received by the FAM (File Access Manager). This component is responsible for the connection of various computers finding the most recent versions of pertinent files, issuing the actual DBMS's queries and recovering from errors. Because of the structure of FAM, a system could be implemented by-passing both INLAND and IDA. While this would make the system cost much less, the two levels of transparency provided would also be lost and security lessened.

2.7 Conclusions

All methods of access control in a distributed database system must necessarily result in the transmission of the authorization in some form over the communications system. This transmission is subject to monitoring by unauthorized persons, and systems penetration could

result. There are ways to protect that transmission through encryption of the transaction or communication link. The methods of encryption will not be discussed in this paper. Therefore, the interested reader should consult one of the following references for more information [DOOL 78, McGL 78, ROTH 77].

Passwords are an effective means of controlling access at the man-machine interface point, but beyond that point they are less effective due to the inadequacies of current operating systems and the probability of compromise due to human nature.

Transaction processing controls or variations of this basic concept provide the best possibilities for reasonable security. Access controls can be implemented in software, hardware or firmware and be controlled so as to not allow manipulation by unauthorized users. Additionally, a greater level of transparency is available with transaction processing type controls.

Backend processors or database machines provide additional security because of the dedication of those systems to the task of controlling the database, and the isolation of the database from the general purpose operating system.

CHAPTER III

SECURITY IN STATISTICAL DATABASES

3.1 Introduction

The problem of enhancing the security of statistical databases has attracted much attention in recent years. This problem can be stated as follows. A statistical database is a collection of records that contain information on some number n of individuals. Each record contains confidential category and data fields; at least two values exist for each such field. The category fields are used to select and identify records, while the data fields contain the information. Given a statistical database containing information about a population of individuals, this database should provide users with statistical data such as totals, median, counts, etc. for groups of individuals without compromising the data on any individual. Compromise can occur by asking for the average of an element where there is only one response set qualifying or by asking two or more related queries resulting in a response set size of one when the queries are compared. The technique is to form queries so that responses will provide information on one individual without a direct query for that information. A compromise is positive if the questioner deduces the value of a given category or data field of an individual and negative if the questioner determines that the individual is not in the selected category or data fields. This problem is complicated by the possibility that the user could ask a large number of questions of the database by submitting a series of related queries. To compound this problem further, if the user has access to some of the data via public or other means (i.e., sex, age,

marital status, etc.) or confidential (salary, etc.) information about certain individuals, he can use this knowledge to frame queries to obtain confidential information on others.

This chapter will give an overview of both problems associated with and the prevention of those problems. It will attempt to present some additional issues that are incurred due to the distribution of a statistical database.

3.2 The Problems

A database is said to be compromisable if any protected element of data for an individual can be determined exactly by one or a series of queries. If the answers are distorted, this exact determination is not possible. However, a user can obtain statistical estimates of the data. The larger the number of queries, the more precise the estimates. Under this definition, disclosure cannot be prevented. It can, however, be controlled.

A few examples will serve to illustrate some of the difficulties. The database (Figure 3.1) is assumed to contain personal information such as age, employer, education, and salary and we are allowed to request totals, counts and averages.

Example 1. If we know Mr. Keys' age, education and employer, the following query could be used: List the average salary of all persons where age is 39, employer is the ABC company and education is a Masters in Computer Science at Kansas State. If Mr. Keys is the only person who meets all of the criteria of the query, the exact salary will be returned.

ABC CO. PERSONNEL FILE

Name	Age	Employer	Div	Education	Salary
KEY	39	ABC CO.	FIN	MS COMPSCI KSU	$34,000
JONES	39	ABC CO.	FIN	MS COMPSCI KSU	$36,000
SMITH	32	ABC CO.	ACCT	BS BUS IOWA ST	$22,000
CORD	34	ABC CO	FIN	BS BUS KU	$25,000
DODD	33	ABC CO	ACCT	MBA U-TEXAS	$31,000

Figure 3.1
Sample Database

Example 2. Many authors have suggested restricting the size of a response set (i.e., require that the response be composed of information on 2 or more persons). Then the following two queries would result in a compromise.

a. List the total salary for all persons where age is in the range 39-50 and employer is the ABC company and education is a Masters degree in Computer Science at Kansas State.

b. List the total salary for all persons where age is in the range 40-50 and employer is the ABC company and education is a Masters degree in Computer Science at Kansas State.

Now subtract the results of query b from query a and you have Mr. Keys' salary.

Example 3. If the query in Example 1 resulted in a response set of size two, anyone knowing the salary of one could mathematically obtain the other (i.e., if Mr. Jones wants to know the salary of his associate Mr. Keys, knowing they are the same age and have the same education, the query in Example 1 and some simple math will provide the correct amount of Mr. Keys' salary.)

Example 4. If we ask the following queries:

a. List the number of individuals where age is in the range 39-50 and education is Graduate Degree in Computer Science from Kansas State and salary is in the range of $30,000 to $35,000.

b. List the number of individuals where age is in the range 40-50 and education is Graduate Degree in Computer Science from Kansas State and salary is in the range of $30,000 to $35,000.

If the answers to these questions differ, we have determined that Mr. Keys makes between $30,000 and $35,000 per year. If more precision is desired, you could perform the queries again and reduce the salary range.

In all of these examples, we have been able to obtain salary information on Mr. Keys without using his name. There are, of course, many other possibilities to compromise the data through inference. One method is by use of a technique called trackers.

3.3 Trackers

Statistical databases can be easily compromised even if some queries are not answerable because their query sets are too small. The questioner divides his preknowledge of a given individual 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 individual has a characteristic previously unknown [SCHL 75].

Trackers were further defined by [DENN 79] into three types:

1. Individual
2. General
3. Double

3.3.1 Individual Trackers

This tracker uses the available information (C) on an individual (I) to formulate two queries such that additional information can be obtained. For example, if the questioner believes that C = 32, ABCCO characterizes Smith, but the restriction of set size of two prevents his asking directly, he could construct a tracker.

$$\begin{aligned} \text{COUNT (32, ABCCO)} &= \text{COUNT (TOTAL)} - \text{COUNT (>32, ABCCO)} \\ &= 5 - 4 \\ &= 1 \end{aligned}$$

This simple example implies a general knowledge of the database by the questioner. Repeated use of the tracker will give the questioner the knowledge needed to gain confidential information.

3.3.2 General Trackers

The individual tracker is based on the concept of using categories known to describe a certain individual to determine other information about that individual. A new individual tracker must be found for each person. The general tracker removes this restriction. It employs a single formula that works for the entire database. No prior knowledge about anyone in the database is required. Again using Figure 3.1, if the questioner wants to know the salary of all persons in the accounting department, the query

$$\text{SUM} = \text{SUM (ACCT; SALARY)} + \text{SUM (ACCT; SALARY)}$$

would result in response set size of two and would be rejected. However, the query

$$\begin{aligned} \text{SUM (ACCT SALARY)} &= \text{SUM (TOTAL SALARY)} - \text{SUM (FIN SALARY)} \\ &= \$148{,}000. - \$95{,}000. \\ &= \$\ 53{,}000. \end{aligned}$$

Because there is no need for a preknowledge of an individual in the database, the general tracker is a powerful technique to effect a compromise.

3.3.3 Double Trackers

The general tracker is not guaranteed to work when more than half of the range of the query set sizes are disallowed. However, this does not imply that the database is secure; it may still be susceptible to compromise by the methods of the double tracker. A double tracker uses a pair of characteristic formulas as opposed to the one described in the previous two methods. For example, to find the salary of Smith in Figure 3.1, we know his degree and department.

```
SUM (BS, ACCT; SALARY) = SUM (TOTAL; SALARY)              = $148,000.
                       - SUM (FIN; SALARY)                = $ 53,000.
                       + SUM (GRAD DEGREE; SALARY)        = $154,000.
                       - SUM (GRAD DEGREE, FIN; SALARY)   = $ 70,000.
```

It is then easy to determine that the graduate degree in the accounting department has a salary of $31,000. and then Mr. Smith must have a salary of $22,000.

```
TOTAL GRAD. DEGREE SALARY   $101,000.     $53,000.  TOTAL ACCT SALARY
FIN. GRAD DEGREE SALARY     - 70,000.     -31,000.  GRAD DEGREE
                            ----------    ----------
ACCT. GRAD DEGREE SALARY    $ 31,000      $22,000   Mr. SMITH
```

The single and double trackers define sufficient conditions under which a uniform procedure will calculate any unanswerable query. It is almost always possible, given a C, to find further information in a record even

when a general tracker does not exist. A double tracker is ruled out only when less than 1/3 of the possible query set size are observable. If the query set size is two or less, individual trackers may exist. However, such severe restrictions of the query set size would ruin the database as a useful source of statistical information without providing security for its records.

3.4 Solutions

Many researchers have considered methods for preventing such penetration of the database. They are:

1. Determine minimum response set
2. Partitioning of database
3. Statistically modifying the response or value dissociation (lying)
4. Detect trackers
5. Random sample
6. Limit the number of queries

3.4.1 Response Set

One simple method involves establishing a minimum size for the response set. However, this control has been shown defeatable by the tracker [DENN 80].

3.4.2 Partitioning

The second method is by partitioning the database. With this technique, records are stored in groups of some minimum size. Queries may refer to any group or set of groups, but never to subsets of records within the group. This prevents a user from isolating a record with overlapping queries. However, the formation of such groups may severely

obscure useful information in the database. In addition, the revision of groupings, as records are added and deleted, may be costly.

3.4.3 Statistically Modifying the Response or Value Dissociation

This approach involves the distortion of the values in the database, that is, storing a value (y + d) where y is the data and d is some random value drawn from a distribution. There are a number of problems with this scheme:

1. It is not applicable to non-numeric fields.

2. For numeric fields with skewed distributions (such as salary), the selection of distribution for d is very difficult.

3. Overlapping queries such as those in Example 2, Section 3.2 can be used to estimate certain values unless the variance of d is significant when compared to the largest value of y.

4. If the value of d is quite high in relation to y, the results of a query could be distorted beyond usability. A related technique is that of value dissociation or lying. Using this approach, the database responds to a query with an exact value from some record, but not necessarily the record requested by the query. The techniques used in Example 4, Section 3.2, may enable the perpetrator to determine which values are likely to be correct.

3.4.4 Detection of Trackers

Recent studies [DENN 80] have revealed that trackers are almost as easy to detect as construct. The number of queries required to find a tracker is at most $Q(\log_2 S)$ queries, where S is the number of distinct records possible. The results of an experiment on a statistical database show that the procedure presented in the above referenced study can find a tracker in one or two queries. The results also showed that the large proportion of the possible queries in most databases are general trackers and may be discovered quickly by guessing. There was, however, no limit established on the number of queries required to find a tracker. The ability to discover the use of trackers will not prevent the disclosure of confidential information. It may lead to discovery of the questioner and his isolation from the database by denying him system access.

3.4.5 Random Sample

A recently developed technique is to answer the query with a random sample response. With this method, a query is answered with respect to a random sample from the response set. While providing a low risk of actual compromise, answers to queries may be obtained with relative assurance of small inaccuracies.

3.4.6 Limit the Number of Queries

This technique would count the number of queries for a given user, allowing each user a set number of queries in a given period of time on a given record or set. The control of this technique would be very difficult and the result could well be denial of service to qualified

users during heavy usage times. This technique will not protect the information over a period of time sufficient for the questioner to formulate his queries and meet the number of queries constraint.

3.5 The Distributed Database Problem

In this section, the discussion will focus on the unique problems associated with distributing the database. It should be noted at the outset that the implications of inference access at any node are as discussed in the first four sections of this chapter, if the user has access to the node directly and not through the distributed system.

To facilitate this discussion, problems will be categorized by the method in which the database is organized. They are:

1. Partitioned
2. Replicated
3. Combination

3.5.1 Partitioned Database

A partitioned database is one where only one copy of data exists and related data is stored at one node in the distributed system. There are some security advantages in storing the data in this manner.

1. Access can be controlled at the node containing the data.
2. Patterns of queries may be more easily detected.
3. Trackers will be more easily detected because related data is stored together.
4. Trackers may be less effective as the number of the related records increase. The probability of unique identifiers will decrease.

Looking at the strict definition of a partitioned database, one could determine that the inference access problem is similar to a

centralized database. There is the possibility of increased control, if the database is partitioned in a manner that would inhibit access to sensitive data elements. This type of partitioning may cause the goal of data availability to be greatly impaired and the expense may outweigh the security gained.

3.5.2 Replicated Databases

A replicated database is one in which identical data elements are stored at more than one site. In this type of distribution, control of inference access is much more difficult. Access must be controlled central to the entire system, as each node will not be aware of queries at other nodes in the system. If the user is permitted to access his local node separate to the distributed system, no method of control will be effective. Application of control techniques, such as number of queries, will not be effective unless control is exercised centrally. Controls, such as response set size, will be hard to control without extensive communication between the central system controller and the node containing the information. It will be virtually impossible to protect information when the total number of records of the same type differs on two separate nodes. For example, using Figure 3.1, if all of the personnel records exist at node A and all but Dodds record exist at node B, the user with knowledge of this and the ability to address queries to separate nodes could ask for a total_salary at each node and via simple math determine Dodds' salary. The use of trackers in replicated data systems could be very dangerous, because as replicated versions become smaller and smaller to provide user availability, the probability of compromise would become larger.

3.5.3 Combination Databases

This may provide a middle of the road solution for database administrators. Some of the advantages of the partitioned method could be realized and data availability could be improved. If the data is replicated such that non-sensitive data are the only items replicated and sensitive items are partitioned, then the security provided by the partitioning is preserved to a degree. The trade-off here will be data availability vs. data security. When we begin to allow aggregates of the database, however, the effectiveness of trackers will increase.

3.6 Conclusions

Control of inference access to statistical databases is a very difficult problem. My brief study reveals that the distributed system will present new problems that are much more difficult than in a centralized system. If the goal of data availability is to be reached using the distributed system, then it appears as if security is on a collision course with that goal.

While this is not a new problem, it may require attention due to government regulation, such as the Privacy Act of 1974. Much of the data contained in a statistical database is covered in that act. It will be essential for the systems designer to develop a system that is transparent to the user, that is, transparent from the standpoint that the user must not know the database structure or location of data sets or records.

CHAPTER IV

DATA INTEGRITY

4.1 Introduction

By integrity we mean that the DBMS must perform as certified. Certification is defined as a compliance of a DBMS with a set or subset of designated specifications and/or standards to meet the requirements of a specific distributed DBMS. As an element of DDBMS security, integrity is essentially the guarantee that the system is functionally correct and complete. In this chapter, I will discuss three main aspects of integrity:

1. Semantic integrity
2. Concurrency control
3. Failure transparency

4.2 Transactions

The concept of transactions plays a crucial role in integrity. The idea of a transaction should be familiar to the reader. A transaction usually results in a set of actions on the database that form a meaningful unit of work, such as reserving a seat on an airplane.

A transaction as seen by the user may involve more than one transaction from the system viewpoint. In this chapter, the system will be the primary issue. A transaction should be designed and executed so that it either completes satisfactorily or makes no change to the database. The transaction manager makes use of a set of operations that can be performed on data objects. An instance of such an operation is called an action. A transaction is thus a sequence of actions.

A transaction can fail to complete for various reasons, such as:

1. An action violates a security or integrity constraint (i.e., the new balance in the credit union is less than the minimum required for membership).
2. The user cancels the transaction.
3. An unrecoverable I/O error occurs (i.e., due to media damage, such as headcrash).
4. The system backs out of the transaction to resolve deadlock.
5. The application program fails.
6. The system crashes.

The essential points in the life of a transaction are begin and commit (or end). Until the transaction commits, none of its changes is seen by other transactions. If it never reaches the commit point, the transaction should have no effect on the database. A transaction may also abort (terminate without committing), save (save the transaction status without committing), or backing (to a save point). If a failure occurs, the recovery procedure may have to undo or redo the transaction. Although some systems do not explicitly define transactions, the transaction concepts still apply.

4.3 Semantic Integrity

Very little work has been done on semantic integrity in the distributed environment. I will attempt to define the problems. This will hopefully cause the reader to consider possible solutions.

Semantic integrity is concerned with ensuring that the database is correct even though users or applications programs try to modify it incorrectly. Assuming that the database security systems prevent

unauthorized access and hence malicious attempts to corrupt data, most potential errors will be caused by incorrect input, incorrect programs, or lack of understanding on the part of the user. Traditionally, most integrity checking has been performed by applications programs and by periodic auditing of the database. Some of the problems of relying on applications programs for integrity checking are:

1. Checking may be incomplete because the applications programmer may not be aware of all the semantics of the database.

2. Each applications program must trust other programs that modify the database. One incorrect program could invalidate the entire database.

3. Code to enforce the same integrity constraints exists in a number of programs, wasting programming effort and risking inconsistencies.

4. The criteria for integrity are buried in procedures and are, therefore, hard to understand and control.

5. Updates, inserts or deletes performed by users of high-level query languages cannot be controlled.

6. In heterogeneous DDBMS environments, data set parameters may vary and applications controls could cause database errors.

Most of these errors could be detected by auditing. However, this is not a satisfactory means in enforcing integrity due to:

1. The difficulty in tracing the source of an error and then correcting it.

2. The time required to perform item 1, above, may result in the incorrect data being used in various ways, causing errors to propagate through the environment.

Current DBMSs provide little support for semantic integrity. However, some work has been done to define the need for such support. Integrity constraints should be expressed by persons with data control responsibility. A validation mechanism then checks changes to insure compliance with these constraints, and predefined procedures are performed when a potential violation of one or more of the constraints exist. It should be noted that integrity constraints can only insure that the data in the database is reasonable, not correct.

Integrity constraints can be defined in a number of different ways.

1. Record or Set - one classification concerns whether a constraint applies to an individual record (or tuple) or to a set of records. This is called granularity (i.e., for a record constraint, all salaries must be less than $75,000; for a set constraint, the average salary for professors must be less than $35,000).

2. Static or transitional - constraints can specify either correct states of the database (static constraints) or correct transitions. The previous examples were both static. A transitional constraint might be: a new salary must exceed an old salary.

3. Selective or general - constraints may be enforced only upon the occurrence of certain actions (i.e., deletion, update) or at any time.

4. Immediate or deferred - constraints that hold after any action are immediate, and those that hold only after the completion of the transaction are known as deferred.

5. Unconditional or conditional - constraints may be enforced only when a certain condition is true (i.e., the salary of all data entry operators must be less than $12,000). This constraint is enforced on only those employees' records where job is data entry operator.

Constraints may also be classified as range (values within specified bounds), format, and statistical (using statistical functions).

Validation of these constraints may be performed at different times (Figure 4.1) depending on the type and specifications. Violations when they are detected may generate a number of corrective responses.

1. If the error is detected during T_1, the action is rejected and a message returned indicating the action was in error.

2. If the error was detected in T_2, only the actions performed after the previous integrity check need be repeated. The error is then isolated to the actions performed after that check.

T_1:	after a primitive database action request.
T_2:	at specific points within a transaction.
T_3:	after completion of a transaction.
T_4:	on request from the DBA or auditor.

Figure 4.1
Possible Validation Times

3. If the error is detected in T_3, the whole transaction is backed out returning the database to its initial state. In this case, it may be impossible to determine which action is in error.

4. If the error is detected in T_4, it may not be possible to determine which action caused the error. Automatic correction is generally not possible. The best action is a warning message to the DBA with a snapshot of the current state of the database, plus a listing of the transaction file since the last verification [BAYE 76].

Enforcement of semantic integrity constraints in a distributed system where replicated data exists could be very difficult and clear cut solutions are not yet apparent. One method might be to apply transactions to a designated master database and then transmission of the updates to other nodes after all constraints have been met. This method would result in inconsistency for short periods during processing of the transactions. Database administrators would be faced with a decision of relative importance, that is, of integrity vs. consistency.

4.4 Concurrency Control

In distributed database systems, the concurrency control mechanism at one site cannot know, instantaneously, what is happening at other sites. A good concurrency mechanism attempts to ensure that one user does not see inconsistent data as a result of concurrent updating by other users. Another requirement of the concurrency facility is to

provide information for recovery. A system of concurrent transactions is said to be synchronized if it is equivalent to a system where all transactions are run serially-equivalent in the sense that the same final state is reached and the same outputs are produced. Such a system is said to be integrity-preserving. Three forms of inconsistency can result from concurrency: lost updates, dirty reads and unrepeatable reads (Figure 4.2). From this you can define three levels of consistency depending on the type of consistency guaranteed. They are:

Level 1 prevents lost updates;
Level 2 prevents reading dirty data;
Level 3 prevents unrepeatable reads.

The main argument for allowing lower levels of consistency is to increase concurrency, thereby increasing throughput. The approach most commonly used to eliminate consistency problems is locking. Many locking schemes have been proposed, the most popular being presented by Menasce and Muntz [MENA 79]. (Due to the extensive discussion of locking and deadlock prevention algorithms elsewhere, they will not be discussed here.)

Locking may be either centralized or distributed. A centralized lock controller maintains lock tables and detects and resolves deadlocks in a way similar to a centralized site. With distributed locking, however, deadlock may not be detected locally by any site. To resolve this, one solution would be to designate one site as the deadlock detector. Other sites in the distributed system periodically send it lists of locks that have recently been requested, granted, or released. Deadlock detection then operates as in a centralized system.

There are some critical decisions to be made concerning the use of locks. The database administrator must evaluate the system to determine

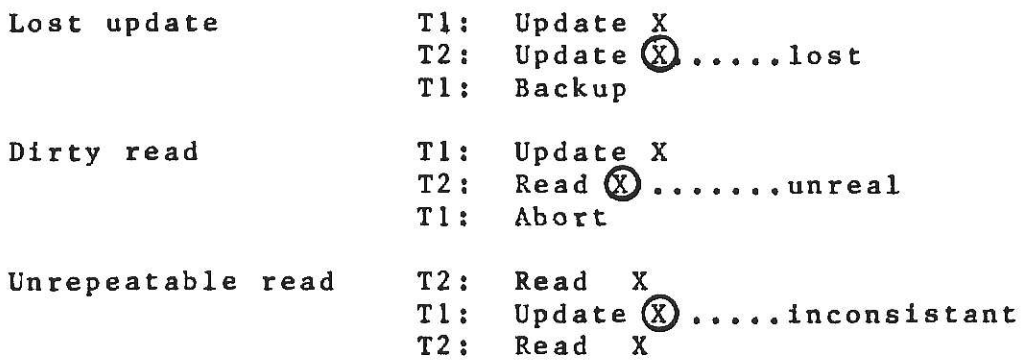


Figure 4.2 Types of inconsistancy due to concurrency

the operational balance between consistency vs. denial of access due to locks.

4.5 Failure Transparency

We have seen that a transaction appears as an atomic action to the user, that is, it either completes successfully or it has no effect at all (Section 4.3). This means that if there is a failure and the transaction has not been committed, the effects of the transaction must be backed out. Care must be taken in a distributed system to coordinate all nodes during a commit that involves multiple sites; otherwise, some nodes might back out while others commit successfully. A two phase commit protocol (Figure 4.3) is sufficient to prevent this type of inconsistency. (Note: This author could not find an implementation where this protocol has demonstrated reliability.) In Figure 4.3, the transaction is entered at node A. When node A receives acknowledgement (ACK) to all its requests for work, it commits the transaction. If all nodes do not ACK, then node A will abort the transactions which will cause the transaction to back out at each node.

While the main objective of the coordination of updates in a distributed database system is to insure proper application of the transaction and database integrity, another goal is site autonomy. For example, the failure of site A should not prevent processing of a transaction that does not address data at that site.

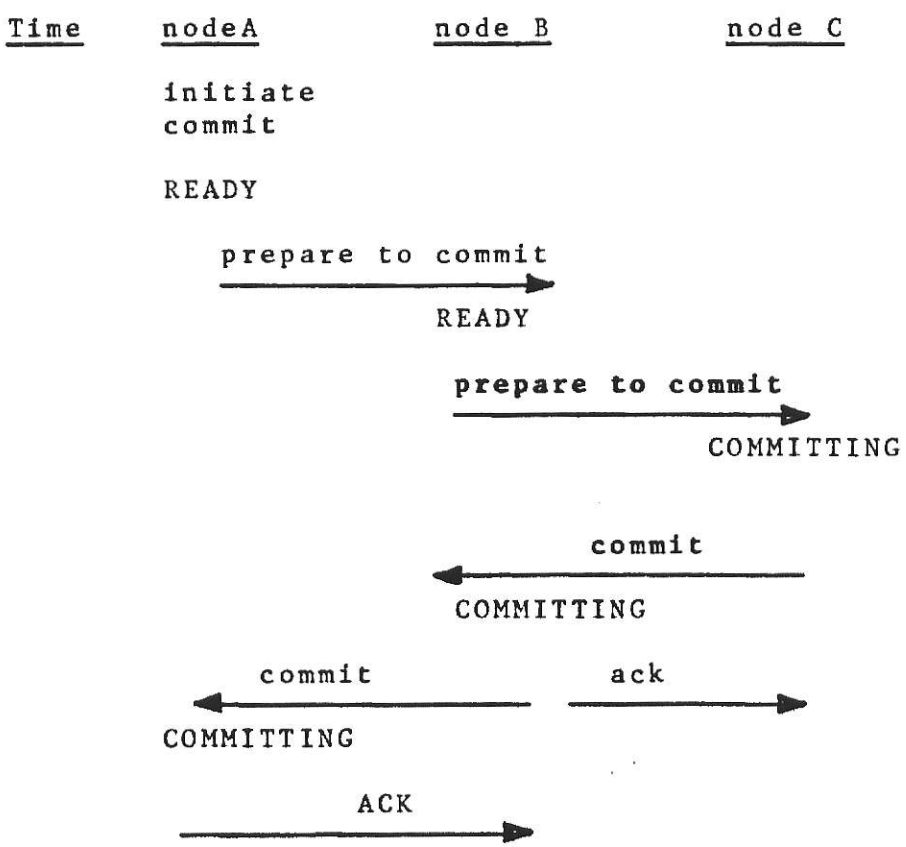


Figure 4.3 Two Phase Protocol

CHAPTER V

COMMUNICATIONS SECURITY

5.1 Introduction

Communications security is not a new subject. The protection of data transmitted over wire lines and/or radio waves has been a matter of concern within the U.S. Military for several years.

Threats to a data communications system vary according to the industry. They fall into three main categories--intentional acts, errors and omissions, and hardware and software failures. Each threat is of concern only in terms of the losses that could result from it actually occurring. In a distributed database system, these threats may cause many adverse affects (Figure 5.1).

It is essential to point out that in a distributed system, cryptography is the only available protection short of physical protection of the entire transmission path. Encryption devices and software routines for computer systems are readily available. The National Bureau of Standards has approved a Data Encryption Standard (DES) that may be used for protection of government and nongovernment information. While the DES is not designed for protection of National Defense classified information, it is adequate for the protection of the privacy of individuals and business proprietary data.

Cryptography, as good as it may seem, is not the final solution for data communication security. In fact, it will only counter those threats listed as intentional acts in Figure 5.1, with the exception of damage to or destruction of equipment (which is a physical security problem).

INTENTIONAL ACTS

- Damage to or destruction of equipment
- Bypassing procedures
- Overriding physical or automated controls
- Wiretapping (passive/active)
- Transaction insertion
- Transaction modification
- Transaction deletion

HARDWARE/SOFTWARE FAILURES

- Misrouted transactions/responses
- Lost (undelivered) transactions/responses
- Transmission errors
- Transmission crosstalk

ERRORS AND OMISSIONS

- Improper equipment operation
- Misaddressed transactions
- Mixing of transactions

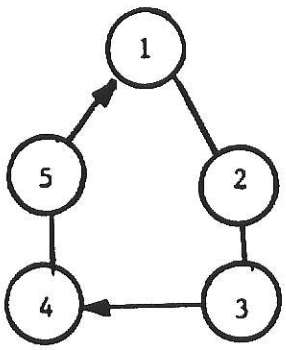
Figure 5.1
Communications Threats

This chapter will then focus on those threats to distributed systems that cannot be countered by cryptography or physical security. The possibility of data disclosure through failure to maintain proper data/transaction flow control will be investigated. I will also attempt to address some of the problems associated with the three distinct network configurations shown in Figure 5.2.

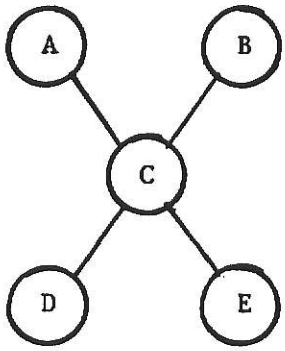
5.2 Flow Control

A flow control policy specifies the channels along which information is allowed to move. Flow controls can prevent the system from disclosing a customer's confidential data. For instance, flow controls can block the transmission of secret military data to an unclassified user. The most general flow controls monitor the detailed flow of data in the system, including the source, transmission path and receiver. However, such controls are very complex and hard to use efficiently. Controls based on security classes are usually efficient, though often very conservative.

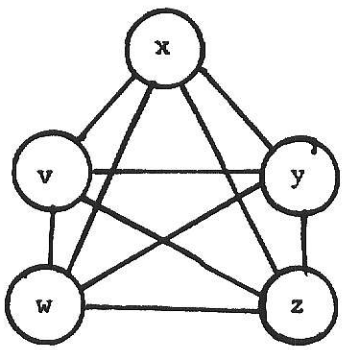
The simplest flow control policy specifies just two classes of information, confidential (C) and nonconfidential (N), and allows all flow of data except from source C to receiver N. Government and military computer systems often have a more complex policy for classified data. In a more complex flow control environment, each security class is represented by a tuple (i,x), where i denotes an authority level and x denotes a category. There may be three authority levels, e.g., confidential, secret and top secret. There are 2^m categories comprising all possible combinations of m compartments;



RING



HUB



STAR

Figure 5.2 Common Network Configurations

typical compartments are (U) unrestricted, R (restricted) and S (sensitive). Information is permitted to flow from an object with security class (i,x) to one with class (j,y) only if $i \leq j$ and only if all the compartments of x are also all the compartments of y.

Simple flow controls can be enforced by an extended access control mechanism, which involves assigning a security class (usually called a clearance) to each transaction. The transaction is allowed to read the database if and only if its clearance (security class) is equal to or greater than the security class of the data. The distributed system then must insure that the results of the transaction are returned over media of the proper security class and returned to the user who entered the transaction and no one else.

The way the user and the system handled the security class is very important to availability of data. All mechanisms based on security classes tend to overclassify information, since only equal or upward flow is allowed. For instance, if user A requests a read on data that is protected as confidential and the system recognizes user A as having a secret clearance, the requested data when returned would be labeled as secret. If that data is then stored in a database belonging to user A, it is stored with a secret clearance and thus is stored under two security classes (confidential and secret). A future update of the now replicated data may sense the difference and upgrade the original database to secret. If the creator of the original database has only a confidential clearance, he has lost access to his own data.

There are ways to control this type of automatic upgrading. The easiest way is through static analysis. This method would be implemented when conflict of security classification was detected during

update. An automatic resolution procedure or one involving the database administrator could then be implemented when conflict is detected. Static analysis would involve the scanning of the database to determine the classification of each data element. The database would be classified at the clearance of the highest data element security class it contained. If contention exists between classification of data elements and the database, automatic downgrading of the database would be allowed. If an element is overclassified, then the determination for downgrading of the security class must be made by an authorized person. A second method is by adding runtime checking to the static analysis. Runtime checking would involve tagging each data element. The system would then monitor the data flow based on the security class of the data element and not according to the clearance of the user. A runtime analysis of the database would be made after each update or modified to insure that the overall security class is appropriate. Static analysis would then be used as described earlier.

To resolve some of the problems of flow control, many system managers have implemented policies of system use. One policy is called "systems high" and involves the clearing of all users, all data processing personnel, and all hardware systems at the highest security class of data being processed. While "systems high" is easy to implement, the costs can be quite high, especially in government systems where a high security class is maintained. Another method called "periods processing" will result in lower implementation costs. This method operates at a particular security class during a set period. During that period, control is as described for "systems high". In distributed database systems, it is not feasible to remove selected data

elements for periods of time. Therefore, "periods processing" is not a feasible solution unless efficient access controls are in place. In light of the discussion on inference access in the previous chapter, "periods processing" may not be a implementation option.

5.3 Error Control

The two major types of data distortion that contribute to data errors are white Gaussian noise and impulse noise. White Gaussian noise is a background hiss caused by thermal activity in lines and amplifiers, and by various modulation techniques. Impulse noise is caused by switching circuit transients, crosstalk and atmospheric conditions. Fortunately, errors resulting from white noise can be anticipated and calculated. However, impulse noise errors are not as predictable; thus, they are usually the primary source of data mishaps.

Echoes are another source of noise on transmission links. They are caused by impedance changes along the line. At points where such changes occur, a transmitted signal is partially reflected back. Although echo suppressors are used on long lines by the telephone company, their effects are partially negated by the use of modems. No device is presently used effectively to eliminate echoes on data transmission lines.

Errors in data links, other than those caused by echoes, can be corrected using a device called data error correctors. These correctors add redundant information to the message to allow error detection and correction capabilities. Because this redundant information is transmitted together with the actual message, it limits the channel's

overall throughput. Error correction devices, for the most part, are based on either retransmission (automatic retransmission request--ARQ) or forward error correcting (FEC) techniques. Both require that redundancy be added to the transmitted data. FEC is used primarily on simplex and half-duplex links requiring continuous forward data transmission. However, FEC often needs as many redundancy bits as data bits to insure data correctness. ARQ's operate on full duplex links. As each data block is transmitted, it is retained in memory until correct reception is indicated on the return channel. If errors are detected, the return channel is used to request retransmission [DANT 79].

Today's distributed database systems demand a degree of reliability and error control that cannot be satisfied by what amounts to a go/nogo test. To achieve a higher level of error monitoring, the software must control performance of the components attached to a node so that it can produce a warning when resources are under-utilized because of excess errors. This must occur before a component's performance problems become severe. In this way, users are aware of affects of transient errors so they can initiate diagnostics or schedule maintenance for error-prone components before serious database damage occurs.

5.4 Network Configuration Implications

This section will address the advantages and disadvantages to the three basic network configurations in Figure 5.2 when various flow control policies are implemented on them. While there may be several variations to these configurations, most of the problems will not vary.

5.4.1 The Ring

The ring network generally employs communications in a simplex mode. If a user at node 1 inputs a transaction that will be employed against the database at node 4, nodes 2 and 3 will have the capability to view the transaction and node 5 can view the results (see Figure 5.2). While this system is efficient in terms of communication costs, it is extremely vulnerable to disclosure of data to users at nodes other than the ones requiring action on the transaction. This network would require the implementation of "systems high" control (as discussed in Section 5.2) to maintain high levels of security. Another method might be to use a network of front-end processors/communications processors to control the communications of the data with the users entering the system via an intermediate processor, such as a general purpose computer (see Figure 5.3). Communication links b would be allowed to see only data destined for the nodal processor.

5.4.2 The Hub

This system requires all transactions to be submitted through a central control or master node. The master node controls switching or connecting of the remote nodes to complete processing of the transaction. Flow control is exercised by the master node and, therefore, more easily enforced. This network is vulnerable to failure. If the master node should fail, there is no alternate path to the nodes in the distributed system.

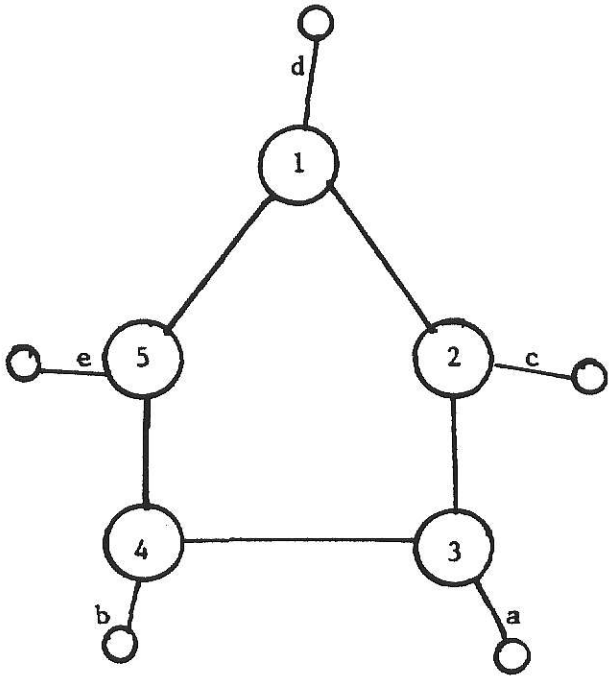


Figure 5.3 Ring Network using Communications Processors

5.4.3 The Star

This configuration is also frequently called totally connected. Each node is connected directly to every other node. Control, therefore, must be distributed to each node; flow control is maintained by each node. While the communications costs are quite high in this network, network resiliency is high. This may be the best solution for operation of a distributed system where multiple security classes are processed concurrently. If nodes x, y and z (Figure 5.2) are authorized to process top secret data and w and v can process secret and confidential, respectively, each node will maintain an authorization table to insure that no data element of a higher classification than the receiving node's authorization is released. For example, node x must restrict release to node v at the confidential or below level. This provides greater security because the node containing the data has absolute control over its release.

5.5 Conclusions

To apply cryptography to resolve all communications security problems is an over-simplified approach. Resolution of data flow questions and development of reliable software to insure maintenance of security classes in distributed systems are major problems and will require attention during the design phase of the distributed system. Techniques for correction of errors in data transmission are well known and presently in use on most distributed systems. However, little has been done to identify expeditiously those error prone components and advise users of possible data error conditions. This problem may be a

large task for the data communications system, but one that must be addressed if data integrity is to be maintained.

BIBLIOGRAPHY

ABBO 79 ABBOTT, ROBERT P. Honeywell Information Systems, Computer Security and Privacy Symposium Proceedings, 1979.

BAYE 76 BAYER, RUDOLF. On the Integrity of Data Bases and Resource Locking, Technische Universitat Munchen, 1976.

DANT 79 D'ANTONIO, RENATO A. The As and Bs of Error-Free Data Communications, Data Communications, 1979

DENN 79.1 DENNING, DOROTHY E. and DENNING, PETER J. Data Security, ACM Computing Surveys, 1979.

DENN 79.2 DENNING, DOROTHY E., DENNING, PETER J. and SCHWARTZ, MAYNERD. The Tracker: A Threat to Statistical Database Security, ACM Transactions on Database Systems, 1979.

DENN 80 DENNING, DOROTHY E. and SCHLORER, JAN. A Fast Procedure for Finding a Tracker in a Statistical Database, ACM Transactions on Database Systems, 1980.

DOLL 78 DOLL, DIXON R. Data Communications Facilities, Networks and Systems Design, John Wiley and Sons, 1978.

HOFF 78 HOFFMAN, LANCE J. Security and Privacy in Computer Systems, Melville Publishing Company, 1973.

LEVE 82 LEVEILLE, GREG R. Data Base Security Systems: Protecting the Source, Todays Office, October 1982.

MENA 79 MENASCE, DANIEL A., POPEK, GERALD J. and MUNTZ, RICHARD R. Centralized and Hierarchical Locking in Distributed Databases. A paper presented to the Advanced Research Projects Agency of the Department of Defense, 1979.

McGL 78 McGLYNN, DANIEL R. Distributed Processing and Data Communications, John Wiley and Sons, 1978.

ROTH 77 ROTH, DANIEL J. On Preserving the Integrity of Databases, The Computer Journal, 1977.

SCHL 75 SCHLORER, J. Identification and Retrieval of Personal Records from a Statistical Data Bank, Methods of Information in Medicine, 1975.

SECURITY IN DISTRIBUTED DATA SYSTEMS

by

STEVEN D. FINCH

BGS, University of Nebraska Omaha, 1976

AN ABSTRACT OF A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTERS OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

ABSTRACT

This report presents an analysis of security in distributed database systems and discusses possible vulnerabilities and countermeasures. Chapter II (access control) presents the traditional methods of access control and discusses problems with those methods. SYSTEM R* and INLAND are presented to demonstrate newly developed systems where access control has been engineered into the system. Chapter III discusses the problem of inference access control and presents the author's view on possible problems in a distributed system. Integrity of the database is discussed to include hardware and software induced errors and errors induced through unauthorized manipulation by users. The chapter on communications security focuses on the software aspects of maintaining data flow and effective error control. Additionally, a conceptual look at the weaknesses and strengths of the ring, star and hub networks is included.