LEVELS OF PROTECTION AND ASSOCIATED OVERHEAD IN THE FORMULARY PROTECTION SYSTEM

by

LYLE KLEOPFER

B. A., TABOR COLLEGE, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1975

Approved by

Virgil E. Wallentine
Major Professor

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to acknowledge the assistance given by Dr. Virgil Wallentine, my major professor, Dr. Paul Fisher and Dr. Myron Calhoun in the preparation of this report. I especially want to thank Dr. Wallentine for suggesting the topic, basic format and proofreading the report.

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

In recent years large computer data banks have moved from a place in the future to a very real part of today's world. No longer do companies depend on clerks to do bookkeeping, most company records are now maintained by a computer. Many people now expect that a nationwide computer information system will be developed in the very near future. A real cost savings will be realized due to the fact it is much less expensive to share information than to reproduce it. Already many companies depend on a central facility to provide information almost instantaneously for management at remote terminals.

Much of this information stored on these on-line files can be considered confidential. Divulgence of such information to company personnel who have no real need to know or to intruders poses a real threat to an information system. Thus protection of the files and verification of attempted accesses is a subject of vital importance.

Protection refers to the logical and physical mechanisms for controlling access to data. The purpose of a protection system is to guarantee that during the execution of a program each attempted access to a data object be verified as authorized. Additionally, the protection mechanism must verify the mode of the access, for example 'read-only' access, to a particular object. The protection system

must be designed in such a way as to guard against failures in either hardware or software that would leave data records unprotected.

Thus the object of a protection system is to devise mechanisms which will provide protection to the greatest extent possible at the lowest cost in terms of hardware usage, run-time or program size. One comment has been made by Farr concerning this area: "An information system can never be completely safe but the cost to an intruder to gain information can be made higher than the value of the information to be gained." (3).

At this time very few cost justification or overhead studies of protection systems have been done. This is due, in large part, to the difficulty in isolating (identifying) distinct parts of the standard protection system which induce overhead and the difficulty in separating the protection system from general operating system overhead. One additional factor needs to be considered when analyzing the costs of the usual protection system; and that is, to upgrade the level of protection usually entails extensive modifications to the overall system.

One protection system model which readily lends itself to both cost evaluations of different modules and modifications of existing protection levels is the formulary protection system model developed by Lance J. Hoffman (7). This ease of measurement and extensibility of protection is due primarily to the modularity of the system. Thus to determine

overhead due to one part, only that module needs to be measured. Additionally, to upgrade the level of protection the user only needs to change one or more modules. Thus, this degree of modularity lends itself to measurement of overhead attributed to the protection system using a software monitor, of which many are commercially available (4).

This report is based on the formulary model of flexible privacy and access controls presented by Lance J. Hoffman in Report 117, Stanford Linear Accelerator Center, Stanford, California, 1970. The objectives of the work reported in this document are as follow:

1. To acquaint the reader with the characteristics of the formulary model,
2. To identify the major levels of protection and sublevels within each in the formulary system,
3. To identify the overhead that may be attributed to the presence of the protection system.

CHAPTER II

DESCRIPTION OF THE FORMULARY MODEL

This chapter presents a comparison between protection and sharing of information in a conventional file system and in the formulary system. The basic modules of the formulary system and the interfaces between modules at execution time are described.

CONVENTIONAL FILE SYSTEM VERSUS FORMULARY SYSTEM

In most information systems which are concerned with privacy only a password is used to protect confidential data. With each file is associated a password which identifies the user and his access rights. Problems arise through the use of passwords which can be copied, lost, stolen or in other ways fall into the hands of a potential intruder. The protection provided by passwords could be compromised by the use of a mini-computer located at a remote terminal which simply tries passwords at random until finally a correct one may be determined. With the speed of the newer mini-computers the protection scheme can be broken within a relatively short time at a low cost.

These systems all make one assumption--that all the information in the confidential files is of equal sensitivity. This clearly is not the case since often one user may need to see part of a large file while being denied access rights to another part. In the conventional file system the information which could be shared often has to be reproduced to safeguard

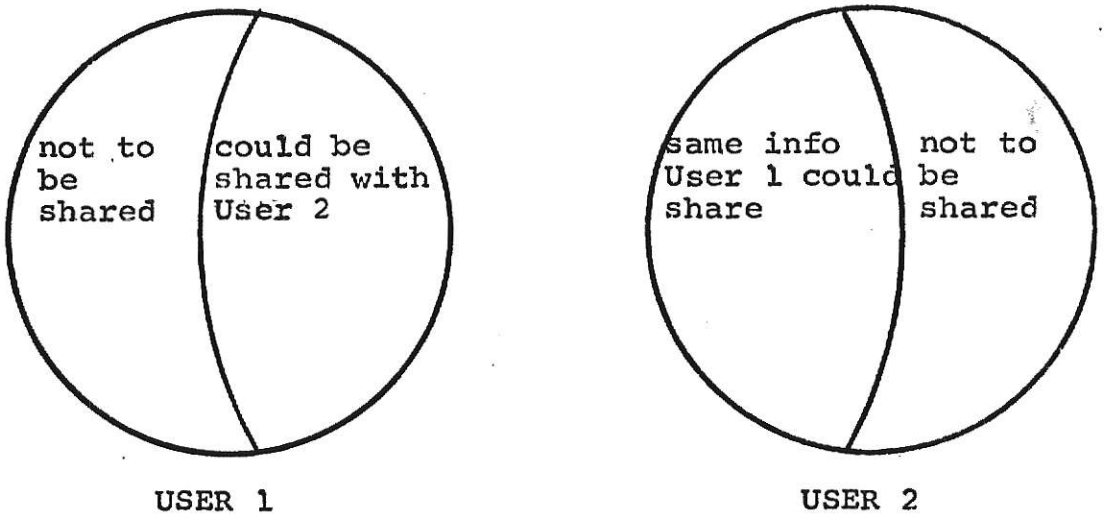
the confidential data (See Figure 1a). As was stated earlier it is much cheaper to share than to keep copies. The problem then becomes one of guarding both the shared and unshared parts from unauthorized access.

Many methods for authorization of access have been proposed, among these is the access matrix (5). This and all other commonly used methods only provide protection at the file level not at the actual data level. These methods use static checking of access rights. That is, when the file is created, capabilities are assigned to the users of the file. A better solution to the problem would be authorization checking at actual data access time and allowing sharing of data below the file level.

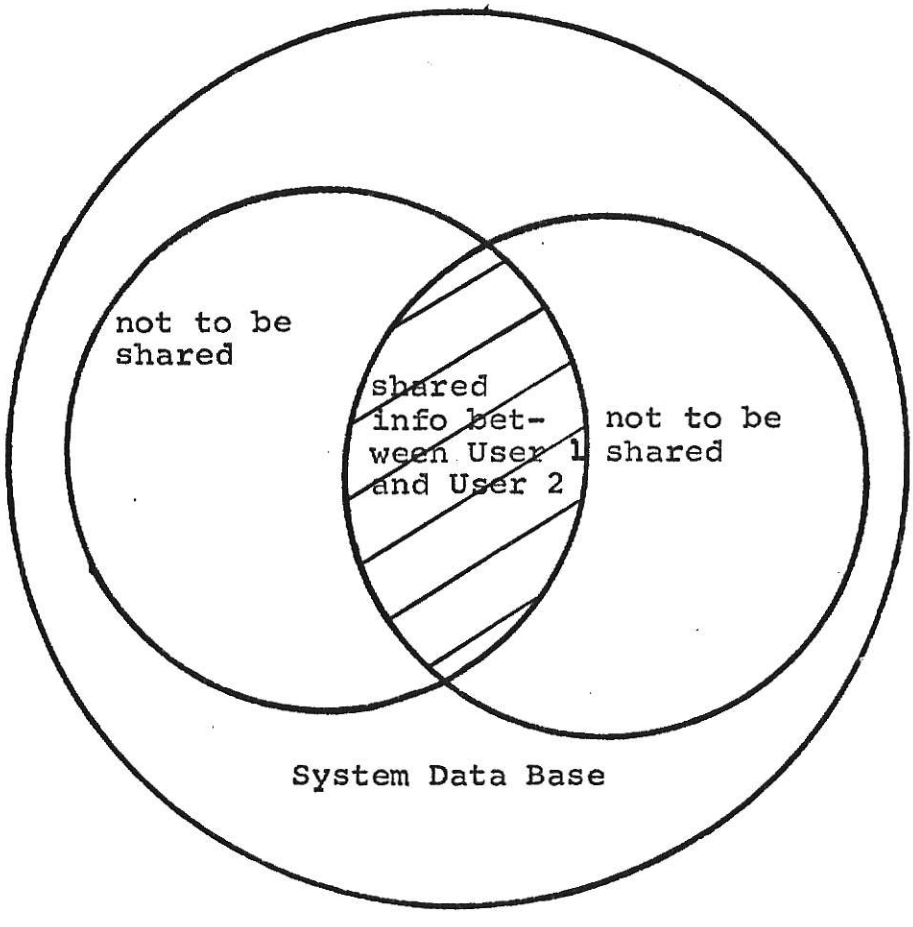
One system model which meets these criteria has been presented by Hoffman (7). The model provides protection of data below the file level and does access authorization checking at run-time. Users may share information with confidence that sensitive data is secure, thus saving a significant amount of space (See Figure 1b). Upgrading the level of protection without totally redesigning the system and measuring the overhead attributed to individual modules are possible due to the modularity of the system.

BASIC SYSTEM MODULES

TALK is the first module to be contacted by the user. Basically, this module provides the user with a bridge between his terminal and procedures which access data. TALK may



(a) Conventional File System



(b) Formulary System

FIGURE 1: Sharing of Information in Formulary System vs Typical File System

have to be a relatively complex procedure in the case of the novice user where extensive conversation may be necessary to determine what data the user wants to access. With the sophisticated user, TALK may be a very simple program. Three parameters are input to this procedure by the user: the description of the data to be accessed, the operation the user wishes to perform, and possible user identification or other information about the user and terminal combination. Basically, this module provides the link between the user's and the system's perception of the data base. TALK uses the parameters to generate what might be called a full path name for the data described by the user.

The next basic module is ACCESS. This procedure provides the message switching among the TALK modules, system primitive functions and the modules which control the access to a particular piece of data. Parameters to this procedure are the user information, the internal name of the data, address where the data is stored, length of the field, the operation to be performed and a place for the completion code to be returned. The basic operations called by ACCESS include FETCH which fetches information from the data base, STORE which stores information, FETCHLOCK and STORELOCK which lock the particular data from other users of the system, UNLOCKFETCH and UNLOCKSTORE which guarantee the integrity of the data accessible to concurrent users of the system. Finally, there is an ATTACH operation which attempts to attach a user and terminal combination to a formulary subject with

the user's capabilities as determined by the default system formulary and a DETACH operation which breaks an already existing attachment.

The last basic module consists of the formulary itself. This procedure consists of a set of subprocedures which control the attempted accesses to the information in the data base. This module is used for every attempted operation. Many different formulary modules may exist in the same system due to the diverse needs of the many users. How the user wishes to control access to his part of the data base determines which formulary is to be used.

The formulary modules consist of four submodules, VIRTUAL, SCRAMBLE, UNSCRAMBLE and CONTROL. VIRTUAL takes the internal name of the data and converts it to a virtual address. SCRAMBLE and UNSCRAMBLE provide the scrambling before the data is stored and the unscrambling immediately following a fetch.

The CONTROL procedure is the real heart of the protection system. It is the module which determines whether to actually allow the attempted access to the data. Parameters passed to this procedure are the internal name of the data and the desired operation to be performed. If the operation is permitted, a completion code of 1 is passed back to ACCESS. Otherwise, a larger number is returned indicating the reason for the failure.

One last module exists in the system. This is the system procedure which allows the system programmer to build new formularies and insert them in the system.

EXECUTION TIME MODULE INTERFACE

As was explained earlier, in order for an actual data transfer to take place, a user at a particular terminal and a system formulary must be linked together. When a user signs on at a terminal, he is initially attached to a default formulary. After the initial identification, the system requires that the user actually attempt to ATTACH to a particular user formulary. The only operation that this default formulary will permit is an ATTACH; it does not permit any attempts to fetch or store from this level. The CONTROL module of the default formulary can be arbitrarily complex to guarantee that only a qualified user ever be allowed to attach to a formulary which actually accesses data. At this point where attachment is actually granted, information about the user formulary replaces the default formulary information in the ACCESS module. After attachment the ACCESS module contains pointers to the SCRAMBLE, UNSCRAMBLE, VIRTUAL and CONTROL areas of the new formulary. The control of access is now totally dependent on the newly attached formulary. This attachment can be broken by the user by either requesting a DETACH operation or by logging off at the terminal.

The general flow of control through the modules is as follows: the user at the terminal describes a certain piece of data that he wishes to access and the operation to be performed on it. The TALK procedure immediately converts this description to an internal name which is passed as a parameter to ACCESS which in turn calls on CONTROL to validate

the attempted operation. If the operation were one that deals with the locking or unlocking of data for concurrent processing, the operation is attempted. Should this operation fail, a message is returned to the user. If the operation were to fetch, the data is transferred to a buffer and unscrambled using the UNSCRAMBLE procedure. Or, with the store operation the data is scrambled and then finally stored in the data base (See Figure 2).

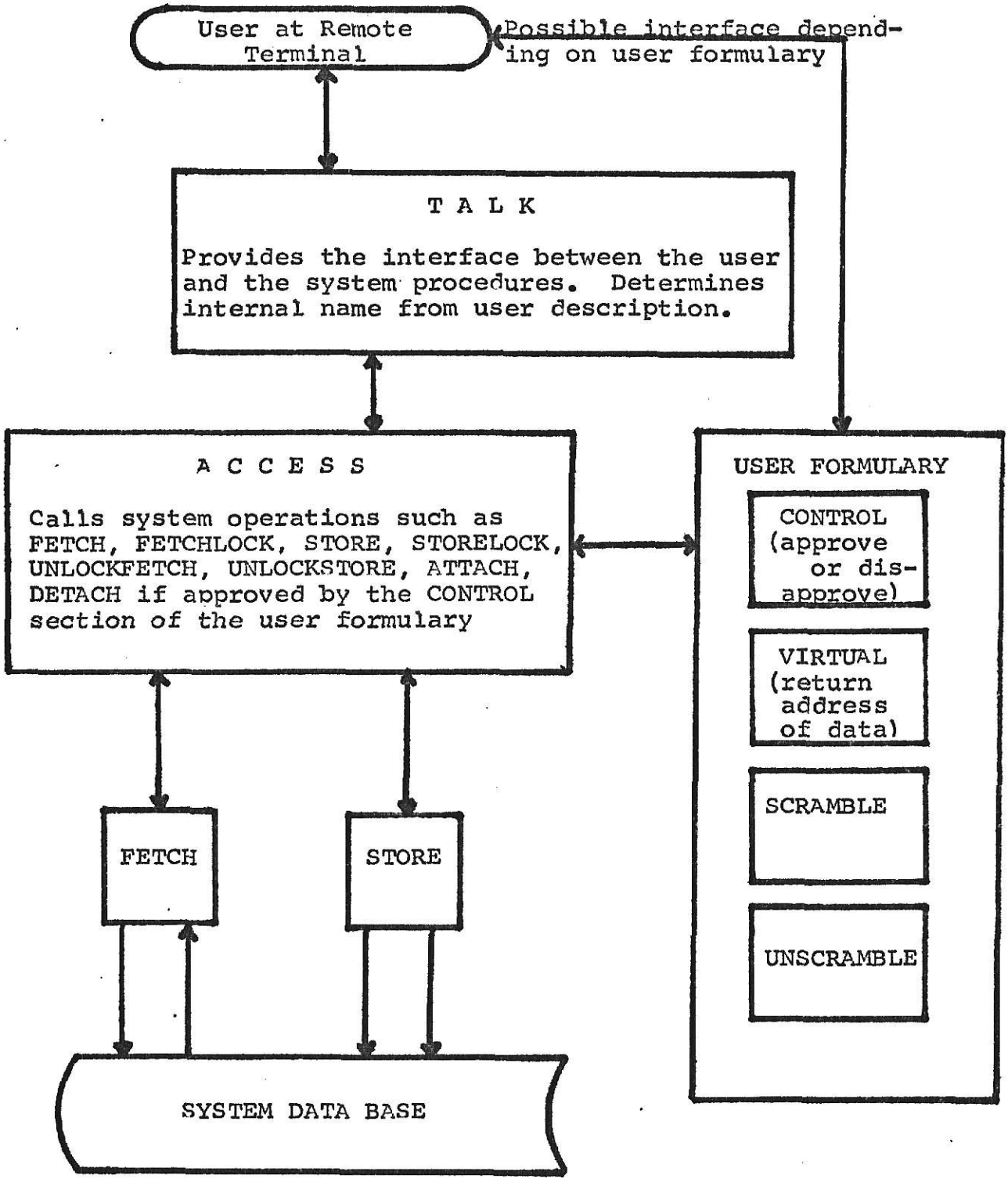


FIGURE 2: Structure of the Modules and Interface of the Formulary System.

CHAPTER III

LEVELS OF PROTECTION

Many levels of protection are theoretically available through the use of the formulary model. This chapter will identify the major levels and some of the possible sublevels in each.

DEFAULT SYSTEM PROTECTION

The first level of protection depends on the default system formulary. This can be identified as a level of protection which is comparable to some of the current password schemes implemented in many systems. The system formulary could require as little as a password for an attempted ATTACH to a user formulary. Another method is to make the ATTACH a time or terminal dependent operation, for example, only during certain times during the working day would valid attach attempts take place from certain terminals.

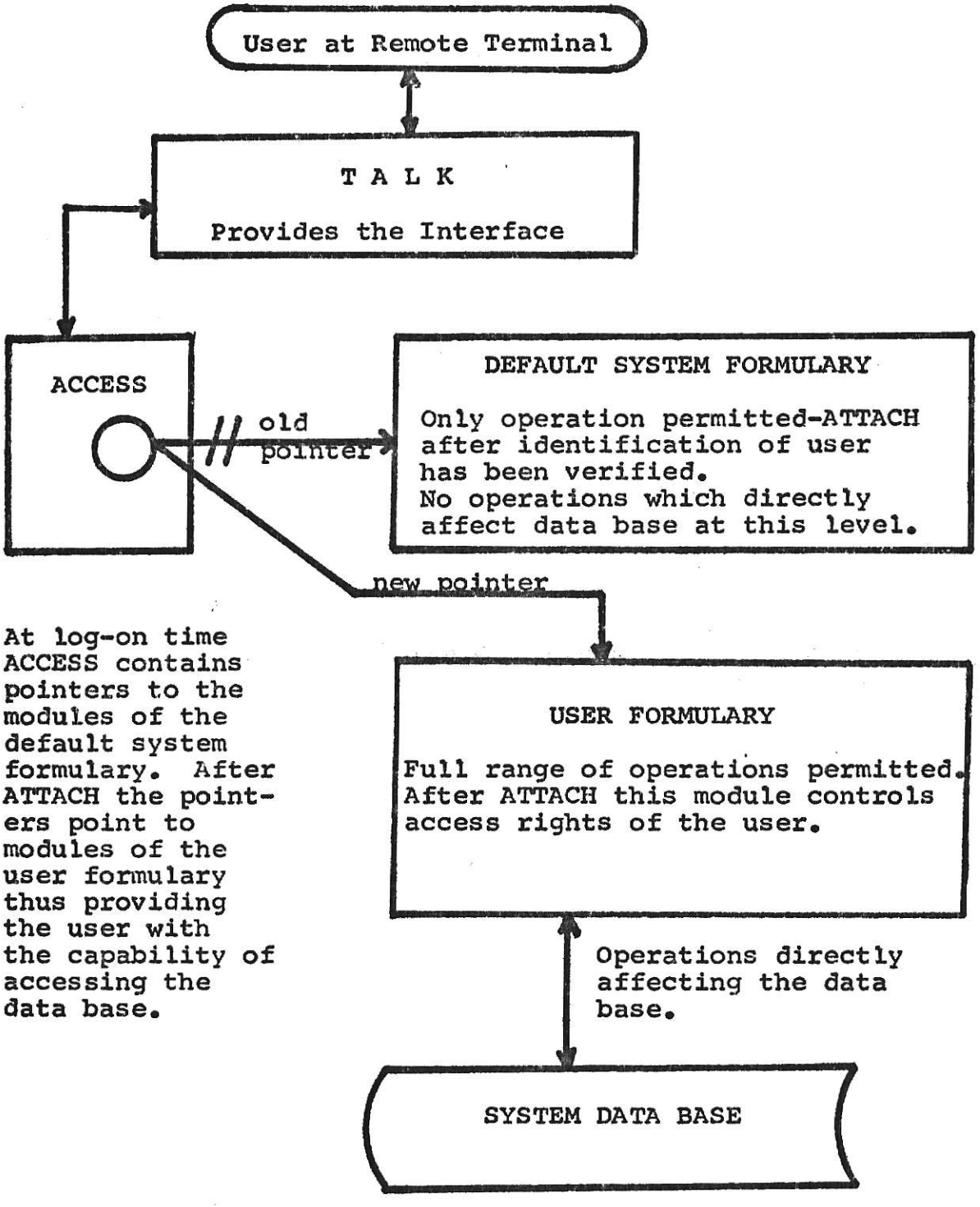
Several common methods of protection could be incorporated at this level. One method to validate the attempted attach is through the use of the common access matrix (5). This method could make the decision based on the capability list of a user. At this level it is crucial that the user be correctly identified before the operation using the access matrix is attempted. This is accomplished with relative ease by using various password schemes combined with a terminal and time dependence. Another method of identification is the non-reusable password. The same problem exists here. The password

list could fall into the hands of an intruder. One method that does alleviate this problem somewhat is to have the user generate a password by using the time of day, possibly day of the month, and a simple mathematical formula.

Another distinct possibility exists at this level. Mini-computers have been suggested as a way to break a security system. One method commonly used to prevent such occurrences is to record all attempts to log-on. The system could be designed in such a way as to give the user a reasonable amount of attempts to log-on before sounding an alarm or switching formularies as discussed in Chapter IV.

Even if an intruder somehow manages to be identified as a legitimate user, he still has to have knowledge of formulary names and general system procedures to actually become attached to a user formulary. At this level he still can do very little harm since he cannot access the data base. Should an attempt to fetch or store be made at this level, the user could be logged off by a sophisticated system and an alarm sounded or the system protection can be upgraded. Thus, at the lowest level, the intruder can not really cause any harm to the data base and can be caught in most instances. Figure 3 illustrates the ATTACH command.

The TALK procedure can be structured to provide some protection to the data base. It can be structured to reply to the requests of the user so that very little system information is divulged, providing a measure of protection. The intruder may be led to believe that the data does not exist when an attempted access is trapped by the system.



At log-on time ACCESS contains pointers to the modules of the default system formulary. After ATTACH the pointers point to modules of the user formulary thus providing the user with the capability of accessing the data base.

After the ATTACH the user has the capabilities granted to him by the user CONTROL module to access only this formulary's portion of the data base.

FIGURE 3: ATTACH Command, Default System Protection

Changing the identification procedure periodically would lessen chances that an intruder could become familiar enough with the system to gain large amounts of information over a long interval of time. The modularity of the system will accommodate this change.

USER LEVEL OF PROTECTION

The second level of protection exists within the structure of the user formulary itself. Consider the example of an intruder who somehow has become attached to a user formulary. This attachment does not guarantee access to the data base. He must still understand how each user formulary control section operates. A completely different method of validation of attempted accesses could be used here by the user; thus each user formulary would be different. Some could be terminal dependent, time dependent, or possibly dependent on an identification scheme for each attempted operation. Again, this level provides the flexibility to make changes in the user formulary.

Thus, at this level a great many protection levels are possible by changing the complexity of the user formulary. As these procedures become more and more complex, more time would be spent in the module thus adding to the total system overhead.

Even at this level the intruder can do only a limited amount of damage to the data base. The only data that can be accessed here is that which has been included in this

formulary which might be a very small part of the total data base.

MOST SECURE LEVEL OF PROTECTION

The third level of protection exists within the SCRAMBLE and UNSCRAMBLE procedures in the user formulary. Again consider the example of an intruder who somehow has managed to not only attach to a valid user formulary, but has actually had an operation to be performed approved by the control module. Thus, in even the most secure system he could well have the desired information in a location which would be accessible to him. Still, the information may not be in a usable form. The information may still have to be unscrambled by the key which originally encrypted it. The model could be changed to the point where it would be necessary for the user to supply the key. At present the formulary model would automatically decode the information using UNSCRAMBLE. With the present model the scrambling protects data primarily in the case of a software failure. An example is VIRTUAL returning an incorrect address.

Thus, by changing the model slightly to where the user would have to supply the key for UNSCRAMBLE, three definite pieces of information would be required before any data in the system could be returned in usable form. The first would be a valid identification and formulary name to supply to the system formulary before an attachment could take place. Secondly, the intruder would be required to have

knowledge of how the user formulary validates attempted accesses. At any of these levels random attempts could signal an alarm. Thirdly, by having to supply the key for UNSCRAMBLE to use, the intruder must be familiar with how the information was scrambled initially.

Thus, the formulary model does provide a very flexible method of protection of data in a large data base using many remote terminals. Each of the modules the user must use could be constructed differently thus foiling random attempts to penetrate the system. Redesigning the modules periodically is another means of frustrating the intruder. Additionally, any of the control modules could utilize the capabilities of commonly used protection devices.

A complete range of protection levels are available in the same system (See Figure 4). At the lowest level the protection provided is comparable to a password system. At the upper ranges, through the use of several different types of protection devices within the formulary model and more complex encode and decode modules, a very high level of security is possible.

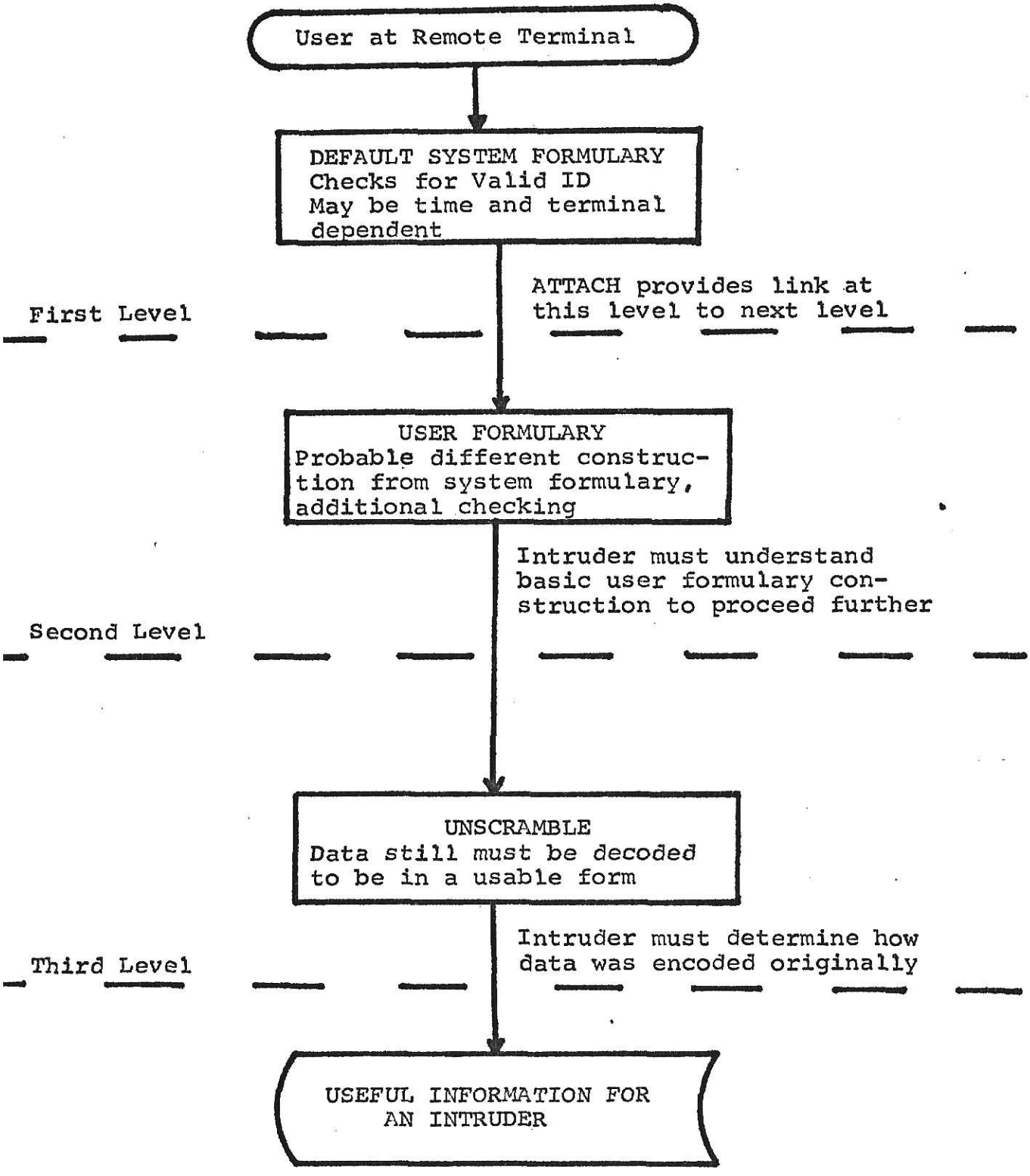


FIGURE 4: Levels of Protection in a System Employing all Possible Levels.

CHAPTER IV

OVERHEAD ATTRIBUTED TO THE PROTECTION SYSTEM

Of equal importance with the levels of protection is cost, or the level of cost versus the level of protection. This cost can be attributed primarily to three things--CPU, memory and channel usage. This chapter will describe where the above three costs are encountered, under what conditions they are most critical, and possible ways to measure the impact of the overhead.

An internal software monitor can be utilized to produce very accurate results in terms of additional overhead attributed to the protection system. The software monitor is an operating system modification which is capable of collecting and recording information about different areas of the system environment. Many software monitors of this type are commercially available (4).

A feedback mechanism could be built directly into the ACCESS module associated with the protection system to determine when the system is under attack from an intruder and take appropriate actions. ACCESS would call a system routine to adjust the protection level upward. Of course, as the level of protection increases the overhead associated with the upgraded protection level would be expected to increase.

CPU OVERHEAD

Any protection system will require a certain amount of CPU usage to be able to validate attempted accesses. In the most secure system each attempted access needs to be validated to guard against what is commonly called "piggy-backing" which refers to tapping into the communication network between the user and the central facility. The intruder will allow the user to establish proper identification with the system and then use the line to obtain protected information. It is important that every access be verified to guard against such intrusions. Thus, the CPU overhead will increase proportionally with the complexity of the module.

A software monitor can provide the user with information concerning CPU usage. Generally, the monitor will record the time a given module is entered and the time it is exited. Many events which are not included in our first model can occur between the time of activation and the exit from the procedure. One way to insure the accuracy of the measurement is to record on tape the times of every event and the times of return to the module. Then through an auxiliary program the event tape could be stripped obtain the actual time spent in the module.

CPU usage is probably the most important of the three criteria. For every attempted access the protection program must be executed. Thus, even a small reduction in the execution time of such a program would result in a significant

savings due to the large amount of times the program is executed. By using the monitor additional CPU usage due to an upgraded protection system could be calculated by comparing the difference in times of execution between the old and new systems.

Another criteria which contributes to total CPU overhead is the one-time cost of linking to an existing module. TALK and ACCESS modules must be executed once for each request from the user. The CPU time used by the possible scrambling and unscrambling of information and time to search the tables of VIRTUAL also contribute to total CPU cost. Page faults will occur during the execution of the formulary. The monitor could accumulate the number of faults for the formulary. From this total a reasonable estimate of CPU usage can be calculated for system resolution of these page faults. Finally, there is the cost of identifying the potential user before allowing attachment to a user formulary.

CHANNEL OVERHEAD

Next channel usage must be considered. The first example of this is channel usage due to the many different TALK procedures used by many different users. This procedure probably would be paged since many users could be active at the same time using different TALK procedures with a relatively long time between requests for each user. By paging, memory usage could be minimized. The ATTACH command requires that a user module be brought into core from backing store. Additionally, page faults during execution

of the CONTROL subsystem will result in increased channel usage. The channel must also be used to interface with the remote terminals. Thus, as the complexity of the formulary increases, more and more messages will have to be sent and received from the user at a remote terminal.

MEMORY USAGE

The last area of the system overhead to be evaluated is memory usage, both primary and backing store. The size of the overhead will vary widely with the individual formulary as it did with the previous two criteria. With a relatively simple formulary a small amount of memory would be required. With a complex system this amount could become quite large due to the sophisticated CONTROL mechanism. Large tables are often associated with the VIRTUAL module of the user formulary. Memory is required by the various TALK and ACCESS procedures. Large amounts of backing store are required to store the formularies when not in use. Here again the software monitor can provide information concerning the memory allocations to various system tasks.

MANAGERIAL CONSIDERATIONS

Through the use of the software monitor some bare facts can be obtained about how much CPU, channel and memory usage a given formulary needs. The manager of the central facility must look further into how implementation of such a formulary could affect the existing system. He can better analyze the trade-offs between the three criteria through use of the data obtained.

If the existing system is using a high percentage of the CPU cycles available it would be doubtful if the protection system could be implemented without some degradation of service. If the CPU utilization is low the system would seem to be in a position to accept the additional overhead with minimal degradation. But channel usage is directly related to CPU utilization. Should the channel usage in the existing system be quite high, the additional channel usage attributed to the protection system could lead to thrashing--extremely high channel usage and low CPU usage. If the channel usage is quite low the system probably would accept the protection system with minimal degradation. In a system where memory space is not critical the formulary system implementation would usually not result in significant degradation due to memory usage. However in the situation where available memory is at a premium, the procedures often are paged. Thus, it becomes a trade-off between memory usage and channel usage. Table 1 illustrates some common system problems and their causes.

A library of general formularies could be maintained by the central facility. Thus management could offer a wide range of types and levels of protection to a new user. By using a monitor to evaluate the overhead involved in a given formulary, management would be able to provide the user with an answer to the cost versus level of protection question. A list of costs associated with each formulary could be maintained so that the user could see where the

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.

SYMPTOMS	HIGH CPU	LOW CPU/CHANNEL OVERLAP	LOW CPU (WAIT)	LARGE SEEK	HIGH CHANNEL USE	LOW CHANNEL USE	LOW CHANNEL OVERLAP	CHANNEL IMBALANCE	LOW MULTIPROGRAMMING	INADEQUATE RESPONSE TIME
HIGH CPU	3	1,2 3								
LOW CPU (WAIT)			4	2						
LOW CPU/CHANNEL OVERLAP	1,2 3	9								
LARGE SEEK			2	5,6 10						
HIGH CHANNEL USE					5,6 9		8	8		
LOW CHANNEL USE						10	7	7		
LOW CHANNEL OVERLAP					8	7	2,8			
CHANNEL IMBALANCE					8	7		2,8		
LOW MULTIPROGRAMMING									1,4, 11,13	
INADEQUATE RESPONSE TIME										3,9, 11,12

CAUSES:

1. IMBALANCED SCHEDULING
2. POOR FILE PLACEMENT
3. INEFFICIENT CODING (APPLICATIONS)
4. INEFFICIENT OPS
5. EXCESSIVE PROGRAM LOADING
6. INEFFICIENT DATA BLOCKING
7. EXCESSIVE CHANNEL CAPACITY
8. POOR DEVICE/CHANNEL PLACEMENT
9. CPU BOUND WORKLOAD
10. I/O BOUND WORKLOAD
11. LIMITED CORE CAPACITY (INADEQUATE)
12. LIMITED CPU CAPACITY (INADEQUATE)
13. LIMITED I/O CAPACITY (INADEQUATE)

Table 1

COMMON SYSTEM PROBLEMS AND CAUSES (12)

costs are incurred. In the case of a user-written formulary an approximation of the cost could be obtained by monitoring the formulary to arrive at estimates for CPU, channel and memory usage.

By providing this type of service the manager should be able to provide a protection level directly dependent on the amount the user is willing to spend. These modules could be made to fit the particular needs of the user without changing the basic structure of the modules.

PROTECTION LEVEL ADJUSTED BY FEEDBACK

A feedback mechanism could be used in conjunction with a system routine to furnish a very flexible level of protection which would be automatically adjusted to the needs of the user (See Figure 5). The system would start with a formulary offering the minimal level of protection and then upgrade protection levels only when attempts to penetrate the system are uncovered. This system results in a substantial savings for the user since the lower protection levels have much less overhead associated with them. In this flexible system the feedback mechanism is used to accumulate the number of unsuccessful attempts to ATTACH or ACCESS. As the count of these unsuccessful attempts increases it is assumed that a systematic attack on the protection system may be underway. When this count reaches a predetermined threshold level it would signal the system routine to increase security. The legitimate user could

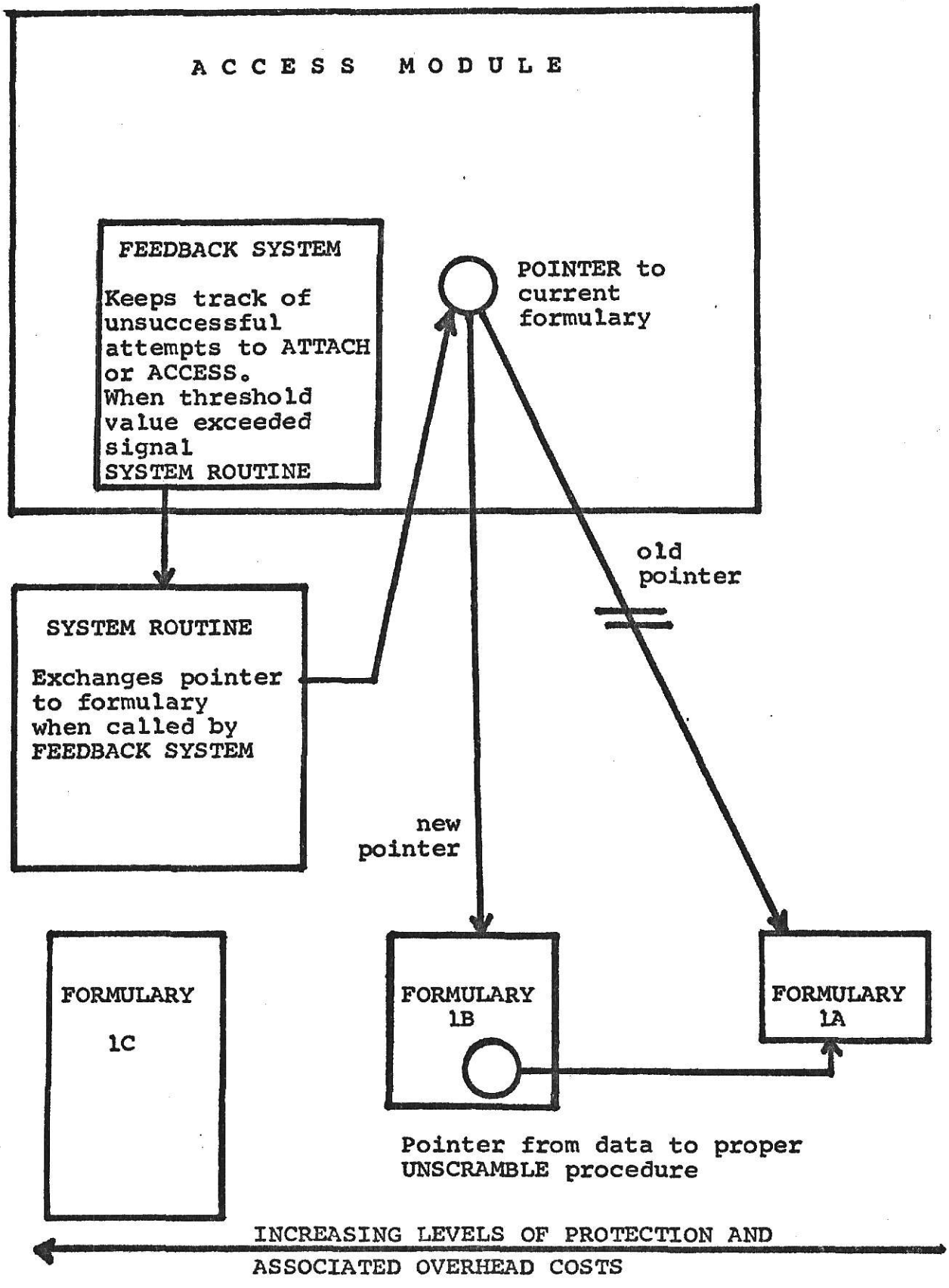


FIGURE 5: Use of a Feedback Mechanism to Determine Threats and Adjust Protection Levels.

set this level at formulary creation time. It is the user who actually knows the value of the data to be protected and how much he is willing to spend to protect the data.

Basically, the system routine would upgrade the level by exchanging formulary modules--the latter one would be a more complete system--one that does more checking and more extensive encryption. This can be done quite easily by changing the entry in the ACCESS module to point to the new formulary. If the SCRAMBLE and UNSCRAMBLE procedures are changed by upgrading the protection level, a problem becomes apparent. Data originally encrypted at the lower level can not be decoded by using the UNSCRAMBLE procedure of the new formulary. One solution to the problem is to maintain with the data a pointer to the UNSCRAMBLE routine of the formulary which originally encrypted the data. Thus, several levels of increasing protection could be available.

This system would require that several system formularies for the same set of data be maintained. This protection system requires keeping a set of formularies with the same general name but with each having an increasing protection level. The system can be continuously monitored to determine if the level of threats have once again fallen below the critical level. Then a system routine could once again restore a lower protection level.

CHAPTER V

DISCUSSION

This report has covered the basic characteristics of the formulary protection system. This includes the basic modules and the interface between the modules at execution time. The levels of protection available and the overhead attributed to the protection system have been identified. Finally a proposal for a system with automatic adjustment of protection has been presented.

In the author's opinion as the extremely large data banks become more and more common, the need to share information but yet protect it from intruders will be of critical importance. The need for such protection will lead to many protection systems being developed. Thus, a good research topic appears to be development of additional protection systems which will allow sharing with a high level of protection. The formulary system could be implemented to actually determine what the trade-offs in terms of CPU, channel and memory usage are and how they affect total system performance. Determination of additional overhead due to the presence of the feedback mechanism in the ACCESS module would be a suitable research project.

BIBLIOGRAPHY

1. Balcom, Philip; Cranson, Gary; USACSC Software Computer System Monitor: SHERLOC, US Army System Command, Ft. Belvoir, Virginia.

2. Cochrum, J. S.; Crochett, E. D.; Interpreting the Results of a Hardware System Monitor, Memorex Corporation, Santa Clara, California.

3. Farr, M. A.; Chadwich, B.; Wong, K. K.; Security for Computer Systems, Manchester, National Computing Centre Limited, 1972.

4. Gotlieb, C. C.; Performance Measurement, Department of Computer Science, University of Toronto, Canada.

5. Graham, G. Scott; Denning, Peter J.; Protection: Principles and Practice, Technical Report Number 101, November, 1971.

6. Hoffman, Lance J.; Friedman, Theodore D.; "Execution Time Requirements for Encipherement Programs", Communications of the ACM, August, 1974.

7. Hoffman, Lance J.; The Formulary Model for Flexible Privacy and Access Control, Edited by Lance Hoffman, Los Angeles, 1973.

8. Hoffman, Lance J.; "Computers and Privacy: A Survey", Computing Surveys, Vol. 1, No. 2, June, 1969.

9. Needham, R. M.; Protection Systems and Protection Impementations, Edited by I. L. Auerbach, Philadelphia, Auerbach, 1973.

10. Turn, Rein; "Privacy Transformations for Data Bank Systems", (AFIPS Conference Proceedings), 1973, Vol. 42.

11. Turn, Rein; Shapiro, Normal; "Privacy and Security in Databank Systems Measures of Effectiveness, Cost and Protector-Intruder Interactions", (AFIPS Conference Proceedings), Fall, 1973.

12. Venese, Daniel M.; A Methodology for Performance Measurement, MITRE Corporation.

LEVELS OF PROTECTION AND ASSOCIATED OVERHEAD IN THE FORMULARY PROTECTION SYSTEM

by

LYLE KLEOPFER

B. A., TABOR COLLEGE, 1971

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1975

During the last several years protection of confidential information in large data banks has become an issue of critical importance. This report describes and extends the structure of the formulary protection system. This system allows sharing of confidential information with authorized users thus eliminating the need to keep copies of the same information. The modules of the system and the interface between individual modules at execution time is discussed.

The three major levels of protection and inherent sublevels in each are described. Examples of where commonly used protection mechanisms could be used at each level are included.

The overhead associated with a protection system is discussed. This is primarily attributed to CPU, channel and memory usage. The effects of this overhead on total system performance are discussed. Finally, a system using an automatic feedback mechanism to provide adjustable protection levels for the user is presented.