This blank page was inserted to preserve pagination.

PRIMARY ACCESS CONTROL IN LARGE-SCALE TIME-SHARED DECISION SYSTEMS

Richard C. Owens, Jr.

July 1971

PROJECT MAC

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge Massachusetts 02139

ACKNOWLEDGEMENTS

I gratefully acknowledge the support of Professor David Ness and Professor Jerome Elkind, both of the Sloan School of Management. Their suggestions have contributed significantly to the design presented here. In addition, thanks are due to Mr. Robert Goldstein and Mr. Douglas Wells, as well as to the other members of the MacAIMS group, all of whom have allowed me to pick their brains endlessly during the past several months.

The work reported herein was performed at Project MAC, an M.I.T. research project sponsored by the Advanced Research Project Agency, and was supported by the Office of Naval Research under Contract N00014-69-A-0276-0002.

PRIMARY ACCESS CONTROL IN LARGE-SCALE TIME-SHARED DECISION SYSTEMS*

Abstract

The computer differs from other tools in that it presently does not provide its users with a working environment transparent to their desires; in particular, current computer systems do not support adequate mechanisms for controlled sharing of sensitive information.

Four primary dimensions of the access control problem are identified. They are: 1) the physical level at which to apply control; 2) the fineness of distinction applied to the term "access", 3) the meaning of the term "user identification", and 4) the degree of sophistication employed in automatically assigning restrictions to new data files.

Within the context of MacAIMS, the Project MAC Advanced Interactive Management System, the design of an access control system is presented which takes positions along these four dimensions appropriate for controlling access in a Management Decision System. Support is provided for constraints specified as general logical restrictions based on 1) the characteristics of the entity requesting access, 2) the content of the sensitive data item, 3) the context in which the sensitive item appears, 4) proper completion of an interactive procedure, and 5) combinations of any of these. The access levels which may be specified are based on the logical (not the physical) nature of the interaction which the user requests.

The system presented here is an interim system in that it does not solve all of the access control problems of MacAIMS. Among the unsolved problems is the problem of Truth -- in a data management system which provides a powerful set of operators, it is easy to create false information in very subtle ways. Another problem is that of conflicts of privacy. Solutions to these problems must be found before the access control scheme will be complete.

---

*This report reproduces a thesis of the same title submitted to the Alfred P. Sloan School of Management, Massachusetts Institute of Technology, in partial fulfillment of the requirements for the degree of Master of Science in Management, May 1971.

Table of Contents

FIGURES

1. INTRODUCTION

1.1. Privacy

Consider for a moment the stone-age axe. Like all the tools which man designed both earlier and later in his history, the axe serves as an amplifier of his abilities -- in this case, an amplifier of his ability to strike. Perhaps the second most important characteristic of the axe is that it is indifferent with regard to what it strikes; it may be used equally well on either a log or another man's head. The axe provides a working environment which is amoral by itself, but which possesses the ability to reflect the morals of its user.

Like the axe, the computer is a tool: it amplifies man's ability to process and disseminate information. Computers are potentially more dangerous than axes by virtue of the magnitude of their amplification, but this difference is not basic. It is the second referent which distinguishes the computer from the axe in an essential way -- in spite of the fact that considerable progress has been made in recent years, the environment provided by most present-day computer facilities lacks the

ability to adequately mirror wishes of the users. It is to a portion of this problem that the present work is addressed.

There is today no widespread public agreement regarding what the issues surrounding the privacy problem really are. In the absence of such a concensus, imposing a specific set of standards upon a computer system (if such an imposition were possible), would be almost as bad as no control at all. What is needed instead is to devise a system which is transparent to the wishes of the users -- their desires to control the flow of information should be exactly expressible within its environment.

At the outset it is useful to consider briefly a few of the more important social issues which are now in the public eye. These issues have been discussed elsewhere in greater detail; the bibliography contains a list of some of the better works.

It is probably safe to say that everyone has an intuitive idea of the meaning of the word "privacy"; it is probably also safe to say that most of those intuitions are over-simplistic. At least one definition of privacy has been suggested which emphasizes many important aspects of the problem: in Privacy and Freedom, Professor Alan Westin has said

> "Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated

> to others.... The individual's desire for privacy is never absolute, since participation in society is an equally powerful desire. Thus each individual is continually engaged in a personal adjustment process...in the face of pressures from the curiosity of others and from the processes of surveillance that every society sets in order to enforce its social norms."

Thus the privacy decision is the choice of an individual in trading off his desire to be an individual against his desire to participate in society.

From the individual's viewpoint, Dr. Westin identifies four primary (and essential) functions of privacy: it 1) provides personal autonomy, 2) gives opportunity for emotional release, 3) allows self-evaluation and introspection, and 4) permits the protected and privileged transfer of information. As the book's title indicates, these functions are very closely related to the concept of freedom. But the need for privacy goes even beyond such logical considerations -- Dr. Westin suggests that privacy may be as much a biological necessity for man as it is for other animals. (31)

The concept of norm-enforcement is inherent in the definition of "society". In order to enforce norms, a society establishes a variety of institutions which watch over individuals and monitor their behavior; thus cumulative social pressure places constraints on each citizen's privacy decision. Clearly such norm-enforcement is essential to the preservation of civilization -- we cannot, for example,

choose to drive at excessive speeds or to burn down our neighbor's house in the name of privacy. In order to accomplish such enforcement, collection and processing of considerable amounts of information about citizens is necessary. A logical framework for viewing the reasons for this collection is discussed in (32).

Perhaps the most important issue of the privacy problem, then, is the tradeoff of the individual's needs against the society's needs. Unanswered questions include the exact costs associated with that tradeoff and the identity of the party (or parties) who should make the final decision.

Another extremely important issue, and one which has received almost no attention to date, is that of conflicts of privacy. Most, if not all, data which are of interest are the joint property of at least two parties -- the person who originated the information, and the person whom the data concern. Moreover, much information may be the joint property of considerably more than two parties. In many cases the privacy rights of these parties conflict. For example, a physician would not wish a patient to see his own medical record if that record showed he was a hopeless hypochondriac. A complete solution to the privacy problem must include mechanisms for the resolution of such conflicts.

1.2. The Computer and Privacy

The essence of the solution to the privacy problem, then, is providing the ability to control the flow of information which is in some way "sensitive". Thus the problem exists in all its complexity whether or not there is a computer at some point in the transfer process. When a computer system is involved, it must be so engineered that the ability to control the transfer of information is not lost. The present work is concerned with designing mechanisms to provide that ability.

Most of the progress which has been made to date in computerized access control has been in the realm of time-shared systems. The motivation behind these advances, unfortunately, has in general been to give systems programmers the ability to test and debug programs without accidentally destroying the work of other programmers; it has been to prevent the destruction of information, not to control the transfer of that information. The ability to protect privacy in the more general sense has been only a by-product of these efforts. That the by-products have been insufficient is clear -- in the IBM System 360, for example, hardware read-protect is not provided as standard equipment; it is possible to prevent someone from writing over your information, but not to prevent him from reading it. (18)

A few systems, however, have attempted to provide more sophisticated access control procedures. In Multics the Multiplexed Information and Computing Service, (2, 5,

12, 20, 27) it is possible to specify on a user-by-user basis the permissible levels of access to each segment in the system. But even the Multics access control environment is not sufficiently general to provide convenient control of access to information at the logical level.

From a more general viewpoint, the primary decisions for an access control design scheme can be taken as choices of positions along four dimensions:

1) The physical level at which to apply access control.

2) The fineness of distinction applied to the term "access".

3) The meaning of the term "user identification".

4) The degree of sophistication employed in assigning restrictions to new data files.

One might, for example, want to associate access control information at the work space level, at the file level, at the record level, at the field level, or at the level of individual data items. Similarly, "access" might be divided in half -- i.e., "yes" or "no" -- or it might be more finely divided to distinguish between "read", "write", etc. Along the third dimension, one might use only the user's name; at a slightly more sophisticated level, one might employ both name and password; and so on. Lastly, an access control system might assign null access (or full access) to everyone for each new file. At the opposite extreme, it might be able to discover the nature of the sensitivity of data in a

new file by knowing the access characteristics of the inputs. The choice of positions along these dimensions determines the power and abilities of the access control scheme.

This work approaches the problem of access control from the general viewpoint within the framework of MacAIMS (the Project Mac Advanced Interactive Management System) on Multics. (Note: the remainder of the thesis assumes a basic familiarity with the Multics system.) MacAIMS incorporates a relational approach to data management using set theory operators for manipulation of relations. A method for controlling access to information is presented which, although heuristic in nature, is more general than any access control scheme in common use today. For each of the four dimensions of access control, a position has been chosen which appears to provide the appropriate levels of power and flexibility for our needs, or else represents the limits of our knowledge. Experience with using the system will allow us to determine for each dimension whether our level of distinction is too coarse or too fine, and adjustments can be made accordingly. It is not claimed that the controls presented here solve all the access problems of MacAIMS; indeed, several classes of problems are presented which they cannot solve. What is claimed, however, is a reasonable first cut at the solution to the access control problem in a large scale, set-theoretic data management

system, and one which should be readily extendable to handling of those problems which we already know about, and those of which we are unaware.

2. CHARACTERISTICS OF ACCESS CONTROL SYSTEMS

This section considers briefly the range of issues involved in a complete access control system for a multi-access computer facility and defines the scope of the present work.

2.1. The Range of Issues

A large number of people have presented overviews of the access control problem; the bibliography provides a list of some of those works. In considering such an overview, a useful framework to follow is the physical structure of the system.

At the outermost level, there is the problem of identifying the user at the terminal. A variety of methods for identification, including passwords, keys, cards, voice-prints, signature recognition, and so on have been proposed, but most current writers fail to notice that the output of any identification procedure is translated to a bit-string which is passed to the computer for evaluation. Any scheme in which that bit string is constant over time is doomed to failure, since the user's password may be had simply by taking the trouble to tap his data line. A more successful alternative is the procedural password, in which

the user responds at each login with the answer to a manipulation performed on a (different) string of random digits supplied by the computer. Thus the password is different at each login.

On the remaining issues the literature is more accurate. Among the problems discussed are tapping of data lines, physical security of the computer facility, residual data in core and on discs and tapes, audit trails, privacy encoding, program validation, and so on.

The technology of sharing in general is not at all understood by many, but is understood well by a few. The interested reader is referred to (7, 8, 19).

2.2. Scope of the Present Work

Of all these issues, perhaps the most critical occur at the point where information is released from a file. The design decisions along the four dimensions of access control determine the behavior of the system at this point, and if those decisions are poorly made, no amount of work in any other area will make the facility secure. It is for this reason that these decisions are termed "primary", and it is at this point alone that this thesis concentrates.

3. CURRENT ACCESS CONTROL SYSTEMS

The literature concerning what one ought to do about access control is supplemented by a much smaller literature about what has been done. The relative importance of access control issues in the past is perhaps best illustrated by the fact that the System 360 Principles of Operation devotes roughly one-half page to the issue of protection. (18)

This section surveys some access control schemes currently in use, and points out their shortcomings.

3.1. Common Systems

IBM's most significant efforts to provide access control are perhaps three: In the CP67/CMS System, files may be released to (all) other users in one of four modes: read only, read/write, read only and erase after one read, and read/write and erase after one read. However, the manual notes that all modes may not be implemented. (3). At about this same level of sophistication are the access specifications for the APL language; the owner of data may specify a password (which is the same for all users) to control access to a work space. (9) Somewhat better is the TSS/360 System, which allows specification of access

restrictions on a user-by-user basis with modes read, read/write, unlimited, and restricts. (26)

In the later versions of the PDP-10 monitor, Digital Equipment Corporation supplies a rudimentary access control system. The term "user" is separated into three categories: 1) the file owner, 2) persons on the same project as the owner, and 3) everyone else. Access to a file may be restricted for each of these three groups by read protection, write protection, and protection protection, where the last category implies the capability to change access control information. In addition, it is possible to name files such that the monitor knows they are procedures. This feature may be used to enforce "execute" access mode. (21)

M.I.T.'s CTSS supports a file system which is organized as a tree structure, and provides for sharing of files through links between branches of the tree. Access modes are essentially read, write, protected, and any combination thereof, and may be assigned at the time the link is established on a user-by-user basis. (4)

3.2. Other Systems

All of the systems listed above provide more-or-less sophisticated access control at the file level. Several other systems have been proposed (and in some cases, implemented) which protect data at a lower level.

The TERPS system (24) allows protection at the

record level within files by associating a descriptor with each file which contains a security code for the fields. However, the term "access" seems to be divided only into "yes" or "no". Restrictions may be based on terminal location, password, and security level.

Hsiao (17) has proposed a somewhat similar system, although it is not clear whether implementation is past the pilot project stage. He distinguishes between the system manager, the owner of a file, and other users, and allows protection of individual records within files.

Hoffman (16) has developed a system which is considerably more powerful than any of these. In his scheme, all accesses to a data base take place through an intermediary program (which may be different for each user) called a formulary. The owner of a file supplies these formularies. The use of a procedure instead of table lookups to control access allows his scheme to be much more clever in assigning access privileges.

3.3. Limitations

The limitations of these systems should be obvious. Several of them provide control at a physical level too high to be of much use. Protection at the file level requires that every data item in a file have exactly the same sensitivity; at a minimum this restriction requires breaking up a logical data base into a number of physical data bases. In non-military situations, where

neither users nor information are structured as strictly ordered hierarchies, such a division is frequently impossible.

Moreover, the term "access" is not finely subdivided. At best, only physical actions are distinguished (i.e., read _vs._ write); at worst the term is divided in half. Constraints cannot be based on the _logical_ actions which the user wishes to perform. One cannot specify, for example, that a user may only extract means and medians.

In several of the systems, it is not even possible to make restrictions based on the user's identity. DEC does not allow one to specify that "Smith can see anything in my files" within the context of its standard access control mechanisms, although such a specification might be implemented by special programming and the "execute only" mode.

Only Hoffman's scheme does not suffer from any of these complaints. The idea of using procedures, though simple, is very powerful. Unfortunately, the job of supplying the formularies is left to the originator of sensitive data; such a task might be quite formidable. Moreover, there seems to be no provision for protecting the user of sensitive information (who might have some sensitive data of his own) from the effects of the formularies he uses.

In addition, none of the schemes mentioned so far provide the user with any more than minimal aid along the last primary dimension of access control: he must specify one-by-one the access rights to any new files in the system.

3.4. Multics

The Multics system provides a much better (though still insufficient) access control system. Like CTSS, the Multics file system is a tree structure; it contains both directory branches, which contain only pointers to inferior branches, and non-directory branches, which are data segments, procedure segments, and so forth. Unlike CTSS, however, access control in Multics is associated with branches of the tree, not with links; a user's access rights are evaluated each time a segment is made known to him.

Permissible access modes are read, write, execute, append, and combinations thereof, and may be assigned on the basis of users and projects. The access modes imply the obvious intent for non-directory branches; for directory branches the meaning is somewhat different. (20) Access Control Lists (ACL's) associated with each branch contain the necessary access control data.

In addition to the file system structure, Multics provides a ring structure for protection which is essentially a generalization of the common "user" state/"supervisor" state idea. The mechanisms provided are

similar to those described by Dennis and Van Horne (7); some of the limitations of the rings are described in (11). For our purposes, the important characteristic of the ring mechanism is that it allows one to specify that his data may be accessed only via a program of his own choosing. Any attempt to access data from an insufficiently privileged ring must take place through a gate into the more privileged ring. It is this mechanism which allows MacAIMS to help protect data for its users.

Multics also provides some aid in assigning access to new segments. The user may specify for each directory a Common Access Control List, which contains default access assignments for segments in the directory. In addition, certain conventions are followed in assigning access to new segments created by various compilers and other routines, based on the type of segment created; thus object code segments are assigned read/execute access.

From MacAIMS's viewpoint, the most important limitation of the Multics access control scheme is that its permissible access capabilities, like those of the other systems mentioned, are not sufficiently general for powerful access control. For example, a perfectly reasonable restriction to place on the <Name, Salary> set might be

> "Smith is allowed to extract statistical data on salary distributions, but not to see individual's names and salaries."

Such a restriction is not expressible in the standard Multics scheme, though it could be implemented by special programming via the ring structure. Thus the term "read" does not discriminate finely enough. It is necessary to subdivide this capability. Like Hoffman's scheme, simply giving the user the ability to write his own access programs is not very much help.

Moreover, Multics controls access only at the file level, and therefore has the inherent limitations common to such schemes.

3.5. The ADEPT-50: automatic classification of new files

In addition, the Multics rules for assigning access to new files are insufficient. A more sophisticated scheme for handling new files is presented by Weissman (29). He views a new file as being constructed by operations which combine information from existing input files, each of which has been assigned a level of classification. The essence of his solution is to assign the new file the maximum classification level (minimum access privileges) of the inputs. In addition, subsequent operations on the file may lower the user's access privileges.

Such a scheme is perhaps adequate in the military environment; in the non-military world it is insufficient. For example, suppose that the set <Name, Group, Salary> exists, that it contains information for all people associated with Project Mac, and that the <Salary> field is

considered sensitive. Suppose further that a user is authorized to print salaries of people at Mac who are also associated with the School of Management, but is not authorized to print anyone else's salary. He would not have access to print the entire <Name, Group, Salary> set, but he should be able to derive (in our system, via set operations) a set which contains only Management personnel, and then be allowed to see the results. Thus it must be possible to obtain a set to which the access is greater than the access to the input from which it was derived.

Both the standard Multics scheme and the ADEPT-50 scheme are therefore inadequate for our purposes. Before presenting a design which corrects some of these problems, it is necessary to describe briefly the structure of MacAIMS.

4. MADAM: SET-THEORETIC RELATIONAL APPROACH TO DATA MANAGEMENT

This section provides an overview of the MacAIms Data Management System. For a more complete introduction to relational data management and MacAIMS, and a detailed description of MADAM, the interested reader is referred to (13, 14, 25).

4.1. Description of MADAM

MADAM operates under the assumption that any information which is to be stored or manipulated by the system consists of sets of Data Elements (DE's) and of sets of relations among those data elements. The DE's themselves are stored in Data Element Sets (DES's), and the relations in Relational Data Sets (RDS's). The primitives of set theory are the operators used for manipulating RDS's.

Suppose that there exist DES's DES1, DES2, DES3, ..., DES*n*. The RDS which expresses relations among elements of these DES's will contain n-tuples (tuples of order *n*), each of which has as its first entry a DE from DES1, as its second entry a DE from DES2, etc. The first tuple in the RDS will contain the names of the sets DES1, DES2, and so on, and is called the Relation Descriptor (RD). Thus, for

example, there might be two DES's:

Name
Jones
Smith
Hilphenhauser

and

Office
505
806
301

The entries "Jones", "Smith", ... "505", etc. are the DE's. The RDS which expresses the relations might then be:

Name	Office
Jones	505
Smith	806
Hilphenhauser	301

Here, <Name, Office> is the RD, and the 2-tuples of the RDS are <Jones, 505>, <Smith, 806>, and <Hilphenhauser, 301>. In addition, the columns of the RDS are referred to as fields: this RDS has two fields, and their fieldnames are "Name", and "Office". The information in this set is the fact that Jones's office is 505, that Smith's office is 806, etc. Neither the character string "Jones", nor the integer "505" alone necessarily carry any information.

At the time when a data element first enters the system, it is assigned a unique identifier which is used thereafter to reference it. This identifier is the Reference Number (RN), and in the current implementation is a 36-bit bit string which may be referenced as an integer.

RN's are not basic to the system; they are used for computational and storage efficiency. Programs called Data Element Modules (DEM's) are provided which obtain Data Elements as input, convert them to the internal storage form (the Standard Form, SF), store them in DES's, and assign their Reference Numbers. DEM's are also called to output SF's in human-readable format. Thus there might be a DEM which handles dates, one which handles names, one which handles integers, and so on.

In order to construct and manipulate RDS's, programs called Relation Strategy Modules (RSM's) are provided. There is one RSM for each storage strategy supported by the system; thus, there might be an RSM for list structures, an RSM for trees, etc. In the example above, the RDS is represented as an array, and the DE's are shown as being in the tuples of the array. This representation suffices for the logical content of an RDS and will be used throughout this paper, but it should be remembered that RDS's are not necessarily arrays, and that, in any event, in the current implementation they contain the RN's of the DE's, not the DE's themselves.

Each RSM provides the following set primitives for manipulating RDS's:

intersection
difference
projection
join
composition

product
union

These operations require two RDS's as input and produce a new RDS which contains the result of the operation. Intersection and union perform in the obvious manner; difference yields all members of the first input which are not in the second; projection removes unwanted fields or permutes fields; join forms an RDS which contains all the fields of both inputs along with those tuples which match on the inputs' common fields. Composition requires that the last field of the first input be the same as the first field of the second, and it produces an RDS which does not contain the common field, but does contain those tuples which matched. Product is the Cartesian product.

In addition, several non-set primitives are provided:

A set is ordered by use of

get_successor

For a given tuple, get_successor returns the next tuple in sequence.

For efficiency the operations

replace_tuple
find_tuple

are defined. Both are redundant in that they could be performed by combinations of set primitives. Find_tuple

searches an RDS for a given tuple; replace_tuple replaces a tuple in an RDS.

The mechanisms for creating RDS's are also furnished:

create_set
insert_tuple

The create_set operation is used to define a new RDS, and takes the necessary steps to establish its storage, its RD, and its associated DEM's and RSM. Insert_tuple inserts a given tuple into an existing RDS. Thus these two operators provide for original data entry into the system.

For details of the functions of these operations, the reader is referred to (25).

In addition to the Reference Numbers assigned by DEM's, there are two special RN's: the "wild card", represented by "*", which is considered to match any RN; and the null RN, represented by "----", which indicates a null DE. Thus the tuple

<*, *>

matches any of the three tuples in the above example. The tuple

<Jones, *>

matches <Jones, 505> above, and would also match any other Jones's in the RDS. The tuple

<Jones, ---->

matches none of the above tuples, but would find any Jones's

who had no office in the RDS.

4.2. Implications for Access Control

The philosophy and the design of MADAM have several implications for the design of its access control mechanisms.

The set operations provided by MADAM are powerful in that they allow concise specifications of large amounts of work. The access control scheme must provide similar power through conciseness of expression, or it will significantly detract from MADAM's usefulness. In a company employing 50,000 workers, for example, it might be entirely unacceptable to specify access to a particular RDS by a list of names: an access list of 3,000 people would be entirely unmanageable.

Moreover, the fact that data bases might be very large implies a great deal of sharing of physical data between users who have widely varying characteristics and information needs. A rather fine distinction of the logical types of access to a data base is needed, so that precise levels of control may be applied in a variety of circumstances. The term "read access" is too coarse a distinction to be useful.

Last, the use of set operations to obtain information from the data bases implies a large number of temporary RDS's containing either intermediate results or special relations which are useful for some application. A

single interaction with the main data base might cause the creation of several new RDS's. Therefore simple-minded schemes for assigning access privileges to new files are insufficient; a fixed default would not characterize properly more than a few of the new sets. In addition, placing the burden of assigning access privileges for new sets on the originator of sensitive information would discourage almost anyone from storing such information in the system. Even Weissman's ADEPT-50 scheme is far too simple; MacAIMS access control must provide the originator of sensitive information with much more significant aid in assigning access rights to new RDS's.

The access control schemes presented in Section 3 have chosen positions along the four dimensions of access control which are too naive for the purposes of MacAIMS. We turn now to the design of a scheme which exhibits much more power and generality.

5. THE MACAIMS INTERIM ACCESS CONTROL SYSTEM

This section describes the initial access control mechanisms for MADAM. The system outlined here is an "interim" system in the same sense that MacAIMS itself is in an interim state: we do not at this time fully understand all the problems (let alone their solutions) associated with access control. Therefore, a first approximation access control system is provided which will yield a simplified solution to the general problem as well as a base from which to experiment with more powerful algorithms for control. Moreover, the system described here is "primary" in that it is limited to the issue of interaction of an identified user with system information.

The interim access control system is considered first at the level of basic assumptions, then at the user's level, and finally at the system design level.

5.1. Basic Assumptions

In order to delimit a common ground for discussing the primary access control problem in the context of MacAIMS, the following basic assumptions are enumerated.

5.1.1. Definition of Access Control

For the purposes of this study, the following definition of access control is sufficient:

> Access Control is defined as the restriction of the use and dissemination of sensitive information to authorized entities.

This definition merits further clarification:

Entity is defined to be the party requesting access. Roughly speaking, this is the user of the system, but the intuitive meaning of "user" is insufficient for our purposes. In MacAIMS the requesting entity corresponds more nearly to the concept of a "computation", as expressed by Dennis and VanHorn (7), and is precisely defined by a list of characteristics associated with the state of the system at the time access is requested. The list includes as a minimum the standard Multics identification characteristics:

1) User_id (name)

2) Project_id (project)

3) Instance tag

and may be arbitrarily extended to include any other information about which MacAIMS has knowledge or can be made to obtain knowledge. Such extensions logically include

characteristics like:

4) Terminal_id (the terminal identifier of the process requesting access. This characteristic is fairly commonly used in military security systems, and will be provided in MacAIMS for the sake of generality, but it is not considered to be especially secure for the purposes of access control, since terminal id's are relatively easy to change.)

5) Program_id (the procedure requesting access.)

6) Time of day

7) Day of the week

etc.

Information in MADAM is considered to reside only in Relational Data Sets. It is therefore those sets which are to be protected.

Information is sensitive as long as there are conditional restrictions on its use or dissemination. Such restrictions may be placed on information either by its originator or by any other party to whom he grants that capability. If all fields of an RDS have the maximum possible access rights, then the information is not sensitive, and the access control system need no longer be concerned with it. The possible levels of sensitivity allowed by the interim access control system are described in Section 5.2.1. below; the procedures and data which describe the conditions of sensitivity may be completely arbitrary.

Use of information is defined by an operator which

an entity requests be applied to that information. In the interim system, possible operators are restricted to those performed by Relational Strategy Modules, i.e., the set theory operators and the non-set primitives.

Dissemination of information is distinct from use by virtue of being a "final result" from the system's viewpoint. Dissemination implies the passing of information beyond the boundary of the access control system's ability to restrict.

An entity is authorized to request an interaction (a use, a dissemination, a write, an append, etc.) with existing information if his characteristics and the operation's nature and the result of the request satisfy the sensitivity constraints which have been placed on the information. It will be demonstrated that all of these data must be considered in determining permissible access levels. In the interim system, the levels of authority granted (as well as the levels of sensitivity) are constrained to fall into a fixed set of access capabilities described below.

Finally, restriction is the act of constraining information accesses to those authorized in the above sense.

5.1.2. System Characteristics

The following general (and rather obvious) characteristics are desired in the interim access control system.

In order to be immediately useful, the interim

system must be general in the same ways that MADAM itself is general. It must not place any restrictions on the nature of the actual information which it protects. Second, and somewhat more difficult, it must allow considerable flexibility in the ways the owner of sensitive information may express the conditions of sensitivity.

The system must be extensible since it will not be a complete solution to the problems of access control. It must eventually be extended to include solutions to the problems which it does not now solve, as well as solutions to problems which we have not forseen. Since the decisions along each of the four dimensions of access control discussed in Section 1 are arbitrary, it must be relatively easy to change those decisions as experience dictates. More important, individual users must be able to extend the access control mechanisms in arbitrary ways to handle their own special requirements for control.

There must, however, be some limit to these arbitrary extensions, lest the situation arise where every user has his own access control system which is so specialized to his needs that it cannot communicate with any other user's system. The interim system provides this common framework for communication by fixing the set of assignable sensitivity (and authority) levels. The set, however, is fixed by choice, not by any inherent limitation of MacAIMS.

The interim system must be compatible with the rest of MADAM in the sense that it utilizes both the data storage mechanisms and the operational mechanisms provided by MADAM to do its work. It is this compatibility which provides the interim system with the ability to meet the first two design goals. In addition, the access control system gains great power because the large body of tools available in MADAM become available for manipulation of access information. Moreover, compatibility means that as MADAM becomes more powerful, the access control system will gracefully evolve in a similar fashion.

Lastly, the cost of controlling access (in terms of both user effort and computation time) should vary directly with the complexity of the sensitivity description of the data; detailed and sophisticated descriptions should be expected to cost more than simple ones. Any user who wishes to interact with someone else's sensitive data should expect to pay in proportion to that sensitivity. Moreover, a user who does not desire protection should expect his costs to be somewhat lower.

5.1.3. Design Assumptions

Other assumptions upon which the following description is based are those implied by Section 2.2. above: namely, the entire range of access control problems that physically exist between the user station and the CPU are not considered. All information concerning the user's

identity and the other characteristics which identify the requesting entity is taken to be accurate; verification of that data is a separate problem.

5.2. Design Specifications

This section outlines, from the user's viewpoint, the interim access control system's appearance and behavior.

At the outset it should be noted that it is not feasible to store access control information on a tuple by tuple basis in MacAIMS, even though the tuple is the basic unit of information. Such schemes would require more storage and computation than reason permits, and would make the user's task in specifying access control information impossibly large. This does not imply that one cannot specify constraints that affect single tuples or small groups of tuples within an RDS. Even in these cases, however, the access control information is not physically attached to the individual tuples.

It is also unreasonable to provide access control on an RDS by RDS basis, since this level of control is too superficial to allow sufficiently general specifications of sensitivity. Suppose, for example, that the RDS <Name, Salary, Religious-Affiliation> were created. Both salary and religious affiliation would probably be sensitive, but there is no reason to believe that the nature of the sensitivities would be the same. Access control information is therefore retained on a field by field basis within Relational Data Sets.

5.2.1. Access Capabilities

A user may request the transfer of information

from an RDS to himself for two (and only two) logical reasons -- he may wish to use the information as data in calculating other information which he needs (i.e., he may wish to use it in the sense of Section 5.1.1.) or he may wish to view the information itself as a final result (he is requesting dissemination). However, the concept of "use" is still too general for access control. It is necessary to distinguish still further, and for the interim system, "use" has been split into two categories: manipulative and statistical use. Manipulation means the application of the set theory operators provided by MADAM; statistical use is the extraction of means, medians, and so forth. This subdivision is strictly hierarchical, so that the concept of "increasing access" will have meaning.

Thus there are the following four levels of access for information which flows from an RDS to a user. In increasing order (by the amount of information which they release) they are:

N -- null; no access is permitted. (This is the default access unless the originator of the information specifies otherwise.)

M -- manipulate; the user is permitted only to apply set theory operators to the information. He may or may not be allowed to see the results of that manipulation, depending upon the access restrictions of the data.

S -- statistical; the system will permit dissemination of statistical data about the information only.

P -- print; the system will disseminate the information.

Each of the lower access capabilities is considered to be a proper subset of the next higher capability. Thus S access includes the right to manipulate the information.

It is implicit in the above statements that the programs which the user invokes in order to apply set theory operators or statistical operations are either standard MacAIMS programs or other programs whose performance has been "guaranteed" to the access system's satifaction. In order to apply arbitrary programs to protected data, the user must possess the capability to have the information disseminated (P access).

The division of "use" into N, S, and M is clearly somewhat arbitrary. It might, for example, be desirable to provide even finer levels of distinction by further dividing S into the capability for extracting the mean, the capability for extracting the standard deviation, and so forth. It would seem, however, that the divisions expressed here are at about the appropriate level of distinction for most management data in the context of MacAIMS; only experience with using the system will allow us to judge more surely.

Similarly, there is a hierarchy of capabilities for the flow of information from an entity to an RDS. These capabilities are essentially those provided by Multics, except that they are viewed as a strictly ordered hierarchy

of privilege rather than as discrete but equal divisions. In increasing order, they are:

N -- null; no access is permitted. (This is the system default.)

A -- append; the user may append relations to the "end" of the RDS (where "end" is the logical, not the physical, end), but he may not change information which is there already.

W -- write; the user may change existing information or add information to the RDS.

C -- change access; the user may manipulate the access control data which is associated with the fields of the RDS.

There are in addition two capabilities associated with each RDS with which the user need not normally concern himself, since they are assigned and maintained by the system. They are:

O -- ownership; the owner of an RDS is the user who caused its creation. He may therefore delete it. (Note that this does not imply the C capability or any other; an acquired data set belongs to whoever caused its creation, but has access capabilities determined by the sets which were used to derive it. Thus a user may derive a set to which he has no access; however, since he owns it, he can get rid of it.)

Ac -- acquired; an indicator that the RDS contains information which was not originated by the owner. A data set is acquired unless 1) it was created through the use of the primitive operations "create_set" and "insert_tuple" or 2) is made from RDS's which were not acquired and which are owned by the user in question. The acquired flag being off implies the C capability.

Thus in the MacAIMS access control system, there are three distinct concepts which in more conventional systems are lumped together under the term "owner". First, there is ownership, which implies responsibility for

creation of an RDS and the right to delete it. Second, there is the concept of the originator of information (acquired flag off), which implies original entry of all data in the RDS and therefore the rights of ownership and access changing. Last, there is the ability to change the access control specifications, which falls to the originator and to whomever else he specifies. The interim system does not, therefore, provide mechanisms for resolution of conflicts of privacy. Such issues must be solved through extensions of the current scheme.

5.2.2. Description of Access Rights

It is clear that in the non-military world there are many different ways in which one might wish to specify sensitivity constraints on information which is to be protected. In the simplest (and probably the most common) case, one might list all the people who can (or cannot) access the information, along with what type of access they have. Such a scheme is supported by Multics. There are many cases, however, where such a listing is either inconvenient or impossible. It is therefore desirable to provide more powerful mechanisms for specifying the conditions under which one's information should be released. Such statements as "Anyone may see information about himself" or "Anyone above the level of plant manager may see the personnel data" should be concisely expressible. It is not possible, obviously, to delimit every way in which such

constraints might be stated, nor would one support them all if it were; the access system must be simple enough to be comprehended if it is to be used. Support is therefore provided for a number of ways to express access control restrictions which should be general enough for most purposes. In addition, we provide a mechanism by which the originator of information can specify procedures of his own choosing for access control.

The originator of information and those users to whom he has granted the C capability may specify, for each level of access capability, conditions for its release by the following tests:

1) Test on the list of characteristics of the requesting entity: He may specify tests to be made on any of the characteristics of the requesting entity listed in Section 5.1 above and included in the state set of the system (see "The Access Control Data Base", below). Such tests logically take the form

"Jones has P access"

"if <user_id> = Smith and <terminal_id> = a64 and <day> = Friday, then access = M"

and so forth. They may also be "negative" specifications, such as

"if <user_id> ¬= Smith then access = P"

With this test, as with all others, a default access may be specified which is logically equivalent to an "else" clause

and which will take effect if all tests fail. If no default is given, the system default is null; the system is basically closed rather than basically open.

2) Test on the content of the sensitive domain: He may specify access restrictions based on the content of the field in question. For example, if salary is a protected field, a constraint might take the form:

"if <salary> = 100000, then access = N"

The interim system supports only statements which are expressible in terms of the existing MADAM operators, i.e., only matching is allowed. This implementation restriction probably makes tests on content alone less useful; one would generally want to make statements such as "if salary is less than 10000 then...". This simple "less than" constraint would be easy to handle, but it is a special case of the problem of subsetting based on general logical restrictions (which is under study at MacAIMS) and will therefore not be solved separately. For the present such constraints must be handled by special programming. Since the access control system uses MADAM constructs exclusively, however, such capability will be available to the access system when it is available to the rest of MADAM.

3) Test on the context in which the sensitive domain appears: He may specify constraints based on the context in which the field in question may appear. These context restrictions may be based only on the fieldname of

the other field, i.e.

"salaries may not be printed in conjunction with names"

or they may be based both upon the fieldname and its content:

"salary may not be released if it is seen in conjuction with name and name = George"

The first of these will be called a simple context restriction, the latter, complex. This distinction is necessary for computational purposes, as described in Section 5.4, but is not basic to the logic of access control.

4) Proper completion of an interactive procedure: He may specify that users supply a password (perhaps a procedural password as described in Section 2) at the console in order to obtain access. The system interactive access routine will provide for one challenge and response; any more fancy interaction will require special purpose programming.

5) Any combination of methods 1 - 4: He may specify combinations of the above; for example,

"Jones may see salaries if salary appears in conjunction with the field <group>, and the contents of <group> are the same as his projectid"
(requesting entity characteristics + complex context.)

It is felt that this level of support of access description is general enough to handle the majority of MacAIMS access problems. At the same time, by allowing the

user to specify access control by stages, considering first the system state, then the content of the field, etc., the system should be reasonably easy to use. Moreover, the simpler the restrictions, the simpler they are to specify.

5.2.3 Acquired Relation Sets

A relation set is "acquired" only if it is obtained by combining existing relation sets that contain information which was not originated by the owner. Thus (obviously) all acquired relation sets are new sets, but (not so obviously) all new relation sets which are the output of a set operation are not acquired sets. If a user combines sets which contain only information which he originated, the new set is conceptually identical to a set formed by using "create_set" and "insert_tuple", and it is therefore not acquired. In MacAIMS as it now stands, sets are acquired only through the standard set operations provided by the RSM's; presumably it will one day be possible to derive sets by other means.

In any event, it is the system's responsibility to maintain appropriate access control to information in acquired sets. Not only are access privileges automatically assigned to the acquirer, but also an Access Control List is assigned to the acquired set which will maintain proper control even if the set is passed on to other users.

5.3. System Overview

This section provides an overview of the design of the Interim Access Control System. The access control data base and the MacAIMS Access Restriction Module (MARM) which is responsible for maintaining that base and for using it to control access are described here; subsequent sections discuss in detail the representation of access information and the rules for computation of access rights to acquired relation sets, as well as methods for defining access rules in non-standard ways.

MARM is the overseer program for access control, and is invoked at each attempted access to sensitive information. The other major program modules which assist it are the access handlers, each of which evaluates its data and returns a boolean value indicating whether its constraints are satisfied, and the RSM for evaluated functions, which maintains RDS's whose tuples require function evaluation. In addition, the ACL_builder provides aid in constructing Access Control Lists.

5.3.1. The Access Control Data Base

The following pages detail the access control data base as it logically exists within MADAM. In some cases, the physical data base may differ slightly from the logical data base for reasons of programming expediency, but such differences are not relevant here.

It is implicit in the discussions of Sections 5.1

and 5.2 that there are two general classes of information which are necessary for access control. First, there is information which pertains to the characteristics list of the entity requesting access; second, there is information which details the nature of the sensitivity of information to which access is requested. All information regarding the characteristics of the requesting entity resides in the "state set" of the system, which has the form of a MADAM RDS, and all information regarding the sensitivity of the fields of RDS's resides in the "Known RDS Table" and its associated Access Control Lists.

5.3.1.1. Requesting Entity Information: the state set

The state set is logically a single tuple of degree n where n is the number of characteristics associated with the requesting entity. It is an RDS with only one entry besides the relation descriptor, and has the form:

<User_id, Project_id, Instance_tag, Terminal_id, ..., Program_id, Time, ...>

"..." represents any other information which the system chooses to support for access control purposes; such choices should be based on user demand. <Program_id> includes both the program name and entry name of the program active when the attempted access occurred.

The state set is accessed via the RSM_evaluated_functions_, and is a special case of the larger class of RDS's which that RSM supports. An RDS

Associated with each data base in the system is a (logical) RDS called the Known RDS Table (KRDST). The KRDST is made up of two physical RDS's. The first, called the KRDST, has the form

<RDS_name, ..., Ownership, Acquired_flag, Restriction_flag, False_content_flag, False_context_flag, Time_of_reference, ACL_control_rds>

where ACL_control_rds points to the second part of the KRDST, which is called the ACL_control RDS, and has the form

<Fieldnumber, Access, ACL_rds>

The user's data base begins with an empty KRDST; i.e., no RDS's are known to him. At the first requested reference to an RDS, an entry in the KRDST is made, and the user's current access privileges to each of the n fields of the RDS are evaluated and stored in the <access> indicators of the ACL_control. When a set operator is performed which results in the creation of a new relation set, the input sets are checked against the KRDST and entered if necessary, and the new set is appended in a similar fashion.

The fields of the KRDST and their uses are as follows:

1) RDS_name (the name of the Relation Data Set)

2) ... (other information which the system needs to keep regarding the RDS, but which is not directly related to access control; this might include pointers to the physical location of the segment, etc.)

3) Ownership (the indicator of whether the RDS is owned by this user, as described in Section 5.2.1.)

4) Acquired_flag (the indicator of whether the RDS is acquired, as described in Section 5.2.1.)

5) Restriction_flag (on if there are content or context restrictions on any field of the RDS)

6) False_content_flag, False_context_flag (for a acquired set, an indicator of whether operations have been performed in the derivation history of the RDS which make it impossible for the access control system to guarantee that the content/context is "True." This problem is discussed in detail in Section 5.5. For a set which is not acquired, the flags are off.)

7) Time_of_reference (the time of the last reference to the RDS)

8) Fieldnumber, Access, ACL_rds (These represent the information necessary to compute access to field <fieldnumber> of the RDS in question. <Access> is the user's currently computed access privilege level, and is computed at the time the KRDST entry is made from the information contained in the ACL's which are pointed to by the ACL_rds's. (In the case of an acquired data set, it will be computed from the ACL's of the input RDS's). At each reference to the RDS, the Time_of_reference is checked against the time of last modification of the ACL's pointed to by the ACL_rds's, and if they have been changed, access is recomputed before any information is released. More than one ACL may be associated with a particular field (this occurs primarily with acquired relation sets). Multiple ACL's are represented by multiple entries in the ACL_control; in these cases, all of the ACL's for field i are evaluated, and the minimum returned access is assigned. This is equivalent to logical "and": if ACLi1 specifies P access under conditions X, and ACLi2 specifies P access under conditions Y, then P will be granted to field i only if X "and" Y is true. The methods for computing the access level and for determining the ACL of an acquired data set from the ACL's of its inputs are discussed below.)

There is one KRDST per data base in the system. Note that some data bases may be temporary (per Multics process), and as such will be destroyed when the user logs out unless explicitly saved.

The ACL's (which may be the same for many RDS's) are themselves RDS's with the following format:

<Access_level, And_flag, Access_handler, Data_type, Data_rds>

The meaning of the fields are as follows:

1) Access_level, Access_handler, Data_rds (Access_handler is the name of a program which may be either system supplied or user supplied and which operates on the data structure pointed to by Data_rds in conjunction with such system state data as necessary to return one of two values -- either "The conditionals specified are satisfied", (True) or "The conditionals specified are not satisfied". (False) If the response is True, then the permissible level of access is that implied by Access_level.)

2) And_flag (used to link together entries of the ACL as being part of the same conditional specification. If two (or more) tuples are linked together by the And_flag, both access handlers must return true in order for Access_level to be granted. This is a logical "and", and tuples linked in this way are called And groups.)

3) Data_type (specifies the nature of the data pointed to by Data_rds.)

Entries in each ACL are in decreasing order by Access_level.

At this point it is possible to draw the logical structure of the access control data base, as shown in Figure 1).

State Set RDS (System Wide)

User_id	Project	Tag	Terminal	Prog	...	Time

Password Set RDS (System Wide)

User_id	Password

==========

Known RDS Table (KRDST)
(One per Data Base)

RDS name	...	Owner	Acq Flag	Rest Flag	False Content	False Context	Time	ACL_cntrl RDS
RDS1								

ACL_control RDS
(Per sensitive RDS)

Fieldnum	Current Access	ACL_rds
1		

ACL
(Per field per RDS)
(May be shared)

Access Level	And Flag	Access Handler	Data Type	Data_rds
P		Prog1		

The Access Control Data Base

5.3.2. MARM: The MacAIMS Access Restriction Module

It is MARM's responsibility to oversee the access control data base, to keep it up to date by assigning access appropriately to RDS's in the KRDST, and to perform the function of restricting access to authorized entities.

MARM "evaluates the access" for a particular field i of an RDS by the following procedure:

1) For each tuple in the ACL_control whose <Fieldnumber> entry is i, "evaluate the associated ACL".

2) Assign the minimum access level returned by these evaluations.

The procedure for "evaluating an ACL" is:

1) For each tuple in the first And_group (there may be only one tuple), call the access handler; if all handlers in the group return True, the access level is <Access>.

2) If the first And_group fails, try the second, etc.

3) If all And_groups fail, assign the originator-supplied default access.

4) If the originator supplied no default, assign N.

Thus, evaluating access to a field of an RDS is essentially a process of executing a list of programs (the access handlers) until one of those programs decides that the situation satisfies its conditionals. This process is

made somewhat more complicated by the need to "and" these programs in order to allow mixed tests, and by the need to "and" the ACL's in order to preserve access rights through acquired data sets. Whole ACL's may be "and"ed on the ACL_control RDS by use of the <fieldnumber>, and individual tuples within an ACL may be "and"ed by use of the And_flag.

In any other case, separate tuples in the ACL_control or an ACL imply logical "or". For example, the ACL:

Access Level	And Flag	Access Handler	DT	DRefnum
P	1	Prog1	-	Data1
P	2	Prog2	-	Data2

is equivalent to the logical expression:

"if <Prog1's conditionals> or <Prog2's conditionals> then access = P"

whereas the ACL

Access Level	And Flag	Access Handler	DT	DRefnum
P	1	Prog1	-	Data1
P	1	Prog2	-	Data2

is equivalent to the logical expression:

"if <Prog1's conditionals> "and" <Prog2's conditionals> then access = P"

At each invocation MARM performs the following steps:

1) It checks the appropriate KRDST entries for the input RDS or RDS's; if they are not already known to the user, it makes them known, evaluates the access to each of their fields, and makes appropriate entries in the KRDST.

2) It checks whether the operation is a non-set primitive. If so, it decides whether an access violation is implied, either does or does not permit the operation, updates the KRDST, and returns.

3) If the operation is a set primitive, then it will result in a new (perhaps acquired) RDS. MARM makes an entry in the KRDST for the new set, and computes its ACL_control. The ACL_control entry for each field is the union of the ACL_controls of the corresponding fields of the input RDS's, with the <fieldnumber> field appropriately set. Thus the new ACL_control will cause MARM to perform the logical "and" of the input ACL's. If both input fields had the same ACL_control, then the new ACL_control (and hence the ACL's themselves) is the same as that of the inputs. Notice that this algorithm gives new RDS's only a pointer to the actual access control data from the input sets. Any subsequent changes in that data will therefore be reflected in derived sets; of course, this characteristic may be overridden by supplying copies of the data instead of pointers.

4) MARM now assigns the current access privileges. If the set operation is neither intersection nor difference, the access level for each field is the minimum of the access levels of the corresponding fields of the inputs. For intersection and difference, the new ACL_control for each field is evaluated in the above manner, and the resulting

access rights are assigned. It is thus possible to obtain a subset (by intersection or difference) to which one has access priyileges higher than his privileges to the original set.

5) In either case, MARM decides whether an access violation has occurred, takes appropriate action, updates the KRDST, and returns. In the interim system, the response to an access violation is a denial of the request. It would be simple to make this action either more severe (ring alarms, log the user out) or less severe (release whatever information he was really allowed to see), as the situation might warrant.

5.4. The Standard Description of Sensitive Data

This section addresses the problem of describing the logical constraints placed upon sensitive information. The requirement for compatibility with the existing MADAM system implies 1) that MADAM constructs be used for storing the access control data, and 2) that MADAM operators be used to operate on it. In fact such constraints may not be concisely expressed in MADAM as it now stands, but the compatibility requirement overshadows such considerations.

These descriptions of sensitivity are the RDS's pointed to by the <data_rds> field of the ACL when the <access handler> is a system-supplied program.

The essence of the solution which the interim access control system adopts is the construction of one or more filtering relations. A +Filter RDS represents information that may be accessed; a -Filter RDS represents information that may not be accessed. Then, given an RDS to evaluate (the Requested RDS), a system access handler logically intersects it with the filter RDS; the result of this intersection is the Accessible RDS. For a +Filter, the Requested RDS passes if the Accessible RDS contains exactly the same information as the Requested RDS. Since the +Filter represents information which can be accessed, all the information in the Requested RDS must get through or else the Requested RDS contains inaccessible data. For a -Filter, the Accessible RDS must be empty in order for the

request to be valid; any information which gets through the filter is data to which the user does not have access.

In the interim access control system, the standard access handlers use the set primitive join instead of intersection, because join ignores fields which the input RDS's do not have in common. Thus filters may be constructed without regard for fields which are irrelevant for access control. This fact facilitates sharing of both filters and entire ACL's between RDS's.

The logical operators used in expressing constraints are "and", "or", and "not". We have already seen that adding tuples to an RDS may serve as a logical "or"; this equivalence also holds within filter relations. In addition, the <fieldnumber> field of the ACL_control and the And_flag of the ACL, when coupled with MARM's evaluation rules, provide two methods for specifying logical "and". Adding fields to a filter is yet another way of specifying "and": examples below demonstrate that "and" is implied between fields of an RDS on a tuple-by-tuple basis. (Thus, a filter containing only "or" constraints has order one.) The "not" operator is represented by a -Filter.

The precise rules by which a requested RDS may pass a filter are:

> +Filter: 1) The join must match all fields of the filter (=> order of the Accessible RDS = order of the Requested RDS.) and 2) the number of tuples in the Accessible RDS = the number of tuples in the Requested RDS.

-Filter: For simple context filters, the join must fail (=> no fields match). For all other -Filters, 1) The join must match all fields of the filter, and 2) the number of tuples in the Accessible RDS = 0.

The <Data_type> field of the ACL indicates which type of filter is pointed to by <Data_rds>. Use of these constructs and the evaluation process are best made clear by examples.

5.4.1. State Set Constraints Alone

Filters for state set constraints alone are the simplest because the structure of the State Set RDS is fixed by convention. Thus the context of any state set entry is fixed and the join operation will always match on all fields. The "*" convention may therefore be conveniently used in adding constraints.

Suppose, for example, that the initial constraint on a field is

"If <User_id> = Smith | <User_id> = Jones | <User_id> = Hilphenhauser, then access = P"

The resulting +Filter would simply be:

User id
Smith
Jones
Hilphenhauser

The result of joining this RDS with the State Set RDS is an RDS which contains one tuple if the current user is Smith or Jones or Hilphenhauser, and which contains 0 tuples otherwise. Since the State Set is one tuple long, it passes

the filter only under the correct conditions.

Now suppose we wish to add the (completely independent) constraint

"if <terminal_id> = a64 then access = P"

This may be accomplished by adding a field to the existing +Filter:

User_id	Terminal_id
Smith	*
Jones	*
Hilphenhauser	*
*	a64

The first tuple of the filter now precisely expresses the constraint

"if <user_id> = Smith & <terminal_id> = any terminal, then access = P"

but this is no different than the original constraint, since all users must be logged in at some terminal. The last tuple in the filter matches anyone logged in on terminal a64, as desired. Thus constraints may be added without regard for previous tuples, as long as "*" entries are appropriately added.

A -Filter for state set constraints is constructed in exactly the same fashion. The join must yield an empty Accessible RDS to satisfy the constraints.

5.4.2. Content Constraints Alone

Constraints which involve only the content of the sensitive field are also easy to construct. Such

constraints do not involve logical "and"s, and the filters are therefore of order 1. Suppose, for example, that <religion> is the sensitive field, and the constraint is of the form

> "if <religion> = Methodist | <religion> = Catholic, then access = M"

The +Filter is then

Religion
Methodist
Catholic

The access handler joins this filter with the Requested RDS, and the rules for passing are the same as above; new constraints are formed by adding tuples. A -Filter would work similarly.

5.4.3. Context Constraints Alone

Context constraints alone are more difficult than either state set or content constraints because it cannot be guaranteed in advance that a filter applied to an arbitrary RDS will match on all fields. This fact means that the "*" cannot be used in the same fashion as in the above examples.

Suppose that <salary> is the sensitive field, and that context constraints are to be specified.

Simple context constraints are easily representable. If the constraint is of the form

> "if <<fieldname>> = <social security number> then access = N"

then the filter is

social security no
*

If the simple context constraint is a -Filter, the form is the same, but the rule is that the field must not match.

Now suppose that the user wishes to express two constraints on the <salary> field, namely

"if <soc. sec. no.> = 100-10-1000 then access = P"

and (independent of that)

"if <name> = <User_id> then access = P"

One's first inclination might be to create the filter

name	soc. sec. no.
<user_id>	*
*	100-10-1000

This filter, however, is inappropriate. By the rule that all fields must match, the first tuple specifies that both name and social security number must appear, and name = <user_id>; the second tuple is interpreted similarly. These are not the constraints specified. Moreover, adopting a rule that only one (or more) fields must match would not solve the problem -- this filter would then pass any RDS of the form <soc. sec. no., salary>, due to the "*" in the first tuple, and any RDS of the form <name, salary> due to the "*" in the second tuple.

Thus specifying "or" constraints on separate fields of the context requires separate filters. Filters of more than one dimension have "and"s between the fields.

Only in the case of the state set may these "and"s be ignored.

5.4.4. Interactive Constraints Alone

The interactive access handler is identical to the state set access handler, except that it uses an RDS of the form

<User_id, Password>

instead of the state set RDS. This RDS, called the Password Set, is also maintained by RSM_evaluated_functions_, and it contains the flagged RN's for <user_id>, and <password>; therefore any reference to it causes calls to functions which obtain the user's name and a password from the console.

Pure interactive constraints are of the form

"if <user_id> = Smith & <password> = openup, then access = P^{π}"

and are represented in a +Filter RDS as

User id	Password
Smith	openup

Use of the interactive access handler is analogous to use of the state set access handler.

5.4.5. Combinations of Constraints

Combination of the various types of constraints is an extension of the filtering concepts presented above. The physical representation of compound constraints is

determined by the nature of the particular expression. In some cases more than one storage scheme for a given compound constraint is possible because of the three different representations of "and". In these instances the normal action should be to fit as many constraints as possible into one filter, in order to minimize both storage for ACL's and execution time for MARM. On the other hand, if a particular filter is sufficiently general that it may be shared between a number of different RDS's, one would not insert an "and" constraint if such an addition would render the filter unsharable.

The above examples demonstrate that context constraints may be "and"ed into the same filter if the meaning of the constraint is

(<context constraint1>) & (<context constraint2>).

In the same fashion, a content constraint may be "and"ed into a context filter. Thus if <salary> is the sensitive field, the constraint

"if (<salary> = 10000) & (<group> = EE) then access = p"

may be combined into one filter:

Salary	Group
10000	EE

One might now "or" compound constraints into this filter (by adding tuples) only if they are of exactly the same form as the first; the filter cannot be used for <salary>

constraints alone, nor for <group> constraints alone. Note also that a +Filter and a -Filter could not be combined in this manner.

In a similar fashion, certain state word constraints may be added to a content/context filter, given that the RSM_evaluated_functions_ has been designated as the RSM for that filter. For example, if an existing filter for the <salary> field looked like

Name	Group
Smith	EE

then we might specify that (in addition to passing <Smith, EE>'s salary) anyone from EE could see his own salary by adding the tuple:

<<user_id>, EE>

where <user_id> is the flagged RN for the current user's name.

This exhausts the methods by which different types of constraints may be combined within a single filter. A pure content constraint could not be mixed in a filter with a pure state set constraint because the first requires a join with the Requested RDS, and the second requires a join with the state set. This is precisely the reason for the And_group construct of the ACL: for example, a compound constraint of the form

if (<pure content constraint>) & (<pure state word constraint>) then..."

is represented by separate filters whose corresponding tuples in the ACL are connected via the And_group field. Any of the types of constraints may be "and"ed in an analogous fashion. The And_group may also be used to combine constraints which are representable in one filter, but this causes MARM more work.

The highest level of logical "and"ing is the <fieldnumber> of the ACL_control RDS. This construct is provided primarily for MARM's convenience in computing ACL's for acquired RDS's, but could also be utilized by an originator of information who wished to share existing ACL's.

In the interim system, the structure of the access control data base at and below the ACL_control RDS level will be largely under the control of the user who is constructing it. It is clearly possible for him to represent his constraints in an extremely inefficient manner. To the extent that he wishes to specify simple restrictions, however, the complexity of the access control data can be minimized.

5.5. Standard Computation of Access Rights to Acquired Relation Sets

It is claimed in Section 4 that the structure and philosophy of MADAM are such that the system must provide substantial aid in assigning access control data to derived sets. In fact, it should be possible to derive a set to which one has greater access privileges than he did to the input sets.

The concept of filters and the MARM mechanisms presented in Section 5.4 provide the ability to determine whether or not a given RDS (and the current user's characteristics) satisfy the logical constraints specified by the ACL's. However, it is easy to demonstrate that the constructs presented there do not solve the problem of access control.

5.5.1. What is Not Sufficient: The Problem of Truth

The problem of Truth may be demonstrated by a slight variation of the example of Section 3. Suppose that there exist two system data sets:

<Name, Salary>

and

<Name, Group>

both of which contain information about all people associated with Project Mac. Let the restriction on the <salary> field be the same as before, namely, that the current user is allowed P access only to the salaries of

people also in Management. His access level to the two existing sets is M.

There are (at least) two distinct ways that the user might derive a set containing only the salaries of Management people. He could create (by create_set and insert_tuple) either the set

Name	Group	Salary	
*	Management	*	(1)

or the set

Name	Group	
*	Management	(2)

Using set (1), he could acquire the desired result by the sequence of operations

(<Name, Group> join <Name, Salary>) intersect (1)

The join yields an intermediate set <Name, Group, Salary> which contains data on everyone from the original sets, so the user should not have P access to it; the intersection yields the set of Management personnel only.

Alternatively, the user might perform the sequence

(<Name, Group> intersect (2)) join <Name, Salary>

In this case, the intersection gives the intermediate set <Name, Group> of all Management people, and the join appends their salaries by matching on the <Name> field.

Either of these actions produces a set to which the user should have P access. The +Filter which represents the sensitivity constraint on <salary> might look like

Group
<group>

where <group> is the flagged RN for the current user's group. If our user is indeed in Management, then this filter will pass the final acquired set.

Suppose however, that the user performed this operation:

<Name, Salary> join (2)

The join would match on the <Name> field, which in set (2) contains a "*"; the result would be the set

<Name, Salary, Group> (3)

which lists all the people from the original system data set, but in which the <group> field contains "Management" in every tuple. The filter shown above would pass this set as legitimate.

This is the problem of Truth. Set (3) contains tuples which associate names with a group to which they do not belong. Although this is a simple example, the problem is not limited to the join operation; projection, composition, and product may also be used to alter the context of a sensitive field. Moreover, it is possible to completely remove the sensitive field in such a fashion that its contents are still known to anyone who knows the series of operations used to derive the final result.

The example used here demonstrates that falsification may be performed in one step; it is easy to conceive of much more subtle ways to do so under other conditions of sensitivity. It is possible to construct cases where falsification occurs without the use of the "*" convention anywhere in the derivation history.

To take a further example, suppose our user wishes to obtain the salary of the Provost, who is associated with the group "Admin", and who lives in Cambridge. In addition to the <Name, Salary> set and the <Name, Group> set, let the data base contain the following two sets:

<Name, Town>

(for everyone), and

Group	Town
Management	Cambridge
Admin	Cambridge
(etc.)	(etc.)

where the last set contains the relation between groups and towns. Since people from different groups may live in the same town, some Data Elements may appear more than once in the <Town> field. In particular, at least one person from Management lives in Cambridge. If the user performs the following operation:

(<Name, Town> join <Name, Salary>) join (<Town, Group>)

then the final <Town, Name, Salary, Group> set will contain false information. The second join in the sequence matches

on the <Town> field, and will contain both the tuple <Cambridge, Provost, salary, Management> and the tuple <Cambridge, Provost, salary, Admin>, since Cambridge matches twice. The user could then intersect out the Management salaries from this (false) set, and obtain the Provost's salary. Clearly, identifying the point of falsification is not straightforward.

The access control system must, therefore, know the derivation history of an acquired relation set in order to assign access rights, or it must know the Truth. It is not sufficient to have knowledge only of the state set variables and the content/context of the Requested RDS. Designing a system which knows the Truth is beyond the scope of the present work.

5.5.2 Access Computation Rules

Independent of the problem of Truth there are a number of cases in which access to an acquired set may be determined without resorting to the filtering technique of Section 5.4.

The only method by which one may increase access is by subsetting away that information which he is not allowed to see. In the example of Section 3, access increases from M to P when the non-Management personnel are removed. If there are no context or content restrictions it is not possible to increase access in this fashion. The Restriction_flag in the KRDST indicates the presence of such

restrictions; if it is off, each field of the acquired RDS is assigned the minimum access level of the corresponding fields of the input. Thus if the only restriction is

"Jones may not see salaries."

then nothing Jones can do will allow him to increase his access rights to the <Salary> field.

Moreover, the set operations union and product and the non-set primitive insert_tuple include all of the information of the input RDS's in the output RDS. The minimum access of the input fields is again assigned.

In the case of intersection and difference it is not necessary to know the Truth. These operations are defined only for inputs which have the same RD's, and therefore no use of "*" or "---" can produce false information in the output. Thus it is possible to increase access by subsetting if there are content/context restrictions on the inputs. In such cases, MARM evaluates the new ACL's of the acquired RDS in order to assign current access privileges.

All of the remaining operations may falsify either content or context during the derivation of the acquired RDS. Even removing the sensitive field completely by projection is insufficient. If the <Name, Salary> set is intersected with

Name	Salary
*	10000

and the <Salary> field of the result is projected away, the user still knows the salaries of the remaining names. Therefore, the minimum access level of the inputs is assigned for replace_tuple, projection, join, and composition. Moreover, MARM cannot guarantee the accuracy of an RDS which has one of these operations anywhere in its derivation history. Thus the False_content_flag and/or the False_context_flag are turned on in any RDS which is either derived directly from one of these operations, or which is derived from inputs which have the flags on. A set with either flag on not only prohibits increasing access, but also results in an acquired set which can never be used to increase access.

This is the most severe shortcoming of the interim access control system; lack of knowledge about Truth places restrictions not only on the users of sensitive information, but also on the originators. In the example of Section 5.5.1, the user could not acquire a set to which he has P access from the given system data base -- the join operation must be used at some point to establish the (true) <Name, Group> relationship. It therefore does not make sense in the interim system to place context restrictions on sensitive information unless that context appears in the basic data set. Moreover the interim system will in some cases over-classify an acquired RDS; information which should be released will not be released.

In summary, the interim rules for assigning access privileges to an acquired RDS are:

1) If there are no content/context restrictions on the inputs, or if the operation is anything other than intersection or difference, or if either of the inputs have the False_content_flag or False_context_flag on, then the access to each field is the minimum of the access levels to the input fields.

2) Otherwise, evaluate the new ACL's to determine the access rights.

5.6. Non-Standard Access Handling

It is the aim of the access control system to support access control at a level sufficient to handle the needs of the great majority of users. However, the system is not closed; a user who wishes to have special restrictions on his data can do so. The only constraint which the interim system places on users is that assigned access privilege levels must be members of the set of capabilities of Section 5.2.1.

Since the construction of the ACL's for an RDS is under the control of the originator of sensitive information, either he or anyone to whom he has granted the capability to change access information may place an arbitrary program (or programs) on the ACL. The <Access_handler> field of the ACL will normally contain the name of a system-supplied program which tests constraints of the form described in Section 5.4. In these cases, the <Data_rds> field will point either to a filter RDS or to the State Set or Password Set. However, the access handler may be a user-supplied program. In this case, the <Data_rds> field of the tuple may, of course, point to any data in MADAM format.

However, other users must be protected from the results of such programs. In the current Multics implementation a program called from a more privileged ring acquires the privileges of the caller; any program called

by MARM will have MARM's privileges. It will therefore be necessary for a user who wishes to use his own access handlers to submit them for approval by system supervisors.

6. EVALUATION OF THE INTERIM ACCESS CONTROL SYSTEM

This section summarizes both the good and bad points of the interim access control system.

6.1. Generality and Compatibility

Perhaps the most powerful feature of the interim system is its ability to represent access control information which is specified as a series of general logical constraints on the system state, the content of the sensitive field, and the context in which the sensitive information appears. The originator of information is no longer reduced to supplying an exhaustive list of the names of persons who can access his data; he can now make statements like

"Anyone can see his own salary."

Through use of logical "and", "or", and "not", the originator may concisely specify complex conditions for the release of his information.

This powerful representation of constraints is supported by a very general definition of the term "user identification" -- the state set supplies an extensive list of characteristics of the user which may be accessed for

many purposes. In addition, the fine distinction of the term "access" greatly extends the types of access that may be assigned.

Power is also provided by the separation of the concepts of the "owner" of information, the "originator" of information, and the person (or persons) who may change the access control restrictions. It is possible to give someone a physical copy of sensitive information and allow him to manipulate it while denying him the ability to pass the information along to another user.

The interim access control data base is compatible with the rest of MADAM data storage, with the exception of the <And_group> field of the ACL and the <fieldname> field of the ACL_control RDS, both of which require special handling. For the "pure" RDS's in the access control data base, all the MADAM operations are applicable; even in the case of these two impure sets, the non-set primitives provide powerful manipulative tools. Moreover, the interim system uses only MADAM operators to make its access control decisions, except for manipulations involving these two fields.

6.2. Extensibility

The interim system may be readily extended along both the second and third dimensions of access control.

The division of "access capability" into P, M, S, N, etc. is an arbitrary program restriction, and might be

made either more or less coarse as experience dictates. The access handlers are completely independent of this division; a little cleverness in the programming of the MARM module itself will make such changes very easy to make. One conceivable extension, for example, would be an expansion of the C capability. In the interim system a user may change all of the access control information for an RDS, or he may change none. However, the access control data base itself is just a series of RDS's which are protected by exactly the same mechanisms as any other RDS; subdividing the C capability would amount to assigning their access rights individually rather than collectively.

In addition, the use of the RSM for evaluated functions means that extension of the system to include more knowledge about the environment is a trivial matter -- since the state set is a pure RDS, and since the only interactions with it are through standard set operations, adding a field presents no problems.

Lastly, the fact that the access handlers may themselves be arbitrary programs provides a convenient escape for special conditions, as well as a short-term solution to some of the limitations of the interim system.

6.3. Limitations

Clearly the most important limitation of the interim access control system is its inability to know the Truth. On this account, its performance in assigning access

privileges to new RDS's is severely impaired. All of the operations which may be used to falsify content or context may also be used in quite legitimate ways for obtaining information which the user should be allowed to see, but which the interim system will not release.

The second limitation of the interim system is the complexity of the MADAM representation of access control restrictions in the ACL data bases. Expression of arbitrary "and"s, "or"s, and "not"s may well result in a large number of small RDS's which express the access control data for a single RDS. However, given the current configuration of MADAM, the representation presented here is the simplest possible.

A subproblem of the representation of logical constraints which further limits the access control system's operational power is the use of the <And_group> and <fieldnumber> constructs, which make their respective RDS's impure and therefore not subject to manipulation by set primitives. This limitation, however, is transparent to the user; it affects only the amount of special programming within MARM itself.

From the user's viewpoint a more noticable limitation is the fact that only current MADAM operators may be used in expressing constraints; constraints involving arithmetic operators therefore presently require special purpose programming. This restriction applies throughout

MADAM, however, and is not limited to access control.

The interim system's position along the dimension of physical level of control is relatively fixed: a decision to protect information at any level other than the field level within RDS's would be much more difficult to implement than a change in assignable access capabilities. However, such a change would affect only the MARM module itself. Moreover, the interim system allows one to protect specific fields of specific tuples; it is difficult to imagine an application for which this level of control would be insufficient.

The last important limitation of the interim scheme is the fact that user-supplied access handlers must be submitted for approval before they may be used. This restriction results from the current Multics limitation that a procedure called from an inner ring has all the privileges of the calling program. Multics is considering eliminating this restriction.

7. SUGGESTIONS FOR FURTHER STUDY

This thesis has clearly identified at least as many problems as it has solved:

First there is the problem of Truth. In a data management system which provides non-trivial mechanisms for manipulating information it is possible to construct false information in extremely subtle ways. This problem is by no means limited to the realm of access control -- it must be solved before MacAIMS can support a user interface which does more than map single commands into single set operations. Indeed, the problem affects users even if they specify set operations one-by-one; falsification might occur almost as easily by accident as by design.

Second, there is the problem of interfacing general logical expressions with the existing MADAM structure. In the access control system, this is evidenced by the complexity of the data structure which results from forcing logical expressions into a data format which is ill suited for their representation. This problem also transcends access control. The addition of boolean operators to MADAM merits investigation in its own right.

The need has also been shown for the addition of

arithmetic operators to the set of tools which MADAM supports. Such an addition is clearly desirable in providing power to the user.

These issues all exist separately from the problem of access control, though their solutions would greatly enhance the access control system's capabilities. In addition there are two problems concerning access control alone which should be examined:

First there is the question of results of computations which are final from the access control system viewpoint, but which the user views as interim results for a computation which takes place outside the computer. The access control scheme cannot know if a user takes two printouts from different operations (or different login sessions) and intersects them in his head. The fact that access rights are assigned at each interim step in a computation provides a powerful tool for solving this problem, but the responsibility for appropriate specification of constraints currently rests with the originator of sensitive information. A better understanding of the problem of partial results would help guide users in setting up their sensitivity specifications.

Lastly, there is the problem of conflicts of privacy. The interim system provides for only one originator of information, along with the capabilities which "originator" implies; it would be fairly easy to extend

these concepts to include multiple originators of information who would have to collectively agree to its use or dissemination. While such a simple idea would be a large step in the right direction, a much better theoretical understanding of conflicts of privacy is needed.

8. CONCLUSION

This thesis has not been a dissertation on privacy; it presents the design of an access control subsystem for a particular data management system. The preceding pages not only demonstrate that no system in common use today provides a working environment in which users may conveniently protect the rights of others; they also show that furnishing that environment is not (and will not be) an easy task. The MacAIMS access control system is complex in structure; it will probably be costly to use. Moreover, it is not the whole solution to the access control problem.

But even establishing the environment is not enough. The MacAIMS interim access control system has no morals; it protects no one's privacy. We do not provide control; we provide the ability to control.

Data processing has no social value in its own right: it is a means to meet other ends. Yet the drive to collect data is extremely powerful in itself; left alone, it often becomes the goal rather than the means. The morals of privacy cannot be built into the hardware -- they must be built into the users, the programmers, the system

administrators.

Today our ability to collect, process, and disseminate information far outweighs our knowledge of what to collect, how to process it, and when to disseminate it. The time to close that gap must be now.

BIBLIOGRAPHY

1. Baran, P., "Communications, Computers, and People", AFIPS Conference Proceedings, Fall, 1965.

2. Corbato, F. J., and V. A. Vyssotsky, "Introduction and Overview of the Multics System", FJCC, 1965, pp. 185-196.

3. CP-67/CMS User's Guide, IBM Cambridge Scientific Center Report 320-2015, Cambridge, Massachusetts, revised July, 1968.

4. Crisman, P. A., ed., The Compatible Time-Sharing System: A Programmer's Guide, M.I.T. Press, Cambridge, Massachusetts, Second Edition, 1965.

5. Daley, R. C., and P. G. Neumann, "A General-Purpose File System for Secondary Storage", FJCC, 1965, pp. 213-230.

6. David, E. E., Jr., and R. M. Fano, "Some Thoughts About the Social Implications of Accessible Computing", FJCC, 1965, pp. 243-247.

7. Dennis, Jack B., and Earl C. Van Horn, "Programming Semantics for Multiprogrammed Computations", Communications of the ACM, v.9, no. 3, March, 1966, pp. 143-155.

8. Evans, David C., and Jean Yves Leclerc, "Address Mapping and the Control of Access in an Interactive Computer", SJCC, 1967, pp. 23-30.

9. Falkoff, A. D., and K. E. Iverson, APL/360: User's Manual, IBM Thomas J. Watson Research Center, 1968.

10. Fano, R. M., "The Computer Utility and the Community", 1967 IEEE International Convention Record, part 12, p. 30.

11. Fillat, A. I., and L. A. Kraning, Generalized Organization of Large Data-Bases; A Set-Theoretic

Approach to Relations, Project Mac TR-70, M.I.T., June, 1970.

12. Glaser, E. L., J. F. Couleur, and G. A. Oliver, "System Design of a Computer for Time-Sharing Applications", FJCC, 1965, pp. 197-202.

13. Goldstein, Robert C., "The Substantive Use of Computers for Intellectual Activities", Project Mac TM 21, M.I.T., April 1971.

14. Goldstein, Robert C., and Alois J. Strnad, "The MacAIMS Data Management System", 1970 ACM SICFIDET Workshop on Data Description and Access.

15. Graham, Robert M., "Protection in an Information Processing Utility", Communications of the ACM, v.11, no. 5, May, 1968, pp. 365-369.

16. Hoffman, L. J., The Formulary Model for Access Control and Privacy in Computer Systems, SLAC Report no. 117, May, 1970.

17. Hsiao, David K., A File System for A Problem Solving Facility, University of Pennsylvania Ph. D. thesis in Electrical Engineering, May, 1968.

18. IBM System/360 Principles of Operation, A22-6821-7, September, 1968.

19. Lampson, B. W., "Dynamic Protection Structures", FJCC, 1969, pp. 27-38.

20. The Multiplexed Information and Computing Service: Programmer's Manual, Project Mac, M.I.T., Preliminary Edition, 1971.

21. PDP-10 Programmer's Reference Manual: Time Sharing Monitors, Digital Equipment Corporation DEC-T9-MTZA-D, Maynard, Massachusetts, August, 1969.

22. Peters, Bernard, "Security Considerations in a Multi-programmed Computer System", SJCC, 1967, pp. 283-286.

23. Petersen, H. E., and R. Turn, "System Implications of Information Privacy", SJCC, 1967, pp. 291-300.

24. Stone, M. G., "TERPS -- File Independent Enquiries", The Computer Bulletin, March, 1968, pp. 286-290.

25. Strnad, Alois J., "The Relational Approach to the Management of Data Bases", IFIP Congress, 1971.

26. TSS/360 Quick Guide for Users, IBM X28-6400-0, May, 1969.

27. Vyssotsky, V. A., F. J. Corbató, and R. M. Graham, "Structure of the Multics Supervisor", FJCC, 1965, pp. 203-212.

28. Ware, W. H., "Security and Privacy in Computer Systems", AFIPS Conference Proceedings, Spring 1967, p. 279.

29. Weissman, C., "Security Controls in the ADEPT-50 Time-Sharing System", FJCC, 1969, pp. 119-133.

30. Westin, Alan F., "Life, Liberty, and the Pursuit of Privacy", Armonk, New York, Think, v. 35, no. 3, May-June 1969.

31. Westin, Alan F., Privacy and Freedom, New York, Atheneum, 1967.

32. Williams, Stephen J., Richard C. Owens, Jr., Edouard Cointreau, and Peter Bloomsburgh, "Privacy, Technology, and the American Citizen," unpublished.

In addition to works specifically referenced, this bibliography contains a number of other articles relating to privacy and access control. For a more complete listing of works on the subject, the reader is referred to:

33. Harrison, Annette, The Problem of Privacy in the Computer Age: An Annotated Bibliography, RAND Corporation Memorandum RM-5495-PR/RC, December 1967.

This empty page was substituted for a blank page in the original document.

CS-TR Scanning Project
Document Control Form

Date : 1/23/96

Report # LCS-TR-89

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 92(98-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☒ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-92) UNTITLED TITLE, ACKNOWLEDGEMENTS,
ABSTRACT, 4-91, UNT BLANK.
(93-98) SCANCONTROL, COVER, DOD, TRGT'S (3)

Scanning Agent Signoff:
Date Received: 1/23/96 Date Scanned: 1/24/96 Date Returned: 1/25/96

Scanning Agent Signature: Michael W. Cook

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Massachusetts Institute of Technology
Project MAC

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP
None

3. REPORT TITLE
Primary Access Control in Large-Scale Time-Shared Decision Systems

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
M.S., Alfred P. Sloan School of Management, May 1971

5. AUTHOR(S) *(Last name, first name, initial)*
Owens, Richard C., Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1971	92	33

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-69-A-0276-0002	MAC TR-89 (THESIS)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D.C. 20301

13. ABSTRACT

Four primary dimensions of the access control problem are identified. They are: 1) the physical level at which to apply control; 2) the fineness of distinction applied to the term "access"; 3) the meaning of the term "user identification"; and 4) the degree of sophistication employed in automatically assigning sensitivity descriptions to derived data sets.

Within the context of MacAIMS, the Project MAC Advanced Interactive Management System, the design of an access control system is presented which takes positions along these dimensions appropriate for controlling access in a Management Decision System. Particular attention is given to computing access characteristics of new data sets derived from existing sets.

In addition, several classes of problems which the system cannot solve are presented.

14. KEY WORDS
Access Control Privacy Controlled Sharing

DD FORM 1 NOV 68 1473 (M.I.T.)

UNCLASSIFIED
Security Classification

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**