

BASE MANAGEMENT STUDY. (U)
DEC 78 H R HARTSON.

1 OF 1
ADA
103176

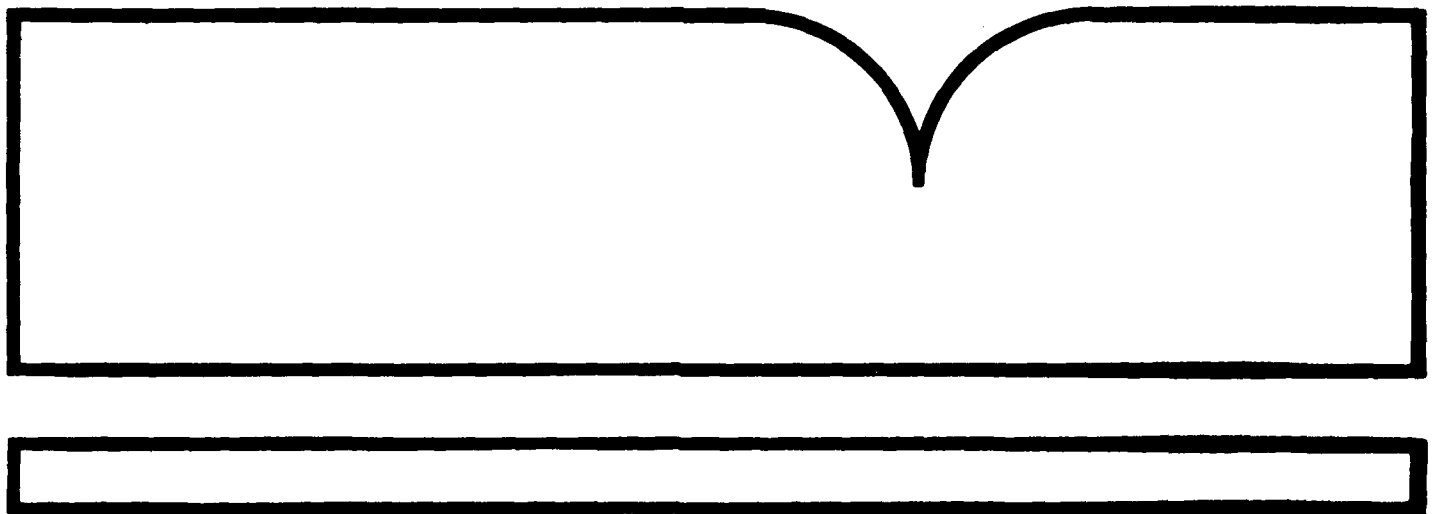
END
DATE
FILMED
32782
NTIS

AD-A103 176

SECURE DATABASE MANAGEMENT STUDY.

Army Institute for Research in Management Information
and Computer Science

DEC 78



U.S. Department of Commerce
National Technical Information Service

NTIS.

AD A103176

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

SECURE DATABASE MANAGEMENT STUDY

TECHNICAL REPORT

**Prepared For
The Army Institute for Research
in Management Information and Computer Science**

by

H. Rex Hartson

December 1978

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

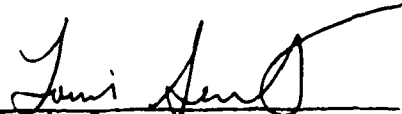
DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited

This Technical Report has been reviewed and is approved.



Clarence Giese, Director
U.S. Army Institute for Research
in Management Information
and Computer Science



Louis Sernovitz
Chief, Applied Science
Division, U.S. Army
Institute for Research
in Management Information
and Computer Science

ABSTRACT

This document reports the findings of a study of alternative system architectures for secure database management systems. System requirements are stated and the relation to operating system security is discussed. Security kernels, their history as well as their advantages and problems, are described. Additional security requirements for database systems are introduced and some models and experimental systems are reviewed, representing non-architectural and architectural approaches to non-secure and secure database systems. Conclusions and evaluations are made throughout. It is also suggested that military planners and system designers re-evaluate their traditional approaches to security policy, in order to take advantage of emerging technology. Broad-based recommendations are then made as a baseline for determining appropriate research directions in the area of data security.

TABLE OF CONTENTS

1. Introduction.....	1
1.1 Goals of the Project.....	1
1.2 Scope of Project.....	2
1.3 Terminology.....	3
1.4 System Requirements.....	4
2. Relation to Operating System Security.....	8
2.1 Introduction.....	8
2.2 Security Kernels--What They Are and Are Not.....	8
2.3 Kernelized Operating Systems.....	11
2.3.1 Background.....	11
2.3.2 Kernelized Secure Operating System (KSOS).....	12
2.3.3 PSOS and the HDM.....	14
2.3.4 Unresolved Issues.....	16
2.3.5 Guard--A KSOS Application.....	17
3. Non-Architectural Approaches to Database Protection..	20
3.1 Models.....	20
3.1.1 Hsiao's Attribute-Based Model.....	20
3.1.2 McCauley's Security Atoms.....	22
3.1.3 Fernandez' Model of Authorization.....	22
3.1.4 Hartson's Semantic Model of Database Protec- tion.....	23
3.2 Some Experimental Systems.....	23
3.2.1 ADEPT-50.....	24
3.2.2 ASAP.....	24
3.2.3 INGRES.....	24
3.2.4 System R.....	28
4. Architectural Approaches.....	30
4.1 Introduction and Motivation.....	30
4.2 Architectural Approaches to Database Management..	31
4.3 Architectural Approaches to Secure DBMS.....	34
4.4 Some Experimental Systems.....	37
4.4.1 The Ohio State Data Base Computer (DBC).....	37
4.4.2 MULTISAFE.....	41
5. Additional Issues.....	46
5.1 Networks and Distributed Data.....	46
5.2 Peripheral Issues.....	48
5.2.1 Abstract Data Types.....	49
5.2.2 Risk Assessment and Security Evaluation.....	50
5.2.3 Auditing.....	50
5.2.4 Data Integrity.....	51
5.2.5 Personnel Problems.....	52
5.3 Advanced Issues.....	52
6. Conclusions and Recommendations.....	54
6.1 Introduction.....	54
6.2 Modern Military Computer Security Policies.....	55
6.3 Summary of Conclusions and Recommendations.....	59
Acknowledgements.....	64
Cited References.....	65

1. INTRODUCTION

1.1 Goals of the Project

The objectives for this study were written in the Request for Short Term Analysis Service (STAS) as follows:

The objective of this requirement is to investigate alternative system architectures for secure Database Management Systems. Appropriate analysis is to include recent approaches such as back-end computers, database machines, networks, and distributed databases and Multiprocessor architectures. Each approach should be studied for feasibility, relative advantages and drawbacks, range of policies implementable by mechanisms provided, cost, and performance. The results of this study will produce a baseline for determining appropriate research directions in the area of security.

As the work progressed, it became increasingly clear that:

- 1) Owing to the state-of-the-art nature of the subject, insufficient detail exists to allow for analysis of cost and performance, and for significant cost comparisons to be made.
- 2) Direct comparisons of the various alternatives and approaches was rendered inapplicable because of the varying stages of development of the approaches. A second, more important, reason is that fundamental differences in approach tend to invalidate direct comparisons because they are comparisons of "apples and oranges."

As a result, through telephone conversations with people at AIRNICS, the COTR organization, the objectives of the work have evolved to deemphasize quantitative analysis, while the main thrust of the study has remained the same. A statement of specific goals used as a guide for interpreting and achieving the stated objectives is as follows:

- 1) Collect information as comprehensively as possible from military, academic, and commercial sources.
- 2) Report findings of 1) above.
- 3) Make broad-based recommendations about which general directions appear most promising.
- 4) Provide information and advice with which military planners and system designers can reconsider their traditional approach and be prepared to take advantage of new developments in the state-of-the-art.
- 5) Attempt, whenever possible, to use up-to-the-minute results to assist in achieving an awareness of alternatives necessary in reaching goal 4) above.

In several cases there was a minor difficulty with 5) above, in that security restrictions presented barriers to obtaining the necessary information. This was particularly true with military organizations and their contractors (e.g., DCA, SDC, I. P. Sharpe). In other cases, with academic and industrial sources, very current information was obtained, including descriptions of work in progress but not yet published. A personal approach to information gathering made use of letters and follow-up telephone calls. Also, personal participation in the planning and execution of an Army Automation Security Workshop was helpful to the author in identifying, and relating this project to, Army needs.

1.2 Scope of Project

This section briefly delimits the type of security to be discussed, except for a short mention of some of the "peripheral issues" in section 5.2. The focus of the work is on secure database management. The security of operating systems is addressed only in that it relates to that of database systems. The scope specifically excludes topics such as personal identification, physical access controls to

computer facilities, and cryptography. While these are interesting and necessary parts of an overall secure operation, they are not in the objectives of this study. These topics are being studied by others, each as a separate field of interest.

1.3 Terminology

This section will serve to define some of the terminology that the reader may encounter in this and other literature on the subject. Some of the definitions are a bit arbitrary in that no standard definitions have yet been established. (Also, some of these definitions--e.g., integrity--are not quite the same as those found in AR 380-380 [ARMYR77], but they are consistent with the literature.)

Protection--The preservation against unauthorized use of computing resources, especially the protection of data from accidental or deliberate disclosure or modification. Protection includes security, privacy, and integrity concerns.

Security--The protection against deliberate disclosure or modification of data in cases involving national defense. Security also covers cases involving industrial economics (e.g., industrial spies) and commercial finances (e.g., fraud).

Privacy--Protection of data about people; in particular, the protection of the legal rights of individuals and organizations to self determination of the degree to which information about themselves can be collected, stored, and shared with other individuals and organizations. Privacy includes confidentiality, which is the protection of individual anonymity while globally processing (e.g., gathering statistics) data about people.

Integrity--Operational integrity is the protection of the logical consistency of data by properly synchronizing accesses that modify data. Semantic integrity is the protection of the logical consistency of data by checking all inputs and updates against specified constraints. (This is different from the AR380-380 [ARMYR77] definition of integrity which refers to it in terms of overall system security as: "The capability of an ADP system to perform its intended function in an unimpaired manner, free from deliberate or inadvertant unauthorized manipulation of the system.")

Protection policies--Policies are the rules of access to computing resources.

Protection mechanisms--Mechanisms are the means to implementing policies within a given protection system. A security system may be viewed as having a "pool" from which mechanisms may be selected and combined to implement any given policy. As an analogy, the basic instruction set of a computer provides mechanisms to be put together to form programs which execute policies. When policy is built into mechanism, it should be done deliberately, with a conscious appreciation for the tradeoff being made of flexibility for economy.

1.4 System Requirements

This section describes a number of protection features which are useful or necessary in a forward-looking secure military data management system. Many of these requirements will be referred to by name later in the report.

- 1) Multilevel protection--In a military system, this term often means that more than one security classification level can exist together within the system. In other contexts, the term has been used to refer to protection

mechanisms which occur at more than one level of software and hardware. The term also can imply that protection checking can be done at several levels (granularities) of data (e.g., system, file, record type, record instance, field).

- 2) No "back doors"--All data is accessed via a common entry point to the DBMS. Hidden entries are guaranteed not to exist.
- 3) Decentralized authorization--Permission to grant access privileges exists among many different authorities, each of which has security responsibilities for different, possibly overlapping, parts of the database. This feature is useful in eliminating the operational bottlenecks found with a centralized database administrator (DBA), but it complicates the authorization programs considerably and requires the concept of resource "ownership" to be implemented. An owner is one who can propagate access rights to a given resource. Propagation of ownership brings about the problem of access privilege revocation [GRIFP76], a problem without an automated solution. Of course, decentralized authorization requires discretionary controls (not a system based on just security levels).
- 4) Reasonable performance--Security has its cost in overhead of processing and storage. Performance penalties must be limited to a reasonable percentage.
- 5) Uniformity of mechanisms--One general protection mechanism, uniformly applied, is safer than a static mechanism requiring the handling of many exceptions [CLAYB78]. The same mechanisms should be used, regardless of the purpose of the protection; for e.g., security, privacy, and integrity should be implemented with a common set of mechanisms.

6) Multiplicity of binding types--More than one access

- 7) Integrated security--Protection mechanisms are to be integrated into the heart of the system architecture from the start and not "added on" to existing database systems. Many early system design decisions must take security requirements into account. Without security requirements, these decisions will be made in a different way, in favor of other factors (primarily in favor of performance).
- 8) Improved protection precision--Fixed unsophisticated protection mechanisms have been limited in their ability to adhere with precision to the increasingly complex policies of the database environment. This imprecision causes access decision errors [HARTH77].
- 9) Dynamic authorization--Authorizers have the ability to change access rights interactively while the DBMS is operational. This is becoming a requirement for any on-line DBMS. It is also useful in supporting the "need-to-know" principle.
- 10) Kernels--Kernels and methodological aids to verifying system security are necessary, in combination with rigorous software testing.
- 11) Cooperative authorization--As a natural consequence of decentralized authorization, authorization from several cooperating authorities [MINSN77] might be required for certain highly sensitive operations.
- 12) Generalized and distributed data--Generalized DBMS are required for the operations addressed in this report; simple file management operations offered by typical operating systems are not adequate. In many cases the data (and DBMS) will be distributed within a network of systems.
- 13) Protection languages--An easy-to-use interface is necessary to allow authorizations to be made correctly and in a timely manner. Protection languages

requested data is authorized for access. The level of granularity with which enforcement mechanisms can deal in such cases (for example, partial access vs. "flat yes or no" decision) is a question of "resolution of enforcement" [HARTH77]. If policies allow for partial access, mechanisms will need to have the ability to filter out the unauthorized portion from the authorized portion of requested data.

- 15) Range of access decision dependency—in general, enforcement decisions need to be based on many variables. The degree to which protection mechanisms are sensitive to these variables determines the range of policies implementable by those mechanisms.
 - a) data names and types (data definition dependency)
 - b) data values within the data being retrieved
 - c) data values elsewhere in the database
 - d) input data values (integrity constraints for insertion and modification)
 - e) system state variables (e.g., time of day, switches, modes of processing, status indicators, etc.)
 - f) user related information (e.g., user ID, re-authentication results, terminal, time of login, etc.)
 - g) access history information
 - h) results of procedural protection measures "triggered" by the access request
- 16) Exception reporting—In real systems, mechanisms are needed for handling and reporting exceptions. This includes an ability to produce journal files, protection logs, and audit trails. It also includes the ability to handle and report penetration attempts (accidental or deliberate) without greatly affecting the performance of normal operations.

2. RELATION TO OPERATING SYSTEM SECURITY

2.1 Introduction

While the security of operating systems is not directly part of database protection, the close and complex relationship between operating systems (OS's) and database management systems (DBMS's) warrants the consideration of OS security in this report. Indeed, many of the functions provided by OS's (e.g., low level input/output, resource interlocking for shared usage, etc.) are critical to DBMS operations. With vulnerable OS functions, secure DBMS are of little use in overall system security. As this section concentrates on recent work in the area of security kernels, the reader is referred to the literature for a more comprehensive discussion of OS security [SALTJ75, HARRM75, JONEA75, etc.].

2.2 Security Kernels--What They Are and Are Not

The testing of large modules of software has been shown to be insufficient, by itself, for demonstrating that the software is secure [ANDEJ72]. On the other hand, thorough software testing has certainly been shown to play an important role. (See Tanenbaum [TANEA76] for an amusing, but realistic discussion of this viewpoint.) In fact, a two step system certification procedure, as a combination of verification and installation review and testing has been suggested for military computer security [WALKS77].

The difficulty of verifying large systems of software has led to the concept of security kernels into which all security related software is to be concentrated into a smaller, certifiable protected nucleus, separate from the rest of the OS functions. Non-kernel OS software can then be allowed to run in an unprotected environment. As most of the recent work in kernel implementation has shown [MILLJ76, POPEG78a, POPEG78b] kernels have gone a long way toward reaching this goal. A kernel-based approach forces the system software to be very well structured and to be highly transparent (straightforward, readable, and easy to understand without following "clever" gyrations within the code). This structure, though perhaps achieved at a slight cost in performance, has increased the level of confidence of OS security greatly over that of the previous generation of flaw-ridden, overly complex patchworks of software.

But kernels, as the literature has not been quick to point out, still do not guarantee security. The proof of a program is not a guarantee of its reliability [TANEA76]. Furthermore, proving individual programs is not the same as proving a system of interacting programs. It is also difficult to prove that all necessary code is in the kernel. This requires proving something about the entire system of software, and one reason for having a kernel is the difficulty of proving things about the entire system. Further, it is a reality (at least within current state-of-the-art) that some security related software must remain outside the kernel, if its bulk is to remain within reasonable limits. Software for resource management and physical device support is typically among the outcast code that, in fact, is sensitive to security flaws. Sometimes this requires "trusted" non-kernel software [POPEG78a], which leads to various levels of kernels. It is part of the "bowl of spaghetti" syndrome, a term coined within the MULTISAFE project (MULTISAFE is discussed in section 4.4.2). As one attempts to separate, with each bite, an amount that can be reasonably han-

dled, the complex interconnections came more and more spaghetti to be pulled from the bowl. Within the MULTISAFE project the operational definition of the ideal kernel is "that minimal collection of software such that no fault or subversion of other software can cause security to be compromised." However, as a practical matter, the bowl of spaghetti syndrome requires that a considerable quantity of security software to be outside the kernel. In all systems this non-kernel software can have an impact on security. An important way that subverted non-kernel software can compromise security is by passing false information to the kernel. If false parameters are accepted by the kernel, unauthorized accesses could be allowed by an otherwise perfect and proven kernel. For example, a subverted (or subversive) DBMS might "lie" about the nature of the data it is trying to pass through security checks back to the user.

It has also been suggested that non-kernel software be allowed to do much of the work of a given function and then have the kernel check the final results. In some cases, this is a reasonable way to reduce kernel bulk. However, the thorough checking of some processes implies a checking of how the results were obtained. In these cases, checking can require the ability to reproduce or emulate the process being checked—not a condition allowing for substantially less volume in software than that of the process (being checked) itself.

Improved hardware and firmware support (see [POPEG78a] for examples) offers hope for even more secure and more compact kernels in the future. Improved methodology [ROBIL77] offers hope for more secure overall systems.

2.3 Kernelized Operating Systems

2.3.1 Background

Over most of the past decade several projects have contributed to the present technology of secure OS. Starting in about 1968, penetration efforts of "tiger teams" revealed the appalling state of security in operating systems. More importantly, those tests showed that finding and patching of security leaks was not a viable approach. Penetrators never failed to find more holes and errors. And, even if more could not be found, their non-existence could not be proved. Designers would be required to start over from scratch, to design and develop operating systems in a new, highly rigorous and systematic way. Early research (circa 1972) sponsored by the Electronic Systems Division of the Air Force yielded the concept of the reference monitor which performs complete mediation of every memory reference (every access). It was partly from this work that the kernel concept and its properties of isolation and correctness were derived. Early Mitre modelling [BELLD73] set military policy (e.g., the star property) in a formal mathematical model.

MULTICS, developed at MIT and one of the earliest security oriented systems [SALTJ74], was not based on formalisms for verification of software. It has been subsequently adapted by Honeywell as Multics, the most secure system commercially available. It also has been kernelized in an experimental project called Guardian within a combined effort by Honeywell [HONEY76], MIT [SCHRM77], the Air Force, and the Stanford Research Institute (using SRI methodology). Formal specifications were developed, but not implemented. Honeywell developed a Secure Communications Processor (SCOMP) [GILSJ75], which began as a Honeywell level 6 mini-computer-based front-end and remote communications processor for the Guardian secure Multics system. As such, SCOMP was

a hardware oriented approach with a Security Protection Module, which connected to the system bus and checked all accesses to protected objects. After Guardian was terminated in 1976, work on SCONP continued, and it developed into a PDP-11 "look-alike" (software compatible) with a KSOS-like (see the next section) secure-UNIX operating system. SCONP is sometimes referred to as "KSOS-6." SCONP uses a Multics-like ring structure to implement domains of execution, and the hardware offers a significant amount of kernel support. Also SDC has been working on KVM, a kernelized version of IBM VM 370 [GOLDB77]. KVM, a three year project which began in 1976, will guarantee the separation of virtual machines in VM/370, removing the necessity for periods processing. Virtual machines executing processes at different classification levels may reside together in the same system. KVM is targeted for use in DCA, DCBC, and other intelligence processing sites. Gerald Popek [POPEG78a, POPEG78b] has developed a kernel based, verifiably secure OS at UCLA and has adapted it to the Bell Laboratories' UNIX operating system for the PDP-11. Mitre has independently, also using a kernel approach, produced a secure UNIX-like OS design.

2.3.2 Kernelized Secure Operating System (KSOS)

The Department of Defense (DoD), however, needs production quality systems, which are the next logical step after these "bread board" systems.

In 1978, the Department of Defense established a Computer Security Initiative, the goal of which is to achieve widespread availability of secure ADP systems within the DoD. Accomplishment of that goal will depend on involvement of computer manufacturers and on effective mechanisms for DoD approval of secure ADP systems. The recent DoD directive (5200.28) establishes proper channels, technical evaluation procedures, and mechanisms to facilitate solutions to

this latter problem. In response to the former, the DoD Computer Security Technical Consortium (Stephen Walker) has funded a competitive effort [WALKS77] to produce a Kernelized Secure OS (KSOS). The maturing technology from the initially separate efforts described above, taking place over different scales of times and with varying objectives, has come together in one project. It is an excellent example of how government funding, applied at the proper stage of development, can provide mutual benefit to military and private enterprise and advance the state-of-the-art at the same time. As a result of this experience (with KSOS) a funding policy may be emerging: the government appears to be willing to fund original design and initial implementation, but not successive refinements after that.

The privilege to produce the DoD production quality kernelized secure OS was won in a competitive design phase, over TRW by Ford Aerospace (E. J. McCauley), with SMI (Peter Neumann) as a subcontractor. Gerald Popek of UCLA was a consultant to TRW for part of their work and is now a consultant on KSOS. KSOS was presented to the U. S. Army Computer Systems Command (USACSC) at a Software Symposium in Williamsburg, Virginia, on 25-27 October 1978. There are also several papers on KSOS being prepared for the 1979 National Computer Conference (NCC). Among the DoD participants are the Defense Communications Agency (Ken Moore, CCTC) and the Army (Ft. Monmouth). The main sponsor is DARPA.

The goal of the KSOS project is a commercially viable (production quality) secure OS, externally compatible with UNIX and implemented on the PDP-11/70. UNIX is a popular, straightforward and uncomplex, OS for the PDP-11, designed by Bell Laboratories to be simple and efficient. However, it was not originally designed with security as a primary objective. There are numerous opportunities for "Trojan horses" and trap doors [NEUMP78]. It is very vulnerable

through its "superuser" state. The goal of the KSOS effort is to produce a secure OS with the good features of UNIX and with comparable operational efficiency. The multilevel design partitions the system into: a kernel, trusted non-kernel security related software, untrusted non-kernel security related software, and an emulator. The emulator provides compatibility with an existing UNIX-like user interface and has no effect on security. The trusted non-kernel security related software is partly motivated by concerns for efficiency. The kernel is actually a "complete," small operating system with efficient I/O and can be used by itself in a dedicated special purpose system (e.g., secure communication front-end). Also, other (non-UNIX) operating systems can be built upon this same kernel.

Phase I (of KSOS), the design phase, is completed. The design exists as a formal specification, and its security properties will be formally verified. In Phase II, the implementation phase, the kernel is supposed to be running by Spring 1979 and completed by the following Fall. Selected parts of the implementation will be proved to be faithful to the design. Many applications are being planned and developed to run on KSOS. Of interest in this report is a secure DBMS, rumored to be like Cullinane's IDMS. Others include a secure network front-end, flexible message handlers, and the Guard system described in section 2.3.5

2.3.3 PSOS and the HDM

Peter Neumann [NEUMP77] and others at Stanford Research Institute (SRI) have designed a Provably Secure Operating System (PSOS) using SRI's Hierarchical Design Methodology (HDM) [ROBIL77]. It is interesting that PSOS is not based on the kernel concept. This makes PSOS more general than KSOS, for example, in that PSOS can support many different security policies, not necessarily based on the star-property. It also more easily supports discretionary access

controls. PSOS uses capability-based addressing and hardware tagging of capabilities to insure their non-forgeability. PSOS is not implemented yet. One of the problems is that existing hardware cannot support it. It is hoped that most of the necessary modifications can be made at the firmware level. The Navy is currently interested in developing a PSOS prototype.

Perhaps the most important aspect of the PSOS project is the Hierarchical Design Methodology, used in the design and development of PSOS. Based on the fundamental notion that software tools are the most important factor in achieving secure systems, HDM provides a broad range of tools allied to an integrated methodology. HDM emphasizes formalism in making the development process precise. HDM is comprehensive in that it applies throughout the design and development process of a system; it is not just about program proving. As such, it enhances software reliability as well as security. (Of course, one cannot really believe in the security of an unreliable system, anyway.) HDM concepts include hierarchical decomposition, structuring, abstraction, modularity, formal specification, data representation, and program verification. HDM mechanisms include abstract machines at many levels (operations and internal data structures), modules, stages of development, and system families. Among HDM languages are a Hierarchical Specification Language, a SPECification and Assertion Language (SPECIAL), and an Intermediate Level Programming Language (ILPL). SPECIAL allows formal, non-procedural specifications to be expressed in a predicate calculus form. Programming languages for implementation of HDM designed systems must be strongly typed, have no aliases of objects, and do no hiding of implementation details. Several modern languages such as Modula, Euclid, Gypsy, and Ada (nee DoD/1)--all variants of PASCAL--are possibilities, but each has its own drawbacks. ILPL is a minimum language designed to just satisfy HDM requirements.

MDM development stages begin with specification of requirements and, then, system modeling. Next are design (yielding specification proofs) and implementation (yielding code proofs). The PSOS project plan is to verify the entire operating system, right down to the code level. The development process, within a hierarchy of abstract machines, builds each machine by writing an abstract program for the functions of that machine in terms of the primitives of the next lower machine. Within PSOS, for example, some sixteen levels of abstraction have been identified.

A detailed example of this methodology, applied to a list processing problem, is given in [SPITJ78].

There are other languages and methodologies which have been developed independently (for example, GYPSY [AMBLA77]), but a review of these is beyond the scope of this report.

2.3.4 Unresolved Issues

The state-of-the-art is sufficiently undeveloped so that there still remains a difference of terminology from one group of researchers to another. Single definitions for terms such as trusted, semi-trusted, untrusted, responsible, and privileged (as applied to software processes) are not universally accepted. Also, a difference of opinion exists as to how much software must be verified. For example, SRI's approach requires verification of all security related software, whether or not it is in the kernel. The author believes, along with the SRI people, that confidence in any approach that does not verify all security related code (such as appears to be the case in UCLA's Secure UNIX and SDC's KVM) cannot be complete.

Some of the trusted software of KSOS cannot be contained in the kernel because it (intentionally) violates the

security axioms. An example of such a process is one involving declassification or "sanitization" of data. In KSOS these processes will be subjected to the verification methodology just the same. Those procedures will point out the intentional "violations" of the security axioms, but no other violations will be allowed.

On the other hand, the KSOS spooler is not in the kernel, either. It is in a grey area called "responsible" software. It is not directly involved in accessing data, but it could possibly be subverted to send data to the wrong people.

In sum, 100% secure operating systems are probably not achievable. However, highly secure systems are attainable, and such systems will eventually be accredited for military use.

2.3.5 Guard—A KSOS Application

For many military applications there is a growing need to be able to interconnect many computers, often several with different classification levels. The passage of data from high to low levels will require careful sanitization and declassification. As a near-term response to this need, MITRE and the Navy (Navalex) have teamed up to develop Guard, a semi-automated system which acts as a filter between two computing systems operating at different classification levels. Guard runs as an application under KSOS on a PDP-11. The sanitization is done manually in a high classification environment by guard officers. A security watch officer is then responsible for passing the data to the low classification environment. In a network such as the ARPANET, the Guard system might be called upon to pass ARPANET messages, data management queries (to the Datacomputer), and data responses. Policy requires one guard officer per high-database/low-database combination, so that the

officer can become familiar with the particular databases and their usage. Both requests and responses can be edited by the guard officer, or either can be completely rejected. Network mail between different classification levels will also pass through Guard. Mail is not edited, but is checked by the security watch officer.

2.4 The Transition to Database Management

At first it might seem that, since DBMS's have certain functions in common with OS's, an extension of OS security principles should cover database security. While it is true that each type of system deals with sharing of structured data, there are several intrinsic differences which make DBMS's much more than an extension of the file handling capabilities of present day OS's. Correspondingly, database security is not just an extension of OS security. Some of these differences are as follows. This list is based on a list developed by the author and expanded on by Fernandez in [FERNE77] and most of the items are taken verbatim from that source:

- 1) Databases need a finer protection granularity. Field-level and data-dependent control are required in order to apply a need-to-know policy in a shared database environment.
- 2) Databases need a finer granularity for sharing, too. Therefore, interlocking for shared access can be more difficult [GRAYJ75, BAYEE76, RIESD77].
- 3) In shared databases there is a large number of data objects with complex interrelationships and a broad variety of data types.
- 4) While the OS is concerned with the name and address spaces of the data, the DBMS is also concerned with the semantics of the data. This implies the need for content and context dependent access control in such

an environment. Thus, there is an increased need for more dynamic protection checking with later binding times.

- 5) The need for data independence in DBMS requires that definitions of the data objects and their protection specifications be logical entities, far removed from the physically oriented objects used by the OS. This is required also to reflect the semantics of the data.
- 6) A richer variety of access types is needed in DBMS's, such as statistical access, administrative access [FERNE75a], etc. Again, OS's usually deal with physical access types such as read, write, and execute.
- 7) In an OS, data units correspond to real resources. In a DBMS there may exist data aggregates (e.g., views) that do not correspond to any physical entity but are dynamically built according to an application request [ASTRM76, SUNMR77].
- 8) In a DBMS each user may have a different conceptual view of the data objects and their relationships (e.g., subschema, submodels).
- 9) User interfaces in DBMS are usually more constrained (e.g., use of special query languages, special procedural languages, special access protocols, "parametric" access).
- 10) The lifetime over which the data is used is normally longer in a DBMS, and the number of associated applications is usually larger.
- 11) Objects are often (possibly null) sets of data entities, defined implicitly by predicates as characteristic functions, causing them to overlap arbitrarily. The tidy definition of objects as disjoint files

of higher level checking, too. Conventionally, DBMS run as large applications "on top of" the OS. In the MULTISAFE project (described in section 4.4.2) it has been proposed that all external storage access operations, both for OS file handling and for DBMS operations, be executed and controlled in one place—a sort of Data Base Access Method. The OS then operates "on top of" the DBMS, instead of vice versa. The checking at the various levels can then be integrated and coordinated.

3. NON-ARCHITECTURAL APPROACHES TO DATABASE PROTECTION

This section uses some selected models and systems to illustrate the state-of-the-art of non-architectural approaches to database protection. It is not intended to be a complete review of all existing database protection work.

3.1 Models

This section briefly reviews a few models of database protection. Each of these models has, since its development as a descriptive model, evolved as the "backbone" of a corresponding architectural approach to database protection. The models are discussed here. The respective system architectures are discussed in section 4.

3.1.1 Hsiao's Attribute-Based Model

their directories encompass many of the more commonly used file structures (e.g., inverted files, indexed sequential, and multilist) as special cases. A record in a file is a set of attribute-value pairs. Selected attribute-value pairs, now called keywords, are used in an index to speed retrieval. Records having keywords in common are linked together by pointers into lists within the database. Typically, a record is a node in several lists at one time; i.e., each record can be an intersection point of several linked lists. The directory for a file contains the information necessary for rapid processing of these lists. For example, for each keyword, the directory has the number of lists corresponding to the keyword, the length of each list, and pointers to the beginning of each list. Queries are expressed as logical (AND and OR) combinations of keywords. Algorithms have been developed to decode the directory entries for given keywords and to process the, possibly many, corresponding lists of records in parallel. The algorithm minimizes the total number of records inspected while retrieving the records which correctly answer the query.

Numerous researchers have since elaborated on the attribute-based model (often called the "Hsiao-Harary" model), including work reported in [YAOSE76, WELDJ76]. The model has also served as the basis for implementation of the Highly Secure Database Management System (HSDMS) [HSIAD76a, HSIAD76b] at Ohio State University. HSDMS has been demonstrated to be viable in a military environment using a Navy ocean surveillance application database [MANOF77]. An architectural approach, the Data Base Computer (DBC), has since been based on the same model and it will be discussed in section 4.

3.1.2 McCauley's Security Atoms

Additional concepts are introduced in [MCCA75] which enhance the security aspects of the attribute-based model. Specific attributes of a file can be selected as "security attributes" and used to group records for security purposes. A security attribute and its value are a "security keyword." Each record then contains a set of zero or more security keywords (security attribute-value pairs). This set is a "security atom" and all records having the same security atom are grouped together logically by "belonging to" that atom. Security atoms are disjoint and they can be protected individually, as objects. The security keywords of each access request themselves form security atom(s), and they are checked against a list of protected security atoms for an access decision. This mechanism, along with others, is used in HSDMS and the DBC.

3.1.3 Fernandez' Model of Authorization

Fernandez, Summers, and Coleman [PERNE75a] have developed a formal model of authorization which can impose access rules on shared databases accessed through high level languages (e.g., PL/I). Programming language extensions allow for database interaction and for users to define and use views of the data. The user's intentions, with respect to the database application, are made explicit. Enforcement can take advantage of more than one binding time, but is accomplished primarily at compile time.

In [PERNE75b] the model is extended by a data structuring scheme to facilitate authorization by providing flexible ways to define access rules. As these access rules are entered, their effects propagate through the system, and the rules are then discarded (to avoid possible internal inconsistency with later rules). Some architectural extensions to this approach are discussed in section 4.

3.1.4 Hartson's Semantic Model of Database Protection

A model of protection semantics divides protection into authorization and enforcement processes. A model is developed in [HARTH76a] as a semantic base for constructs in protection languages used by authorizers to express protection policies. The model is based on sets of users, resources, and data operations. Predicates, called access conditions, allow access decisions to have a wide range of dependency on system state. Access conditions are combined and evaluated by the enforcement process to produce the access decisions. Additional protection features are introduced in [HARTH76b]. Access history keeping allows dependency on the occurrence of previous data operations. Auxiliary program invocation provides for additional procedural protection measures such as auditing, alarm and recovery, and threat surveillance. The tradeoff between precision and performance is studied in [HARTH77]. Here, the questions of enforcement and dynamics and timing are addressed in the context of the semantic model and the concept of access decision binding time is introduced.

3.2 Some Experimental Systems

This section uses a few existing experimental secure database systems to illustrate the state-of-the-art for systems not based on an architectural approach. It is not a complete review of all experimental systems. The discussion emphasizes two recent relational database systems: INGRES (representing academic research) and System R (representing industrial research).

3.2.1 ADEPT-50

Although ADEPT-50 [WEISC69] is neither state-of-the-art nor a database system, it earns a brief mention here because it is one of the first military oriented attempts at secure OS file handling. ADEPT-50 provides security for objects such as users, terminals, jobs, and files in a time sharing environment. In a set theoretic approach, ADEPT-50 features the concept of a "job umbrella," a derived clearance of a user/terminal/file combination. The job umbrella's "high water mark" is used to automatically classify new files created by a job.

3.2.2 ASAP

ASAP [CONWE72] is an example of an early information system with security mechanisms. Data independent checking is done when high level queries are compiled. Field level protection is accomplished by assigning each user a list of security classes (based on attributes, or field names) he/she can access. Data dependent checking is done at execution time by Boolean expressions. ASAP is now available commercially.

3.2.3 INGRES

INGRES (Interactive Graphics and Retrieval System) is a relational database system developed and implemented by Stonebraker and others at the University of California, Berkeley [STONN76a]. The approach to protection taken in INGRES [STONN74] has attracted a lot of interest. Queries, in a high level query language language, are modified (using information about what kinds of queries a user is authorized to ask) as they enter the system. Modification is done in a way so that the modified version of the query is a legal request. In other words the modified query asks for only the authorized subset of what the original query requested.

Query modification is done before retrieval processing begins. Retrieval may then proceed, using the modified query, without further access checking. This high level approach to protection offers easy implementation and small execution time overhead.

In order to describe the way in which query modification is accomplished, the concept of relational databases is sketched. The relational model of data probably has the distinction of being the most complicated model, and--at the same time--the simplest model of data. On the one hand the model is associated with predicate calculus, meets and joins, functional dependency, normal forms, and so forth. On the other hand a great deal of understanding can be had without going beyond the plain fact that data is kept in a flat, unstructured table (the most natural form imaginable) without pointers, hierarchies, or even any computer related notions. The protection of INGRES can be explained without departing far from this simple view. The following examples (most examples in this section are adapted from [STONM74]) will be used to demonstrate INGRES query modification. Consider this relation (table of data), named EMPLOYEE:

<u>NAME</u>	<u>DEPT</u>	<u>SALARY</u>	<u>MANAGER</u>
Smith	toy	10,000	Jones
Jones	toy	15,000	Johnson
Adams	candy	12,000	Baker
Evans	candy	14,000	Todd
Baker	admin	20,000	Harding
Harding	admin	40,000	none

This table simply lists values for four attributes of employees of a certain company. The query language used to retrieve (and update) data in INGRES is called QUEL (QUEry Language) and is typical of a class of such languages which retrieve by data name and content and do not require knowledge of data structure or retrieval algorithms. (SEQUEL--Structured English QUEry Language [ASTRM76]--is another language of this type.) Queries in these languages typically have parts that state in simple terms:

- a) the name of user work area in which the retrieved data is to be returned
- b) the name of the attributes to retrieve
- c) the name of the table to retrieve from
- d) a condition for selecting rows to be retrieved

The following query is adapted to illustrate the use of a language like QUEL or SEQUEL for retrieval from a single table:

```

INTO W
RETRIEVE SALARY
FROM EMP
WHERE NAME = 'JONES'

```

The individual parts are explained as follows:

INTO W—the name of the user work area in which to put the retrieved data.

RETRIEVE SALARY—for whatever rows are retrieved, it is the **SALARY** attribute (column) which is of interest in this query (other attributes will not be given to the user).

FROM EMP—the name of the table from which to retrieve.

WHERE NAME = 'JONES'—the condition (attribute and value) to be used for selecting rows to be retrieved.

From the sample database, this query would select one row, namely the second row (as this is the only row having a value of 'JONES' for the **NAME** attribute) and return the **SALARY** value of that row, namely 15,000, to the requesting user's work area (buffer) named "W." The attributes to be retrieved (**SALARY** above) are called "target" attributes.

An access control restriction for a user can itself be specified in the form of a query. For example, suppose that Smith can see only information about himself (i.e., rows where **NAME = 'SMITH'**). A query-like statement of this restriction is:

```

INTO W
PERMIT NAME, DEPT, SALARY, MANAGER

```

```
FROM EMP
WHERE NAME = 'SMITH'
```

If Smith inquires about the salary of Jones with:

```
INTO W
RETRIEVE SALARY
FROM EMP
WHERE NAME = 'JONES'
```

the query is modified with the restriction by forming a logical AND of the conditions:

```
INTO W
RETRIEVE SALARY
FROM EMP
WHERE NAME = 'JONES' AND NAME = 'SMITH'
```

Since each row can have only one value for the NAME attribute and since this modified query asks to select all rows with both values under the NAME attribute, no records are retrieved and the illegal request is successfully denied. In general, the modified selection condition becomes the original query condition ANDed with a quantity which is the logical OR of all applicable access control specifications. Given this additional control specification, allowing Smith to see information about people who are in the candy department:

```
INTO W
PERMIT NAME, DEPT, SALARY, MANAGER
FROM EMP
WHERE DEPT = 'CANDY'
```

Query selection predicates will then be ANDed with:

```
NAME = 'SMITH' OR DEPT = 'CANDY'
```

The single relation (only the EMPLOYEE information) hides some complexity, but serves well to illustrate the principle. Cases involving more than one relation and queries involving aggregates (e.g., the average of all salaries

in a certain department) are detailed in [STONM74]. Certain anomalies are associated with queries involving aggregates, and some users might consider it a drawback to get answers to queries which are different from those entered into the system (especially if they don't know what queries the answers do match).

As INGRES is implemented on the UNIX operating system, it is afforded physical file protection by UNIX and can take advantage of its tree structured file and directory organization. Extensions have also been designed for implementing INGRES as a distributed database system [STONM76b] in a network of UNIX based systems. A development more important to this report is the adaptation by Mitre of INGRES to run with a secure kernelized UNIX and to incorporate multilevel DoD security policies and controls [WAGNB77].

3.2.4 System R

System R, developed and implemented at IBM's San Jose Research Laboratory, is perhaps the most complete experimental relational database system in the United States [ASTRM76]. In addition to the access control mechanisms, there are also provisions for semantic integrity assertions, logging and recovery, concurrency interlocking at many levels of granularity, and "triggering" of transactions by database events. In System R a table is an existing relation as it resides in the database. A table can also be a "view," or logical variation derived from an existing table. Views can be tailored to the needs and applications of individual users. Authorized users creating new tables (either relations or views) can automatically have access rights to access them and to share these rights with others. The enforcement mechanism simply checks to see if a requesting user has been granted (by someone) the right to perform the requested operation on a given table. The access privilege information is an access list [LANPB71], also stored as a

relation. By this information the user is kept within his/her view. Attempts to write outside this view are protection violations and attempts to read from outside it yield the null response. This enforcement checking is done at access time (not compile time) and is done once per "transaction." A transaction, which can encompass several related database commands, is the unit of protection and integrity locking in System R.

Creators of tables can not only propagate access rights, but they can propagate the right to grant further access rights. Therefore, more than one authorizer can grant access privileges to the same object. This serves a need for decentralized authorization, but complicates the revocation process. Now, to achieve a desired state of authorization, an entire chain of grants might have to be revoked. In [GRIFP76] one particular policy (built into the mechanisms) is described which attempts to return as closely as possible to the authorization state which would exist if the revoked authorization had never been granted. (It is claimed that the whole system returns to its original state, but most likely it was intended to refer to just the state of the authorization information. Obviously, previous database accesses made under the privilege being revoked are irreversible.) In achieving this state all grants propagating from the revoked grant are also revoked, except those "supported" by a previous grant from another (not being revoked) source. Time stamps are used to establish relative timing of grants and revocations. A recursive algorithm, which traverses the graph of grant sequences, is given in [GRIFP76] to accomplish this revocation.

INGRES and System R are further described and compared in [MCLED77]. It is said that the revocation mechanism described above is not implemented in System R.

4. ARCHITECTURAL APPROACHES

4.1 Introduction and Motivation

The functional requirements of section 1, and underlying security mechanisms needed to support these requirements according to the principles of section 2, impose the need for these very general characteristics in a system design: modularity, simplicity, isolatability, and flexibility. It has been observed [MANOF77] that thus far the solutions to data security problems have largely been ad hoc and brute force. A view that can stand back and consider the entire system architecture, rather than concentrating on tuning up individual mechanisms, offers hope for more elegant solutions in the future. Perhaps the single most important conclusion of this report is, not to suggest a solution or a particular system, but to recommend a direction and an approach. The bulk of the evidence and information that went into the making of this report points toward system architecture approaches as the most effective way to satisfy the increasingly complex and often competing requirements. In fact, it appears to be the only approach in which the design of such large systems can be understood and believed to be secure.

In [MARYF78] the separation of DBMS functions from applications programs is argued because then "software that spies on the database cannot be constructed." Claybrook [CLAYB78] shows that the distribution of functions in a DBMS results in:

- a) specialized modules
- b) smaller and simpler modules
- c) small operating system, kernels, and other subsystems
- d) more easily verified security kernels

Modular system architecture approaches, such as the functional specialization in Ohio State's DBC and the functional distribution of Virginia Tech's MULTISAFE, offer--in addition to the advantages listed above:

- a) isolation of protection functions (in accord with the software concept of a security kernel)
- b) integration of many security related functions
- c) elimination of "back doors" (all access paths to the database are identified)
- d) a well structured system

The last point, that of structuring the system, is important. Good structuring provides for a clean design, from the hardware on up. As Jones and Lipton [JONEA75] have put it: "In order to be credible the basic framework must be simple and clear. No one will believe an unstructured system is secure." It is now widely accepted that, although a small amount of clever assembly language code for a function can be very efficient, it is not as reliable and easy to understand as if it were structured and in a higher level language. So it is in systems, especially systems in which security is important and systems that will be maintained over periods of years. As with structured programs, structured systems may suffer small performance penalties compared to "clever" non-structured systems. For example, response time might be higher due to increased communication paths to isolated protection functions. The added overhead must be accepted as part of the cost of doing business properly.

4.2 Architectural Approaches to Database Management

In the late 1960's and early 1970's it was recognized that, since data on secondary storage devices must be moved into the CPU for processing, it would be beneficial for

applications such as DBMS to move some processing capabilities out into secondary devices. The presence of functions for searching, comparing, and combining data made the secondary storage system appear to the CPU as an associative retrieval system. These associative approaches to database searching have been shown to have performance advantages over conventional file and index organizations [DEFIC71, DEFIC73, BERRP74].

Researchers at the University of Florida [COPEG73, SUSYW73] developed a stand-alone, associative, cellular system called CASSM, which is constructed by attaching logic units to a head-per-track, segmented rotating storage device. CASSM was designed to support general database structures such as hierarchies. Data manipulation is accomplished within the secondary memory without the need for control by the CPU. A relational associative processor (RAP) was designed at the University of Toronto [OZKAE75, OZKAE77], and it also is a stand-alone rotating head-per-track scheme for supporting relational databases. RAP followed the idea of cellular architecture as found in CASSM, but with more sophisticated logic for database accessing that improved processing time. RAP can handle very large databases (up to 100 megabits and larger). Later, at the University of Utah [LINCS76] a rotating associative storage device, called RARES, was proposed for relational databases. In RARES search logic is attached to the fixed heads. RARES stores data across adjacent tracks, whereas CASSM and RAP store data along adjacent tracks.

A system called Infoplex [MADNS75] is being designed at MIT to access databases through a hierarchical complex of low-cost microprocessors. This system achieves a high degree of parallelism and pipelining facilitated by the liberal use of queuing between levels of processors in the hierarchy.

One of the earliest associative DBMS was the Information System For Associative Memories (IFAM)--developed for, and partly by, the Rome Air Development Center (RADC) and implemented in 1971 on the Goodyear associative memory. Associative memories, because of economic considerations, must be small (for example, the RADC STARAN uses four arrays of 256 by 256 bytes). The problem, therefore, with using associative memories for database operations has been the time required for loading the associative memory from secondary storage. However, important recent results from RADC [FARND76] have shown the feasibility of a gigabit storage device which eliminates the delays in loading the associative memory, by transferring whole blocks at a time. Based on this new memory technology, RELACS, an associative computer architecture for supporting a relational data model, is currently being designed at Syracuse University [OLIVE79].

In 1974 Canaday et al. [CANAR74] reported a somewhat different approach. Instead of designing special purpose hardware to build into the secondary storage devices, they employed a general purpose minicomputer connected with standard disk drives as a "back-end computer" (BEC). As did the associative devices, the back-end computer accepted DBMS commands from the CPU and returned results without requiring interim processing by the CPU. The following year Kansas State University, in an effort sponsored by AIRMICS [AIRMI78], began to explore distributed processing and the applicability of DBMS to Army problems. KSU researchers investigated the use of BEC's and the application of BEC's to production quality DBMS [MARYF76a, MARYF76b]. Under the joint support of the U. S. Army Computer Systems Command (USACSC), the Naval Ship Research and Development Center (NSRDC), the Naval Material Command Support Activity (NMCSA), and another DoD agency, Cullinane Corporation developed a prototype version of the IDMS database system on a BEC. The work at KSU provided the basic process protocol methodology that was used to develop the inter-computer message system in the Cullinane prototype.

The architectural approaches described in this section are largely oriented toward efficient storage and retrieval in DBMS. Little or no emphasis has been given to security in these systems.

4.3 Architectural Approaches to Secure DBMS

After the separation of DBMS functions from the CPU, as described in the preceding section, it was realized that--in addition to advantages in storage and retrieval--there could be some advantages with respect to security. The kind of isolation and modularization provided by a BEC were ideal for the elimination of "back doors" and other threats to security found in more traditional DBMS. The BEC concept is also highly compatible with the notion of a security kernel. The BEC, in fact, can contain a data security kernel. In Fernandez' system architecture (discussed later in this section) and MULTISAFE (section 4.4.2) the protection functions are even separated from the DBMS, as well as the applications and user programs.

Bisbey and Popek [BISBE74] "encapsulated" the OS and security software on a minicomputer away from the user and application software. It was then easier to certify the system from malicious attacks through user or application software. Downs and Popek [DOWND77] then extended this to the problem of secure database management. Data security modules reside in the nucleus of the DBMS as a data security kernel. The data management module, which maps logical data structures into physical data structures, is supported by a separate "data management kernel." All physical update and retrieval operations are performed by the data management kernel. For aid in updating, a second kernel, called the "kernel input module," is used for obtaining the update data from the user's update request, because the data management

module does not handle actual data. The input data are matched with their corresponding physical structures supplied from the data management module. After security checking, the data management kernel issues a physical I/O request that performs the actual update. For data retrieval, the kernel input module is not involved. The data management kernel validates the retrieval request from the data management module prior to issuing a physical database access.

In an effort to improve security and reduce the sensitivity of compile-time checking to changes in the database and/or the authorization rules, Lang, Fernandez, and Summers [LANGT76] proposed a division of applications software (object programs) into three partitions:

- 1) the non-database application object module (A-program)
- 2) the data interaction object module (D-program)
- 3) the data control object module (C-program)

As the A-program executes, a call is made to the D-program whenever there is a need for database manipulation. The D-program then calls the C-program for all protection checks before responding to the A-program. Neither the A-program nor the C-program can access the database and all security related functions are now concentrated in one place, the C-program. Now changes in data structures or access rules do not require recompilation of the entire applications program. Instead, only the C-program or parts of the D-program must be recompiled. In [FERNE78] a set of architectural extensions to an IBM/370 type machine are proposed to accommodate this new system architecture.

Following the development by Shaefer and Hinke at SDC [HINKT75] of a secure relational data management system for MULTICS, Kirkby and Grohn at I. P. Sharpe [KIRKG77a] describe a reference monitor technique extending [GROHM76] the

Bell and LaPadula model of DoD security policy. Model enhancements include concepts from flow analysis [DENND76]. The DMS kernel is specified with Parnas-type specification techniques [KIRKG77b] and verified to be secure [KIRKG77c]. Protection granularity, however, appears to be only at the relation (file) level. Their work has taken a "bracketing" approach to reference monitoring, in which a front-end processor (software) monitors communication between user terminals and the CPU and a back-end processor (also software) between the CPU and peripheral memory. The front-end and back-end processors are added-on to an off-the-shelf OS. Generality of the approach has been demonstrated to the extent that the concept applies to both MVS/370 and PDP-11 REX. This work is not yet implemented.

The addition of a Data Base Machine (DBM) to WWMCCS has been addressed in a study by Systems Development Corporation (SDC) done for the Defense Communications Agency (DCA) [CADYG78]. The basic objective of the SDC study is to provide improved security of shared WWMCCS databases without any new performance penalties. The DBM approach is used because of the advantages which have been expressed in this present report. Some of the security features proposed appear to be "add on" rather than being integrated into the heart of the WWMCCS design. How well they can eventually be integrated depends largely on how well WWMCCS was originally designed in terms of flexibility for change. Claybrook [CLAYB78] is studying for DCA some aspects of the addition of DBM's to WWMCCS and the effects that such architectural changes might have on security policy.

Cary [CARYJ79] and Hoffman at George Washington University are working on a secure DBMS which isolates database functions across a set of functionally specified hardware. This approach emphasizes single or multipoint real-time surveillance and threat monitoring. This approach appears to be very suitable to implement the functions of a WWMCCS ADP System Security Officer (WASSO) terminal.

There is one new problem introduced by the various architectural approaches to secure DBMS. The isolation of DBMS functions into a BEC or DBM requires careful consideration of the security of messages between the database processor and the main CPU. This is a different issue from that of network communications security and is discussed briefly with respect to MULTISAFE in section 4.4.2.

4.4 Some Experimental Systems

Two experimental systems, the Ohio State University Data Base Computer (DBC) and the Virginia Polytechnic Institute and State University MULTISAFE, are chosen as examples of integrated system architecture approaches to secure database management. Similar work might possibly be underway presently at other places under the wraps of secrecy, but these were the only systems of this type which could be discovered during this study. The work at Ohio State, directed by Professor David Hsiao, and the VPI & SU work employ fundamentally different approaches. The two systems, being at different levels and involving different structures, are complementary rather than competing. The DBC is much more highly developed than the MULTISAFE project and volumes of details (especially in terms of design and implementation) are available from O.S.U. technical reports. Currently, in fact, an experimental software version of the DBC is implemented, and a hardware version is being considered for prototype implementation by a commercial computer manufacturer. MULTISAFE is still very much in the conceptual stages.

4.4.1 The Ohio State Data Base Computer (DBC)

The Ohio State Database Computer (DBC) [BANEJ78] is a functionally specialized architectural approach to high performance, secure data management based on near term future

technology. Using special purpose hardware, it extends and combines the concepts of back-end processors and associative processors. The history of the DBC begins back with Hsiao's attribute-based data model (section 3.1.1) and includes McCauley's security atoms (section 3.1.2), the Highly Secure Data Management System (HSDMS) [HSIAD76a, HSIAD76b], and Baum's [BAUMR75] design of a system architecture. The motivation for specialized hardware is the fact that modules performing DBMS functions need diverse performance capabilities for a balanced high level throughput. This diversity cannot be obtained if all the functions are implemented on the same underlying hardware.

Operating as a BEC with a special OS, the DBC is not constrained to be used with any particular kind of general purpose host computer. In fact, several host computers can share the DBC, possibly in a distributed data environment.

A key design concept within the DBC is the partitioned content addressable memory (PCAM). Large fully associative memories are not feasible under current technology. Instead, the DBC uses a hierarchy of advanced technology to provide numerous smaller associative memories at varying levels of capacity and access speed. A block of the stored data resides in a partition of content addressable memory at some level within this hierarchy.

A schematic diagram of the flow of information and control in the DBC is taken from [BAUMR76] and appears here as figure 1. The operation of the system is divided into two "loops." Common to both loops is the Data Base Command and Control Processor (DBCCP) which serves as an interface with the host computer, controls the operation of both loops, schedules execution of all database commands, and does part of the security checking. Requests from the Program Execution System (PES) enter the DBCCP and are processed by the upper loop ("structure loop") with respect to structural

— Information Path
 - - - Control Path

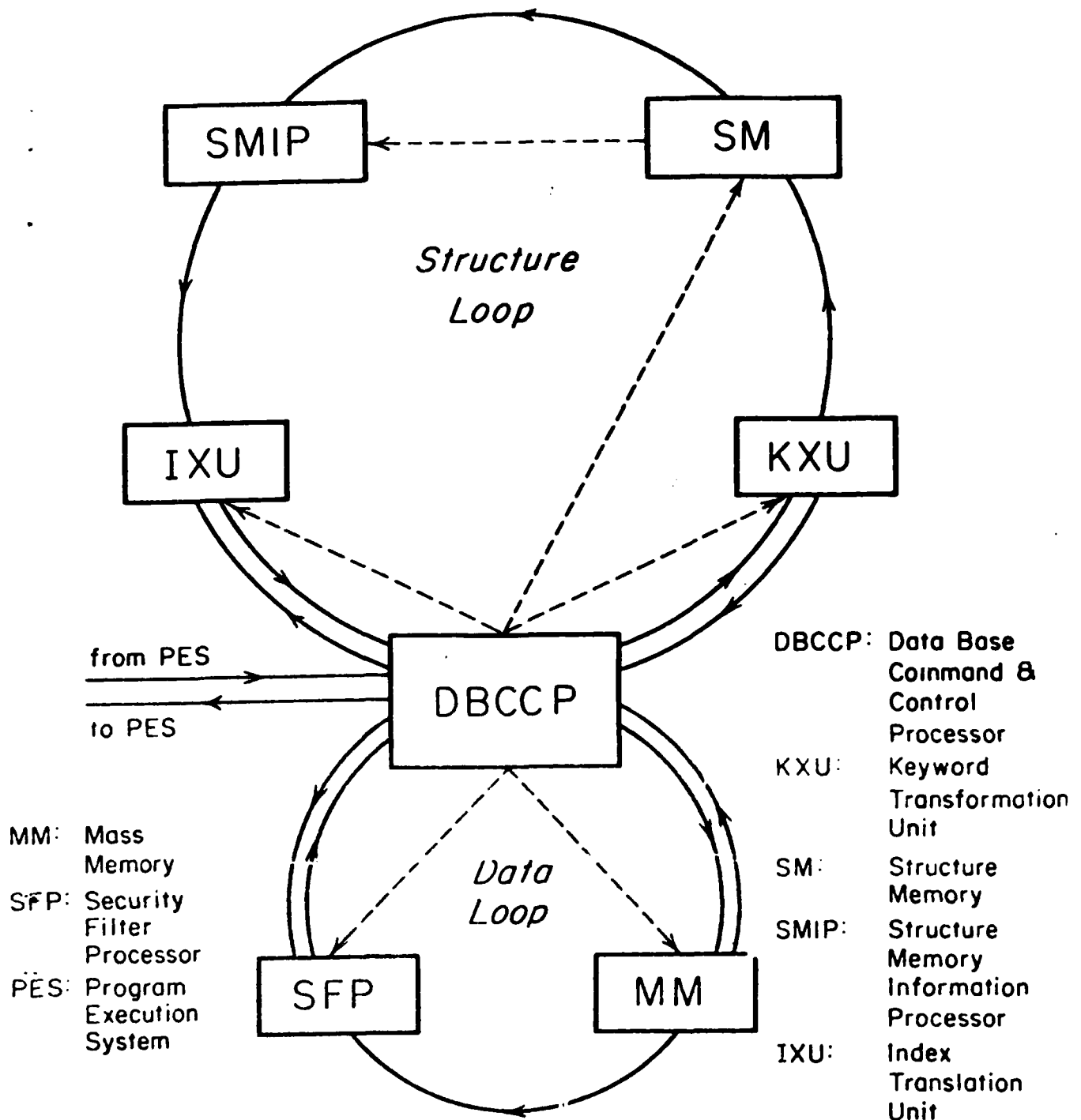


Figure 1. Architecture of DBC
 (Taken from [BAUMR76], by permission)

information [HSIAD76c]. The keyword transformation unit (KXU) converts keywords into the internal forms used by the rest of the system. The structure memory (SM) stores and retrieves the large amount of structural information about the database. PCAM's are used to implement the SM. A "look-aside" buffer maintains high SM performance during updates. The structure memory information processor (SMIP), which also uses PCAM's, performs set operations on structural information from the SM. Then the index translation unit (IXU) decodes and returns to the DBCCP structural information from the SMIP. The four units operate concurrently in a pipeline.

The lower loop (the data loop) is then used for data retrieval (and update). The data loop contains the mass memory (MM), where the database is stored in PCAM's, and the security filter processor (SFP), which does sorting and any security checking beyond that which can be done by security atoms (see section 3.1.2). In the MM a partition of a PCAM is a cylinder of a moving head disk memory. (New advances in associative memory technology would allow for replacing any present PCAM hardware with cheaper, faster, and more reliable devices--such as an array of microprocessors and their memories--without affecting any of the DBC design.) Track information processors (TIP's) provide associative access to each cylinders, and all tracks of a cylinder are processed simultaneously. The MM can search for and retrieve records that satisfy queries. In addition to the security checking based on security atoms, which can be done at directory translation time, security checking can be done using security specifications which are kept by the PES in the form of user capabilities. These security specifications are expressed in the same form as are queries, allowing the full power of the query language to specify records to be protected. The SFP checks retrieved data (but not yet returned to the user) against these specifications.

The DBC has also been shown to be suited as a base system for higher level data models such as the hierarchical [HSIAD77], the network [BANEJ77a], and the relational [BANEJ77b]. Results of a simulation study relating response times, loading conditions, and through-put is available [HSIAD78a].

4.4.2 MULTISAFE

A MULTiprocessor system for supporting Secure Authorization with Full Enforcement (MULTISAFE) for database management is now being developed [TRUER78] by Trueblood and Hartson at VPI & SU. The goals of the new system organization are improvements in performance and security, to be achieved by combining the concepts of multiprocessing, pipelining, and parallelism.

The new system configuration is based on functionally dividing a DBMS into three major modules:

1. the user and application module (UAM)
2. the data storage and retrieval module (SRM)
3. the protection and security module (PSM)

The basic idea is to implement each module on one or more processors forming the multiprocessor system called MULTISAFE. These processors can be as small as microprocessors, or they can be as large as mainframe processors (full sized main frame processors). In a conventional uniprocessor environment these three modules function sequentially in an interleaved fashion. In MULTISAFE all three modules function in a concurrent fashion. That is, the UAM coordinates and analyzes user requests at the same time that the SRM generates responses for requests. Simultaneously, the PSM continuously performs security checks on all activities.

The basic (or minimal) multiprocessor architecture for MULTISAFE is composed of three separate processors which are connected to three separate primary random access memory blocks. Thus, each functional module has its own processor

and primary memory. The system organization follows the multiport-memory organization with private memories [ENSLP77]. A memory is made "private" by connecting only certain processors to it, thereby providing physical separation between the user's memory and the PSM and SRM memories, for example. This separation (or isolation) can significantly improve security because it is physically impossible for a user to access the PSM or the SRM memories. System performance is also enhanced by the concurrent processing.

With this new MULTISAFE approach there are some expected advantages as well as disadvantages. The advantages of the architecture are as follows:

1. better performance—concurrent processing
2. improved security—more powerful mechanisms
3. improved verifiability—isolation of mechanisms
4. modularity—changeability of software

Performance penalties are avoided by overlapping the processing time of the PSM with that of the UAM and SRM. Greater resolution of enforcement can therefore be had without substantially degrading the performance of the overall system. Security is improved because a separate processor has been dedicated for administering the protection policies. The processing paths through the system are controlled by this processor. The protection processor operates concurrently with other processors with the capability of interrupting their processing at any point in time. This capability provides for more flexibility in the security mechanisms—particularly, with regard to the times and places where access decisions can be made. In summary the new system architecture:

1. reduces the possibility of "back doors" or sneak paths, by isolating the protection mechanisms and by forcing all accesses of data to occur through a single path;
2. offers safer failure modes, by showing that failures or penetrations of hardware or software outside of the PSM cannot compromise security; and
3. features a relationship between the operating system and the database system which can protect the data against much of the system's own software.

The verifiability of protection is facilitated by the logical and physical isolation of protection mechanisms from other mechanisms in the system. Not only are protection mechanisms isolated, but the database access mechanisms also are isolated from the user.

Some disadvantages of the proposed architecture are the additional communication connections and the overhead for process interruption and synchronization. However, with improved security and concurrent processing, the cost of the additional communication and processing overhead seems justifiable. This conclusion parallels that of the similar situation in OS kernels, in which added security compensates for a small increase of internal communication.

In contrast to the DBC, which has a highly developed DBMS, limited resources have forced MULTISAFE research to concentrate on just one of the three modules--the PSM. The PSM design is based on a generalized model of database protection (see section 3.1.4), and it employs multiple access decision binding times [HARTH77]. MULTISAFE is an event-driven data-flow system. Thus, all processing is initiated and controlled by events occurring within the message flow, including such events as the transmission of data to and from the database. The flow of messages in MULTISAFE is, therefore, a critical factor. Three approaches are used to study the characteristics of the messages. These approaches involve message structure, message classification, and message sequences. Messages between modules are divided into two parts: a short, fixed length descriptor and a variable length text. The message descriptor is composed of three parts: 1) a message classification code, 2) a message ID, and 3) a message text address.

The security of the descriptor is ensured by putting it in a "locked box." The descriptor is set up as part of an abstract data type. Its contents are set and checked by

protected procedures which are invoked parametrically. No user or user process can directly access message descriptors. The security of the message text is ensured either by PUSHing (depositing the text in the receiver's memory) or by PULLing (retrieving the text from the sender's memory). For example, all texts for the UAM are deposited (PUSHed) in the UAM's memory, because the UAM is not allowed to access the memory of any other modules. On the other hand, all texts for the PSM are retrieved (PULLED) from the sender's memory, because no other modules can write into the PSM's memory. The PUSH/PULL security mechanism is implemented directly in hardware by the private memory structure of the system and is a key mechanism supporting secure inter-module communication.

A message is characterized by five attributes. These attributes are:

- 1) class
- 2) source
- 3) target
- 4) type
- 5) subtype

Messages classes are: request, response, and status. The message type and subtype identify the functional context of the message. Not every combination of values for these attributes is allowable in MULTISAFE. The attributes are used to establish a hierarchy of secure message classifications.

The set of secure message sequences is partitioned into four groups for identification—login, data (access), display (of authorization information), and change (of authorization information) subtypes. Messages from either authorizers or users are subject to two kinds of security checking: 1) checking specific to the request, and 2) system occupancy checking. System occupancy checks relate to overall permission to be an active user of the system, without regard to how the system is being used. The system

occupancy check is always made in conjunction with login. For example, the conditions (separate from user identification) for a given system user may be that occupancy is allowed only between 8:00 a.m. and 5:00 p.m. System occupancy checking at data request time provides an (optional) additional binding time for these conditions.

Formal axioms are used to specify certain properties of safe message classifications and sequences. Formal language grammars are used to describe acceptable sequence structures. Under the combined constraints of the axioms and the grammar rules, it is possible to construct only secure message sequences.

An extension of Petri nets [PETEJ77] is being used to model dataflow within MULTISAFE. The aim is to prove that, given only valid message sequences in the dataflow, no unsafe state can be reached. In addition, there are also relationships between Petri nets and formal language aspects of message sequences.

The single "station" configuration (one UAM, SRM, and PSM) is easily expanded to a distributed environment. Stations now communicate by interstation dataflow. The system remains secure if and only if every station appears (functionally) as a user to each other station. This implies that stations are always connected UAM to UAM. In this way, every request, whether arriving directly from a user or by way of another station, requires an access decision from the PSM in the station to which the request is directed. Certain front-end message handling operations will also require an interface between each station's UAM and the rest of the network. There are some preliminary principles which are currently under investigation. The principles are based on the assumption that data is located at a particular station usually because it was created or entered there, or at least because those who will control its use typically are users local to that station. These principles are:

- a) Authorization for access to data stored at a given station will be done at that station.
- b) Enforcement for access to data stored at a given station will be done at that station.
- c) Messages are to be considered as resources, like data. All message handling will then be governed by access control rules. Both sending and receiving of messages will be subject to enforcement checking.

An interesting side effect of this general approach to communication protection is that access rules can now be used to impose user views of the network configuration.

5. ADDITIONAL ISSUES

5.1 Networks and Distributed Data

Computing is in an early phase of a strong, long-term trend toward networks and distributed databases. Presently, the Datacomputer (Computer Corporation of America) on the ARPANET is one of the few DBMS designed specifically for sharing in a heterogeneous network. However, hardware and software are nearly mature enough for increased activity in this area, and the need to share data already exists [MARYF78]. Therefore, all approaches to future systems considered by the Army ought to take into account the context of networks and distributed systems. In fact, future military databases will be distributed, if for no other reason than to reduce vulnerability and enhance survivability, especially for applications such as Battlefield Automated Systems.

The separation of database management functions into BEC's is a major step toward meeting these new requirements. Now, a database can be shared by many host computers without regard to their physical location. Thus, in a sense a network environment is a natural extension of multiprocessor approaches to secure database management. However, there are additional problems (security and non-security) in networks. A detailed investigation of these other problems is not in the scope of this report, but some of the more important points can be mentioned.

The National Bureau of Standards has been engaged in exploring the issues and problems of network operating systems (NOS) jointly with RADC, some results having been reported by Klableton et al. in [KIMBS76, KIMBS78a]. NOS user and program access controls are proposed at three levels: 1) systems access (occupancy), 2) file access, and 3) database access. NBS is interested in integrated access controls for supporting program access to multiple remote DBMS's, featuring a combination of discretionary and non-discretionary controls.

The subject of authorization and enforcement sites (distributed protection) was addressed for distributed data in section 4.4.2. Another important extension to authorization which could have great significance, especially in military and government networks, is N. Minsky's concept of cooperative authorization [MINSN77]. Important actions often require cooperative authorization; computer networks provide just the required means for such cooperation.

Networks, too, bring out the question of the security of communication among nodes. As this problem is so basically different from internal computer security, it will not be discussed here at all. NSA's COMSEC group is responsible for the security of communications devices used by the government and military. It is also worthwhile to note that

It has been shown that the correctness of data communication protocols cannot be absolutely guaranteed [SUNSC75].

A further problem of military networks, such as AUTODIN II, is that they will eliminate the possibility of having only one level of security classification in a system at one time. In section 2.3.5, a near-term approach to connecting systems of different levels has been described.

A detailed discussion of distributed system topologies can be found in [ENSLP77]. One configuration, the "cluster network," is of special interest here. Beyond the ordinary communication among processors, high bandwidth memory-to-memory connections between processors form "clusters." There are many possibilities for hardware implementation of nodes, including functionally specialized (DBC) hardware and mini- or micro-processors. Mini- or micro-processors can also be used as a cluster monitor for scheduling of workloads and for file allocation. The MULTISAFE system has basically a cluster type of organization with its interconnected memories. The addition to clusters of private memory ports and partitioning of functions over processors has proved to be a very worthwhile approach to secure data management.

5.2 Peripheral Issues

In this section several miscellaneous matters are gathered together to be mentioned briefly. More details of these, and many other similar issues, are addressed in very readable fashion in a text by Hoffman [HOFPL77] and a monograph by Hsiao, Kerr, and Madnick [HSIAD78b].

5.2.1 Abstract Data Types

Abstract data types (ADT) and the concept of type extension is mentioned here because of their increasing importance in high level protection mechanisms. In very simple terms an ADT is a structured data object type and a set of operations which may be performed on an object of that type. No access of any kind can be made to objects of that type except by calling one of these predefined operations. Therefore, ADT's provide "pre-packaged" parametric access to data. Higher level user interfaces (e.g., query languages) are inherently more secure than low level ones (e.g., via calls from programming languages). At lower levels (such as might be used with assembly language programs) the system's internal operations are exposed [MANOF75] through back doors and sneak paths. High level interfaces to data are an important function of ADT's. They control how data is used, even after access control mechanisms have determined to allow access. (See Jones and Liskov [JONEA76, JONEA78] for a description of programming language extensions to include ADT's that control which operations may be performed on objects.) In a secure environment, "navigation" of data is undesirable, browsing is to be discouraged. ADT's will become a standard tool of the secure system designer.

Several protection oriented concepts are related to ADT's. An ADT is like a ring of software surrounding its object. Access to the object can be made only through the "pre-packaged" entry points which are the defined data operations. For example, the rings of Multics are a kind of hardware implementation of ADT's. Software in its inner rings can do sensitive operations such as I/O. An outer ring requiring I/O must ask an inner ring to do it by a call which must go through a pre-defined gate. This is exactly analogous to the parametric use of ADT operations. Having a two state (supervisor/user) situation (as found in OS/370)

is like having only two rings, with the supervisor call (SVC) as the gate. Thus, the supervisor and all of its objects make up a (bloated) ADT.

5.2.2 Risk Assessment and Security Evaluation

Security evaluation is the assessment of effectiveness of security techniques. Hoffman has implemented a system called SECURATE [HOFFL78] to evaluate and analyze security provisions at specific computer installations. SECURATE is based on a consideration of objects, threats, and security features. Using "fuzzy metrics" it helps determine weak and strong points and facilitates the comparison of alternative security designs. Ratings are developed using a "natural" language interaction with the user.

Risk analysis [FIPSP74, COURR75] is the assessment of threats to security, in terms of likelihood of threats and cost of losses due to the manifestation of the threats. In a military environment (as well as in many others), metrics for cost analysis of risk, threat, and protection are very useful tools for supporting budget requests for ADP security measures. Risk management also emphasizes physical facility protection and contingency planning.

5.2.3 Auditing

Auditing involves after-the-fact analysis and is extremely important to detecting system flaws and penetrations not detected by the rest of the security system. Auditing has two aspects: the certification of overall system security and the analysis of transactions which have taken place. The present state-of-the-art of computer security auditing is very far behind the needs. In both military and civilian environments more qualified people and more effective techniques are sorely needed. An NBS workshop was held this November, in which auditing was tied in

with secure data management. Proceedings should be available from Ms. Cella Ruthberg at NBS.

5.2.4 Data Integrity

Data integrity is of two kinds: operational and semantic. Operational integrity implies a consistent state of the database which is achieved by synchronizing concurrent accesses of multiple users to eliminate interference between read and update operations. Although sophisticated solutions exist for operating systems, the problem is more difficult in a database environment, because of the increased granularity [GRAYJ75, RIESD77] required and "phantom" data occurrences due to the use of predicates to define subsets of the data [BAYER76]. The problem becomes even more difficult, as do most timing problems, in networks. Results exist for system level concurrency in distributed databases [ROSED78, BERNP78]. It would appear, however, that the best solutions to interlocking for concurrency are solutions which take a global view of the interaction of deadlocks and recovery as well as concurrency [STONM78]. Crash recovery is especially difficult in a distributed system.

Semantic integrity [ESWAK75, HANMM75] is a matter of correctness of data values and relationships in a database. This correctness can be protected to some extent by user supplied assertions which state constraints on the properties of (values, ranges and data types), and relationships among, data elements. The assertions must remain true after inputs or updates in order to preserve certain semantic relationships between the existing data and the incoming data. A prescribed action is taken in case the assertion is about to be violated. (Of course, semantic integrity checking is limited to obvious errors and cannot detect subtle errors.) There is no reason that semantic integrity needs to be singled out, as it has been in the literature. Integrity assertions are merely data dependent (on existing and

incoming values) access conditions, and any protection system having access conditions can handle integrity constraints as part of its regular business. One interesting different approach uses clustering analysis [LEERC76] in processing an existing database to detect data errors and even estimate values for missing data.

5.2.5 Personnel Problems

The military is well aware of the importance of the security problems of personnel and the physical operating environment. However, this report would be incomplete without at least a reminder of their significance. Physical access remains as an overriding factor of system security. Without protection of the physical operating environment (computer operations, terminals, removable media, etc.), sophisticated internal logical controls will be only a useless expense. There is a story that the great wall of China was breached several times in the first century after it was built—not with direct attacks, but it always began with the bribing of a gatekeeper. There is more computer abuse by authorized personnel than by unauthorized penetrators.

5.3 Advanced Issues

Most of the protection mechanisms described so far control the use of existing data, usually used as inputs to various programs. This is broadly true even in cases where access at the user's level was via a query language. D. Denning [DENND76, DENND77a] has gone beyond this by analyzing the flow of information through executing programs, and deriving the protection level of program outputs. This work addresses the question of what level of classification to assign derived data in order to protect information from flowing from a high security class to a low security class.

The practical integration of secure information flow into existing and proposed protection systems should be a significant goal for the near future. The security of information flow should also be extended to the database environment. To do so in a general way would need to consider predicate (access condition) based discretionary controls as well as the traditional hierarchical levels of non-discretionary military controls (i.e., classification levels). Rather than combining levels within a lattice, access conditions now must be combined logically with Boolean operators. Perhaps, it would be most reasonable to implement secure database information flow on a higher level using the concept of abstract data types. If the exact effects (in terms of information flow) of each ADT operation were specified and verified, calls to the ADT operations (and their input and output parameters) could be used as the basic instructions to analyze for flow, not the individual detailed program instructions within these routines. Information flow at this high level could achieve a savings in overhead without sacrificing security.

In any case, there also still remains a small but important class of operations that do not follow the star property and are not subject to the same kind of information flow analysis. These are the operations which need to write from high levels to objects of lower classification levels. An example is the "sanitizing" of data in order to lower its classification. These operations are difficult to automate and require the system to trust the user. They are, however, becoming more and more important, as increasing attention is being given to the problem of overclassification.

In an even more difficult class of protection problems are those dealing with unauthorized information obtained through inference and deduction--statistical and otherwise [BAUMR74, DOBKD76, CONWR76, KAMJB77, DAVIG78]. A summary of work in this problem area and a list of further references

is given by Denning [DENND78], who concludes that to ensure completely secure statistical databases would require such severe constraints that the utility of the databases would be seriously debilitated.

Mechanisms for protection of derived data are developed by D. Cohen in [COHED77] and are based on the user's access history as a representation of the knowledge acquired by him from the database.

Both information flow and inference controls are discussed in a very readable style by Denning and Denning [DENND77b].

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

The objectives of this study called for an investigation of alternative system architectures for secure DBMS. In reaching that objective, we have discussed backend computers, associative processors, database machines, networks, and distributed systems, as well as multiprocessor approaches. We have looked into security of operating systems and weighed the advantages and problems of security kernels. Several data security models and experimental systems have been reviewed, representing non-architectural and architectural approaches to non-secure and secure DBMS. Conclusions and recommendations have been made along the way as the various approaches were analyzed and evaluated. A few more suggestions are made in the next section with regard to military security policy. The report is then con-

cluded with a summary list of conclusions and recommendations for future research directions.

6.2 Modern Military Computer Security Policies

One objective of this study (see section 1.1) is to help develop more awareness of alternatives to policies and approaches which presently characterize defense oriented agencies dealing with computer security. It is reasonable to look to new technology to improve the effectiveness of data security. This report surveys many technical approaches and solutions. However, it is a strong conclusion of this study that, in addition to following this technical thrust, it may be equally useful for government and military ADP operations to re-evaluate their traditional approach to protection policies. Some examples are suggested in this section.

The first suggestions are about coarse granularity and hierarchical security levels. These two characteristics of modern military data security policy combine to reduce the effectiveness of protection. Historically, most modelling and system design of computer related security (in the military world) has been built around levels of classification, a concept that existed to suit a prior non-computer environment. Broad, fixed security classifications were necessary, because file drawers and other physical repositories of documents each had only one lock. If a person had access to any documents in the drawer, he/she had access to the whole drawer. The application of these same levels to computer data, however, is probably inappropriate. The levels are too broad to effectively resolve "need-to-know" on an individual basis and the security classes are most generally applied to a granularity that is too coarse (e.g., at the file level).* Furthermore, policies based on this hierarchy

of security levels have been built deeply into systems, many times without an appreciation of alternative possibilities. Policies to preserve the star property ("write up" and "read down") are examples. Perhaps in a computer environment, where finer granularity and more individualized need-to-know categories are possible, it might be more useful (i.e., a better model of reality) to have classifications that are not necessarily ordered, but which overlap arbitrarily. With these, there would be no "up" or "down;" instead, access privileges can be tailored to authorized access needs.

As an example, there is a necessity to identify intelligence sources [MANOF77]. This special need should not be lumped together with other different needs. This requirement is not necessarily "above" or "below" other types of requirements in a system; it is just different. There already are some non-hierarchical categories within the military, such as eyes-only, NATO, nuclear, etc., but hierarchical classifications are still dominant.

The effect of the star property and high water mark policies is a rising classification level of files up to the highest level of any item in the file. The result is an over-classification of, and redundancy of, vast amounts of data. It's reasonable to guess that a very large percentage of all military data is overclassified. (Poor granularity is also a cause of overclassification--a granule must be classified at the highest level it contains.) As pointed out strongly by G. Cole [COLEG78], levels of authorization that define a hierarchy of increasing privileges are "not universally accepted as being desirable." He refers to Wulf et al. [WULFW73] "who claim that 'such structures are

* Even now, SDC, in a report for DCA's CCTC (p.3-18 of [CADYG78]), expresses their belief that file level granularity is "state-of-the-art" and that finer granularity would be "breaking new ground." Their references for comparison, however, are other military oriented projects, not the general database security literature. Their claim about increased complexity and performance cost to achieve finer granularity is accurate, though, given present technology.

Inherently wrong and are at the heart of society's concern with computer security.¹⁰ New mechanisms, with less emphasis on security levels will be needed to meet the complex variety of requirements already existing in policies (e.g., access privileges varying depending on system state, data content, access history, statistical constraints, information flow). It would be natural to have a mixture of classifications for many records at different levels, existing together in the same file. Each user would be authorized to access only an appropriate subset. A current direction of interest at the National Bureau of Standards is in the right direction--a combination of discretionary (granted rights) and non-discretionary (labels and security levels) controls. It will be implemented at the data element level, too, thus addressing the companion problem of granularity. Access decisions will be on a record-by-record basis.

Of course, the present state-of-the-art doesn't reliably allow this kind of approach. That is why there are still very few systems which can process more than one classification level at once, and those that do are carefully certified [WALKS77]. However, the technical requirements will be met in time, and new policies need to be ready.

A separate area of concern about military security policy is one which relates to the notions of absolute and relative security. Security guarantees cannot be given for systems today. It is not clear that they ever can be given. To be sure, improved technology and methodology has brought us closer to completely secure systems. It now appears, though, that to get arbitrarily close will come at extremely high cost, with ever more diminishing returns. To some, the term "security" implies a system level guarantee (although it actually is used within this report to mean relative security, not guaranteed security). One area in which security may come close, but not reach perfection is in regard to the notion of security kernels and verification of

software design and implementation (see section 2.2). Another is protection against inference in a statistical database (see section 5.3).

One of the significant questions under the subject of absolute security is: How important is the confinement problem [LAMPB73]? Confinement is the prevention of information leaks through sneak paths, especially as caused by the program which is processing the sensitive data. Confinement has been a deep concern within military oriented work, especially in work concerned with guaranteeing security. The author believes that--contrary to the importance ascribed to it in the literature--confinement is no longer an important issue in a realistic present-day protection system. The issue is, rather, data security. If confinement is an overwhelming concern, the system design will be inordinantly constrained. Overt channels, such as shared files, can be closed off by making the procedures in question memoryless. Most covert channels and Trojan horses can be ferreted out by kernelization and software verification. Perhaps we may have to endure the remaining low bandwidth channels, if any, as part of the cost of doing business. Kimbleton [KIMBS78b] has suggested that a sensitivity analysis might be helpful for a given system, in showing how far it is cost effective to go toward absolute security.

An additional subject, toward which a new viewpoint might benefit both military people and others interested in security, is that of performance overhead (see [HARTH77]). With any function that does not contribute directly to the users' access of resources, there is associated the "necessary evil" of an overhead cost in storage, processing, and I/O. The distinction between the usage of resources considered to be directly serving the user and that considered to be overhead is not always a clear one. The actual minimum software required to access a record of data is relatively small. However, if the addition of software for the operat-

ing system, virtual machine emulation, access methods, the use of higher level languages, backup and recovery, file management, etc. are considered to be only "conveniences," then the resource usage claimed by overhead approaches 100%. As the cost of personnel and software is increasing and the cost of computing power is decreasing, one can conclude that to use computers to automate as many of these functions as possible is economically sound—not to mention greatly more reliable. This is especially true in the military environment where 100% redundancy (of hardware, software, and personnel) is not an unheard of commitment to achieve security for different classification levels. It is likely that protection will soon be considered as an integral part of the system (with respect to performance) and not an add-on overhead.

6.3 Summary of Conclusions and Recommendations

This section recapitulates the important points and conclusions of this study, beginning with a couple of general observations made during the course of the study:

- 1) A large variety of policies, mechanisms, approaches, and system views were encountered among organizations representing commercial software contractors, internal military development, industrial research and development, and academic research. This variety makes it clear that the subject area is still very much developing; there are still more questions than answers—especially answers with proven effectiveness and practicality.
- 2) There is a large amount of money and effort invested in current operating systems and database systems, including their protection subsystems. The inertia of this mass dictates that development in the area of data security will be evolutionary and not revolutionary.

The following conclusions indicate recommended future technical directions in which the author believes research funds may most profitably be expended.

- 1) Architectural and systems approaches to database security—Almost all CPU's today are designed for numeric processing. Database and database security applications need different functions. Specialized hardware is becoming economically feasible and is highly suited for database and database security applications. Architectural building blocks abound—in multiprocessor configurations, associative hardware, block oriented random access memories, minicomputers, microprocessors, and intelligent terminals. The approach is attractive to security applications because of its isolation of functions and the help that modularity gives to system verification. The lead quotation in a paper on computer system organization in the 1980's [APFEN78] is: "Soon, system architects will treat all system components—hardware as well as software, user interfaces as well as databases—as structural or architectural elements."
- 2) Distributed databases and networks—This is the future, especially the future of military data systems. The army must plan now, in order to be there. The architectural approach fits in very well here, too.
- 3) Broader approach to authorization—The Army (and all military systems) needs to go beyond present policies in order to get away from hierarchical levels of security classification. These levels are the cause of many problems, not the least of which is extreme overclassification. Decentralized authorization, discretionary controls, dynamic (on-line) authorization changes, and protection languages for improved interfaces to authorizers are all part of the new approach to authorization.
- 4) Precision and flexibility of enforcement—New security requirements, such as improved auditability and legislated privacy protection will require finer granularity, partial enforcement capability, and access decision

dependency on a large number of system variables (data dependency, time of day, terminal identification, etc.).

- 5) Integration of operating system and database functions--The close relationship among OS and DBMS functions dictates this, for both performance and security reasons: much work remains to be done toward this goal. The days of operating systems with just file handling are numbered. Every OS will have a DBMS. Concepts such as "database operating systems" and "database access methods" for OS are already developing.
- 6) Software kernel methodology--To meet the profound need for increased confidence in kernel techniques, more work is required in the areas of kernel specification, verification, faithful implementation, and testing. Perhaps, the most important problem (and one not receiving the most attention) is that of the partitioning of kernel and non-kernel functions, i.e., dealing with the "spaghetti bowl" syndrome. More treatment is also needed with the secure handling of resulting non-kernel trusted software.
- 7) Additional protection mechanisms--Beyond mechanisms for directly controlling access, there is an increasing need to automate reliably functions such as audit trails, monitoring, surveillance, and exception reporting. The WASSO terminal is a beginning.
- 8) Multiple levels of data in the same system--This should be an immediate goal, and many other recommendations made in this report (such as the suggested changes in security policy) will contribute to its achievement. However, of course, appropriate hardware and software technology to support this requirement is not quite yet generally available.
- 9) Military privacy--Military ADP installations will have an increasing need to protect privacy (as well as security). For example, privacy protection requirements for such functions as personnel and payroll under the Army Standard Systems have been the subject of much recent legislation.

- 10) Advanced problems--Hope for future solutions to the more difficult problems (such as control of information flow or inferential extraction of database information) must be based on increased support to present research in these areas. Personnel identification is another difficult problem that must be solved as an artificial intelligence or pattern recognition problem.
- 11) The role of the Army in database security--There are several general ways in which the Army can increase its role in the advancement of the state-of-the-art in database security.
 - a) Broader interest in database security itself--The treatment of database security in existing ADP security regulations (e.g., [ARMYR77]) is relatively sparse. This is, of course, partly due to the fact that the necessary technology has not arrived. This study itself is evidence of the fact that interest is increasing.
 - b) Increased participation and involvement--An example of such involvement is the U.S. Army Automation Security Workshop, 11-12 December 1978, in Leesburg, Virginia. This workshop is supported by AIRMICS and funded by the USACSC ISRAD program. The workshop will bring about communication between researchers and practitioners in the area, as well as with users and management types. It also will serve to generate some new ideas and help to put existing methodology in perspective. The Army's participation in the KSOS project is another example of how the Army is already expanding its role.
 - c) More sponsored research in computer security--This is, perhaps, an area in which the Army needs to increase its commitment, if it seriously desires to be a significant force (along with the Navy and Air Force, who are already established in the area) in shaping the future state-of-the-art. In this regard, there seems to be a slight trend toward a lessening

of Navy and Air Force spending on security and protection. In addition, there seems to be a move in Air Force emphasis toward prototype demonstrations. Also, sponsored research must not all be heavily mission oriented, either. Some latitude for experimentation with different ideas and directions is necessary to serve the longer term future.

- d) Communication with the other branches--The service branches have been competing, rather than communicating and cooperating in their ADP security efforts. Each one has, to some extent, developed a program that independently parallels the others. Each appears to be solving problems without sharing the solutions. The result is a waste of effort and resources, reinvention of concepts, and a hindrance to future interoperability. In fact, the future requirements for interoperability could conceivably transcend the various service branches and extend to cooperative systems among NATO countries. The army presently has an opportunity to be a moving force in bringing the various groups together. Perhaps, the DoD Consortium can be instrumental in implementing an amalgamation of effort and results.
- e) Avoid confusion of research with development and implementation--By and large, the resources available at universities are limited to conceptual research and very early stage bread-board prototypes. In most universities the emphasis is heavily toward the former. As a rule, an academic department cannot afford to commit large amounts of resources and personnel to producing working systems. (Perhaps Kansas State University and its work on the back-end computer was a notable exception.) Academics can best contribute in very early stages with concepts, ideas, directions, and possibly high level designs. Software firms can then do a better job at serious implementation.

Acknowledgements

The author gratefully acknowledges the support given by the U. S. Army Institute for Research in Management Information and Computer Science (AIRMICS) of the U. S. Army Computer Systems Command (USACSC). Thanks also go to the many people who responded to inquiries by sending technical reports and other information. Special thanks are due to Stephen R. Kimbleton (National Bureau of Standards), David Cohen (Bell Telephone Laboratories), Billy Claybrook (University of Connecticut, on leave from VPI & SU), E. J. McCauley (Ford Aerospace), Stephen Walker (CCCI of the Under Secretary of Defense for Research and Engineering), and LTC Robert P. Campbell (U. S. Army, DAMI-ANP, Pentagon) for discussions that helped to formulate the conclusions and recommendations. Appreciation is also expressed to Professor David K. Hsiao for information about the Data Base Computer and Robert P. Trueblood for help with the description of MULTISAFE.

Cited References

- AIPN178 "Report on Development of a Back-End Data Base Management System," Draft of Technical Report by the U. S. Army Institute for Research in Management Information and Computer Science (AIRMICS) (October 1978).
- AMBLA77 Ambler, Allen L., et al., "GYPSY: A Language for Specification and Implementation of Verifiable Programs," Proc. of the ACM Conf. on Language Design for Reliable Software, Raleigh, NC (March 1977), I-10.
- ANDEJ72 Anderson, J. P., "Computer Security Technology Planning Study," ESD-TR-73-51, Vols. 1 and 2, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, Mass. (October 1972).
- APFEH78 Apfelbaum, Henry, et al., "Computer System Organization: Problems of the 1980's," IEEE Computer (September 1978), 20-28.
- ARMYR77 Army Regulation #380-380, "Automated System Security," Headquarters, Department of the Army (December 1977).
- ASTRM76 Astrahan, M. M., et al., "System R: Relational Approach to Database Management," ACM Trans. on Database Systems 1, 2 (June 1976), 97-137.
- BANEJ77a Banerjee, Jayanta, David K. Hsiao, and Douglas S. Kerr, "DBC Software Requirements for Supporting Network Databases," Technical Report OSU-CISRC-72-77-4, Department of C. I. S., Ohio State University (June 1977).
- BANEJ77b Banerjee, Jayanta, and David K. Hsiao, "DBC Software Requirements for Supporting Relational Databases," Technical Report OSU-CISRC-TR-77-7, Department of C. I. S., Ohio State University (November 1977).
- BANEJ78 Banerjee, Jayanta, Richard I. Baum, and David K. Hsiao, "Concepts and Capabilities of a Database Computer," ACM Trans. on Database Systems 3, 4 (December 1978), 347-384.
- BAUMR74 Baum, Richard I., and David K. Hsiao, "A Data Secure Computer Architecture (Part I)," Technical Report OSU-CISRC-TR-73-10, Department of C. I. S., Ohio State University (July 1974).
- BAUMR75 Baum, Richard I., "The Architectural Design of a Secure Database Management System," Ph.D. dissertation, Ohio State University, Technical Report OSU-CISRC-TR-75-8 (1975).
- BAUMR76 Baum, Richard I., David K. Hsiao, and K. Kannan, "The Architecture of a DataBase Computer (DBC): Part I: Concepts and Capabilities," Technical Report OSU-CISRC-TR-76-1, Department of C. I. S., Ohio State University (September 1976).
- BAYER76 Bayer, Rudolf, "On the Integrity of Data Bases and Resource Locking," in G. Goos and J. Hartmanis (eds.), Lecture Notes in Computer Science, Vol. 39: Data Base Systems (Proc., 5th Informatik Symp. IBM Germany, Bad Homburg v.d.H.) (September 1975), Springer-Verlag, Berlin, Heidelberg, and New York (1976), 339-361.

- BELLD73 Bell, D. E., and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model, Vols. I, II, and III, Mitre Corp., Bedford, Mass. (November 1973-June 1974).
- BERNP78 Bernstein, P. A., J. B. Rothnie, N. Goodman, and C. A. Papadimitriou, "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," IEEE Trans. on Software Engineering SE-4, 3 (May 1978), 154-168.
- BERRP74 Berra, P. Bruce, "Some Problems in Associative Processor Applications to Data Base Management," Proc. of the AFIPS NCC (1974), 1-5.
- BISBR74 Bisbey, Richard L., II, and Gerald J. Popek, "Encapsulation: An Approach to Operating System Security," Proc. of the ACM Annual Conf. San Diego (November 1974), 666-675.
- CADYG78 Cady, George, et al., "WWMCCS Data Management Analysis and Data Base Machine Requirements Definition and Functional Description," TM-WD-79.1/000/00, A working paper by Systems Development Corp., McLean, Va., for CCTC of the Defense Communications Agency (July 1978).
- CANAR74 Canaday, R. H., R. D. Harrison, E. L. Ivie, J. L. Ryder, and L. A. Wehr, "A Back-End Computer for Data Base Management," Comm. of the ACM 17, 10 (October 1974), 575-582.
- CARYJ78 Cary, John M., "A Distributed Architecture Security System for Centralized and Distributed Data Base Systems," Ph. D. dissertation in progress, George Washington University.
- CLAYB78 Claybrook, Billy G., "A Study of Architectural and Security Issues Associated With Integrating Database Computers into WWMCCS," Technical Report to be published by the Defense Communications Agency, Reston, Va. 22090.
- COHED77 Cohen, David, "Design of Event-Driven Protection Mechanisms," Ph.D. dissertation, Department of Computer and Information Science, The Ohio State University (1977).
- CCLEG78 Cole, Gerald D., "Design Alternatives for Computer Network Security," NBS Special Publication 500-21, Vol. 1, National Bureau of Standards, Washington, DC 20234.
- CONWR72 Conway, Richard, William Maxwell, and Howard Morgan, "Selective Capabilities in ASAP--A File Management System," Proc. of the SJCC (1972), 1181-1185.
- CONWR76 Conway, Richard, and David Strip, "Selective Partial Access to a Database," Proc. of the ACM Annual Conf., Houston (October 1976), 85-89.
- COPEG73 Copeland, George P., Jr., G. J. Lipovsky, and Stanley Y. W. Su, "The Architecture of CASSM: A Cellular System for Non-Numeric Processing," Proc. of the First Annual Symp. on Computer Architecture, (December 9-11, 1973), 121-128.

- COURR75 Courtney, Robert H., Jr., "Security Risk Assessment in Electronic Data Processing Systems," Working document of IFIP-TG-15, available from IBM Corp., P. O. Box 390, Poughkeepsie, NY 12602 (December 1975).
- DAVIG78 Davida, George, et al., "Database Security," IEEE Trans. on Software Engineering, SE-4, 6 (November 1978), 531-533.
- DEFIC71 DeFlora, Casper, Neil Stillman, and P. Bruce Berra, "Associative Techniques in the Solution of Data Management Problems," Proc. of the ACM National Conf. (1971), 28-36.
- DEFIC73 DeFlora, Casper, and P. Bruce Berra, "A Data Management System Utilizing an Associative Memory," Proc. of the AFIPS NCC (1973), 181-185.
- DENND76 Denning, Dorothy E., "A Lattice Model of Secure Information Flow," Comm. of the ACM 19, 5 (May 1976), 236-243.
- DENND77a Denning, Dorothy E., and Peter J. Denning, "Certification of Programs for Secure Information Flow," Comm. of the ACM 20, 7 (July 1977), 504-513.
- DENND77b Denning, Dorothy E., and Peter J. Denning, "The Limits of Data Security," mock-up issue of Abacus published by AFIPS, vol. 0 no. 0 (June 1977), 22-30.
- DENND78 Denning, Dorothy E., "Are Statistical Data Bases Secure?" Proc. of the AFIPS NCC (1978), 525-530.
- DOBKD76 Dobkin, David, Anita K. Jones, and Richard J. Lip-ton, "Secure Data Bases: Protection Against User Inference," Research Report #65, Department of C. S., Yale University (April 1976).
- DOWND77 Downs, Deborah, and Gerald J. Popek, "A Kernel Design for a Secure Database Management System," Proc. of the 3rd International Conf. on Very Large Data Bases (1977).
- ENSLP77 Enslow, Phillip H., Jr., "Multiprocessor Organization--A Survey," ACM Computing Surveys 9, 1 (March 1977), 103-129.
- ESWAK75 Eswaran, Kapali P., and Donald D. Chamberlin, "Functional Specifications of a Subsystem for Data Base Integrity," Proc. of the International Conf. on Very Large Data Bases, Framingham, Mass. (September 1975), 48-68.
- FARND76 Farnsworth, D. L., C. P. Hoffman, and J. J. Schutt, "Mass Memory Organization Study," Rome Air Development Center Technical Report RADC-TR-76-254 (September 1976).
- FERNE75a Fernandez, Eduardo B., Rita C. Summers, and Charles D. Coleman, "An Authorization Model for a Shared Data Base," Proc. ACM-SIGMOD International Conf. on Management of Data San Jose (May 1975), 23-31.
- FERNE75b Fernandez, Eduardo B., Rita C. Summers, and Tomas Lang, "Definition and Evaluation of Access Rules in Data Management Systems," Proc. of the International Conf. on Very Large Data Bases, Framingham, Mass. (September 1975), 268-285.

- FERNE77 Fernandez, E. B., and Wood, Christopher, "The Relationship Between Operating System and Database System Security: A Survey," Proc. of the IEEE Computer and Software Applications Conf. Chicago (November 1977), 453-462.
- FERNE78 Fernandez, Eduardo B., Rita C. Summers, Tomas Lang, and Charles D. Coleman, "Architectural Support for System Protection and Database Security," IEEE Trans. on Computers C-27, 8 (August 78), 767-771.
- FIPSP74 FIPS Pub. 31, U. S. Department of Commerce, National Bureau of Standards (June 1974).
- GILSJ75 Gilson, John, and John Mekota, "Analysis of Secure Communications Processor Architecture," Honeywell Information Systems, Inc., Federal Systems Operations, McLean, Va. 22101 (November 1975). Available as NTIS AD-A055 164/8WC.
- GOLDB77 Gold, B., R. Linde, M. Shaefer, J. Scheld, "VM 370 Security Retrofit Program," Proc. of the ACM Conf. (October 1977), 411-417.
- GRAYJ75 Gray, J. N., E. A. Lorie, and G. R. Putzolu, "Granularity of Locks in a Shared Data Base," Proc. of the International Conf. on Very Large Data Bases, Framingham, Mass. (September 1975), 428-436.
- GRIPP76 Griffiths, Patricia P., and Bradford W. Wade, "An Authorization Mechanism for a Relational Database System," ACM Trans. on Database Systems 1, 3 (September 1976), 242-255.
- GROHM76 Grohn, Michael, "A Model of a Protected Data Management System," ESD-TR-76-288, I. P. Sharpe Associates Ltd., Ottawa, Canada (June 1976).
- HAMMM75 Hammer, Michael M., and Dennis J. McLeod, "Semantic Integrity in a Relational Data Base System," Proc. of the International Conf. on Very Large Data Bases, Framingham, Mass. (September 1975), 25-47.
- HARRM75 Harrison, Michael A., Walter L. Ruzzo, and Jeffrey D. Ullman, "On Protection in Operating Systems," ACM Operating Systems Review, 9, 5 (November 1975), 14-24.
- HARTH75 Hartson, H. Rex, "Languages for Specifying Protection Requirements in Data Base Systems--A Semantic Model," Ph.D. Dissertation, Dept. of Computer and Information Science, The Ohio State University (August 1975), Research report: OSU-CISEC-TR-75-6.
- HARTH76a Hartson, H. Rex, and David K. Hsiao, "A Semantic Model for Data Base Protection Languages," Proc. of the International Conf. on Very Large Data Bases Brussels (September 1976), 27-42.
- HARTH76b Hartson, H. Rex, and David K. Hsiao, "Full Protection Specifications in the Semantic Model for Database Protection Languages," Proc. of the Annual Conf. of the ACM Houston (October, 1976), 80-85.
- HARTH77 Hartson, H. Rex, "Dynamics of Database Protection Enforcement--A Preliminary Study," Proc. of the IEEE Computer and Software Applications Conf. Chicago (November 1977), 349-356.

- HINKT75 Hinke, Thomas H., and Marvin Shaefer, "Secure Data Management System," RADC-TR-75-266, Systems Development Corp., Santa Monica, Calif. (November 1975).
- HOPFL77 Hoffman, Lance J., Modern Methods for Computer Security and Privacy, Prentice-Hall, Englewood Cliffs, NJ (1977).
- HOFFL78 Hoffman, Lance J., Eric H. Michelman, and Don Clements, "SECURATE—Security Evaluation and Analysis Using Fuzzy Metrics," Proc. of the AFIPS MCC (1978), 531-540.
- HONEY76 "Multics Security Kernel Certification Plan," Honeywell Information Systems, Inc., Federal Systems Operations, McLean, Va. 22101 (July 1976). Also available from NTIS as AD-A055 171/3WC.
- HSIAD68 Hsiao, David K., "A File System for a Problem Solving Facility," Ph.D. dissertation, Moore School of Electrical Engineering, University of Pennsylvania (May 1968).
- HSIAD70 Hsiao, David K., and Frank Harary, "A Formal System for Information Retrieval from Files," Comm. of the ACM 13, 2 (February 1970), 67-73.
- HSIAD76a Hsiao, David K., "A Software Engineering Experience in the Management, Design, and Implementation of a Data Secure System," Proc. of the ACM 2nd International Conf. on Software Engineering, San Francisco (1976), 532-538.
- HSIAD76b Hsiao, David K., and Richard I. Baum, "Information Secure Systems," Advances in Computers, Vol. 14, Academic Press (1967), 231-272.
- HSIAD76c Hsiao, David K., and K. Kannan, "The Architecture of a Database Computer; Part II: The Design of Structure Memory and Its Related Processors," Technical Report OSU-CISRC-TR-76-2, Department of C. I. S., Ohio State University (October 1976).
- HSIAD77 Hsiao, David K., Douglas S. Kerr, and Fred K. Ng, "DBC Software Requirements for Supporting Hierarchical Databases," Technical Report OSU-CISRC-TR-77-1, Department of C. I. S., Ohio State University (April 1977).
- HSIAD78a Hsiao, David K., and K. Kannan, "Simulation Studies of the Database Computer (DBC)," Technical Report OSU-CISRC-TR-78-1, Department of C. I. S., Ohio State University (February 1978).
- HSIAD78b Hsiao, David K., Douglas S. Kerr, and Stuart E. Madnick, Computer Security Problems and Solutions, monograph, Dept. of Computer and Information Science, The Ohio State University (1978).
- JONEA75 Jones, Anita K., and Richard J. Lipton, "The Enforcement of Security Policies for Computation," Operating Systems Review 9, 5 (November 1975), 197-206.
- JONEA76 Jones, Anita K., and Barbara H. Liskov, "A Language Extension for Controlling Access to Shared Data," IEEE Trans. on Software Engineering SE-2, 4 (December 1976), 277-285.

- JONEA78 Jones, Anita K., and Barbara H. Linkov, "A Language Extension for Expressing Constraints on Data Access," Comm. of the ACM 21, 5 (May 1978), 358-367.
- KANJB77 Kam, John R., Jeffrey D. Ullman, "A Model of Statistical Databases and Their Security," ACM Trans. on Database Systems 2, 1 (March 1977), 1-10.
- KIMBS76 Kimbleton, Stephen R., and R. L. Mandell, "A Perspective on Network Operating Systems," Proc. of the AFIPS NCC (1976), 551-559.
- KIMBS78a Kimbleton, Stephen R., Helen M. Wood, and M. L. Fitzgerald, "Network Operating Systems--An Implementation Approach," Proc. of the AFIPS NCC (1978), 773-782.
- KIMBS78b Kimbleton, Stephen R., National Bureau of Standards, personal communication.
- KIRKG77a Kirkby, Gillian, and Michael Grohn, "The Reference Monitor Technique for Security in Data Management Systems," IEEE Data Base Engineering 1, 2 (June 1977), 8-15.
- KIRKG77b Kirkby, Gillian, and Michael Grohn, "On Specifying the Functional Design for a Protected DMS Tool," ESD-TR-77-140, I. P. Sharpe Associates Ltd., Ottawa, Canada (April 1977).
- KIRKG77c Kirkby, Gillian, and Michael Grohn, "Validation of the Protected DMS Specifications," ESD-TR-77-141, I. P. Sharpe Associates Ltd., Ottawa, Canada (April 1977).
- LANGT76 Lang, Tomas, Eduardo B. Fernandez, Rita C. Summers, "A System Architecture for Compile-Time Actions in Databases," IBM Los Angeles Scientific Center, Report No. G320-2682 (December 1976).
- LINCS76 Lin, C. S., D. C. P. Smith, and J. N. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications," ACM Trans. on Database Systems 1, 1 (March 1976), 53-65.
- LAMPB71 Lampson, Butler W., "Protection," Proc. Fifth Princeton Symp. on Information Sciences and Systems, Princeton University (March 1971), 437-443; reprinted in ACM SIGOPS Operating Systems Review 8, 1 (January 1974), 18-24.
- LAMPB73 Lampson, Butler W., "A Note on the Confinement Problem," Comm. of the ACM 16, 10 (October 1973), 613-615.
- LEERC76 Lee, R. C. T., J. R. Slagle, and C. I. Mong, "Application of Clustering to Estimate Missing Data and Improve Data Integrity," Proc. of the 2nd International Conf. on Software Engineering, San Francisco (October 1976).
- MADNS75 Madnick, Stuart E., "INFOPLEX--Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex," Proc. of the AFIPS NCC (1975), 581-586.

- MANOF75 Manola, Frank, and Stanley Wilson, "Data Security Implications of an Extended Subschema Concept," Proc. of the Second USA-Japan Computer Conf., Tokyo (August 1975), 481-487. (Also available as NRL Report 7905, July 1975.)
- MANOF77 Manola, Frank, and David K. Helao, "An Experiment in Database Access Control," Proc. of the Computer Software and Applications Conf., Chicago (1977), 357-363. (A slightly more complete version is available as NRL Report 8176 (March 1978) from the Naval Research Laboratory, Washington, D.C. 20375.)
- MARYF76a Maryanski, Fred J., P. S. Fisher, and V. E. Wallentine, "An Evaluation of a Conversion to a Back-End Data Base Management System," Proc. of the Annual ACM Conf. (October 1976), 293-297.
- MARYF76b Maryanski, Fred J., V. E. Wallentine, and P. S. Fisher, "A User-Transparent Mechanism for the Distribution of a CODASYL Data Base Management System," TR-CS-76-22, Department of C. S., Kansas State University, Manhattan, Kansas (December 1976).
- MARYF78 Maryanski, Fred J., "A Survey of Developments in Distributed Data Base Management Systems," IEEE Computer (February 1978), 28-38.
- MCCAE75 McCauley, E. J., III, "A Model for Data Secure Systems," Ph.D. dissertation, Department of C. I. S., Ohio State University (1975).
- MCLED77 McLeod, Dennis, "A Framework for Data Base Protection and Its Application to the INGRES and System R Data Base Management Systems," Proc. of the IEEE Computer and Software Applications Conf. (COMPSAC), Chicago (November 1977), 342-348.
- MILLJ76 Millen, Jonathan K., "Security Kernel Validation in Practice," Comm. of the ACM 19, 5 (May 1976), 243-250.
- MINSN77 Minsky, Naftaly, "Cooperative Authorization in Computer Systems," Proc. of the IEEE Computer and Software Applications Conf. (COMPSAC), Chicago (November 1977), 729-733.
- NEUMP77 Neumann, Peter G., R. S. Boyer, R. J. Feiertag, K. N. Levitt, and L. Robinson, "A Provably Secure Operating System: The System, Its Applications, and Proofs," Final Report, Project 4332, SRI International, Menlo Park, Calif. 94025 (February 1977).
- NEUMP78 Neumann, Peter G., "Computer System Security Evaluation," Proc of the AFIPS NCC (1978), 1087-1095.
- OLIVE79 Oliver, Ellen, "RELACS, An Associative Computer Architecture to Support a Relational Data Model," unpublished Ph.D. dissertation, Syracuse University.
- OZKAE75 Ozkaran, E. A., S. A. Schuster, and K. C. Smith, "RAP -- An Associative Processor for Data Base Management," AFIPS National Computer Conference, Vol. 44, (May 19-22, 1975), 378-387.

- OZKAE77 Ozkaran, E. A., S. A. Schuster, and K. C. Sevcik, "Performance Evaluation of a Relational Associative Processor," ACM Transactions on Database Systems, Vol. 2, No. 2, (June 1977), 175-195.
- PETEJ77 Peterson, James L., "Petri Nets," ACM Computing Surveys, 9, 3 (September 1977), 223-252.
- POPEG78a Popek, Gerald J., and Charles S. Kline, "Issues in Kernel Design," Proc. of the AFIPS NCC (1978), 1079-1086.
- POPEG78b Popek, Gerald J., and David A. Farber, "A Model for Verification of Data Security in Operating Systems," Comm. of the ACM 21, 9 (September 1978), 739-749.
- RIESD77 Ries, Daniel R., and Michael Stonebraker, "Effects of Locking Granularity in a Database," ACM Trans. on Database Systems 2, 3 (September 1977), 233-246.
- ROBIL77 Robinson, L., K. N. Levitt, P. G. Neumann, and A. K. Saxena, "A Formal Methodology for the Design of Operating System Software," in R. T. Yeh (ed.), Current Trends in Programming Methodology, Volume 1: Software Specifications and Design, Prentice-Hall, Englewood Cliffs, N.J. (1977), 61-110.
- ROSED78 Rosencrantz, David J., Richard E. Stearns, and Philip M. Lewis II, ACM Trans. on Database Systems 3, 2 (June 1978), 178-198.
- SALTJ74 Saltzer, Jerome H., "Protection and the Control of Information Sharing in MULTICS," Comm. of the ACM 17, 7 (July 1974), 388-402.
- SALTJ75 Saltzer, Jerome H., and Michael D. Schroeder, "The Protection of Information in Computer Systems," Proc. of the IEEE 63, 9 (September 1975), 1278-1308.
- SCHRM77 Schroeder, M. D., D. D. Clark, and J. H. Saltzer, "The Multics Kernel Design Project," Proc. of the Sixth Symp. on Operating System Principles, ACM SIGOPS Operating System Review 11, 5 (November 1977), 43-56.
- SPITJ78 Spitzer, Jay M., Karl N. Levitt, and Lawrence Robinson, "An Example of Hierarchical Design and Proof," Comm. of the ACM 21, 12 (December 1978), 1064-1075.
- STONM74 Stonebraker, Michael, and Eugene Wong, "Access Control in a Relational Data Base Management System by Query Modification," Proc. of the ACM Annual Conf. San Diego (November 1974), 180-186.
- STONM76a Stonebraker, Michael, Eugene Wong, Peter Kreps, and Gerald Held, "The Design and Implementation of INGRES," ACM Trans. on Database Systems 1, 3 (September 1976), 189-222.
- STONM76b Stonebraker, Michael, "A Distributed Data Base Version of INGRES," Memorandum No. ERL-M612, Electronic Research Laboratory, University of Calif., Berkeley (September 1976).

- STONV78 Stonebraker, Michael, "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES," Proc. of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks (August 1978), 235-258.
- SUNMR77 Summers, R. C., and E. B. Fernandez, "A System Structure for Data Security," IBM Los Angeles Scientific Center, Technical Report G320-2687 (May 1977).
- SUNSC75 Sunshine, C., "Issues in Communication Protocol Design—Formal Correctness," Reprints of ACM Inter-process Communication Workshop (March 1975), p. 118.
- SUSYW73 Su, S. Y. W., G. P. Copeland, Jr., and G. J. Lipovski, "Retrieval Operations and Data Representations in a Content-Addressable Disk System," Proc. of the ACM SIGPLAN, SIGIR Interface Meeting (November 1973), 144-160.
- TANEA76 Tanenbaum, Andrew S., "In Defense of Program Testing, or Correctness Proofs Considered Harmful," ACM SIGPLAN Notices (May 1976), 64-68.
- TRUER78 Trueblood, Robert P., and H. Rex Hartson, "A Working Paper on the Development of Multiprocessor Architectures for Supporting Secure Database Management," Technical Report CS78007-R, Department of C. S., Virginia Polytechnic Institute and State University, Blacksburg, Va. 24061 (September 1978).
- WAGNB77 Wagner, B. N., "Implementation of a Secure Data Management System for the Secure UNIX Operating System," Mitre Corp. Technical Report for the Air Force ESD, ESD-TR-78-154 (September 1977).
- WALKS77 Walker, Stephen T., "A Certified Multilevel Secure Minicomputer Operating System," SIGNAL 32 (November 1977), 37-39.
- WELDJ76 Weldon, Jay-Louise, and Shankant B. Navathe, "An Attribute-based File Organization for a Relational Database," Proc. of the ACM Annual Conf., Houston (1976), 319-323.
- WEISC69 Weissman Clark, "Security Controls in the ADEPT-50 Time Sharing System," Proc. of the FJCC 35 (1969), 119-133.
- WULFW73 Wulf, W., et al., "HYDRA: The Kernel of a Multiprocessor Operating System," Carnegie-Mellon University, AD 762 514 (June 1973).
- YAOSB76 Yao, S. Bing, "Modeling and Performance Evaluation of Physical Database Structures," Proc. of the ACM Annual Conf., Houston (1976), 303-309.

