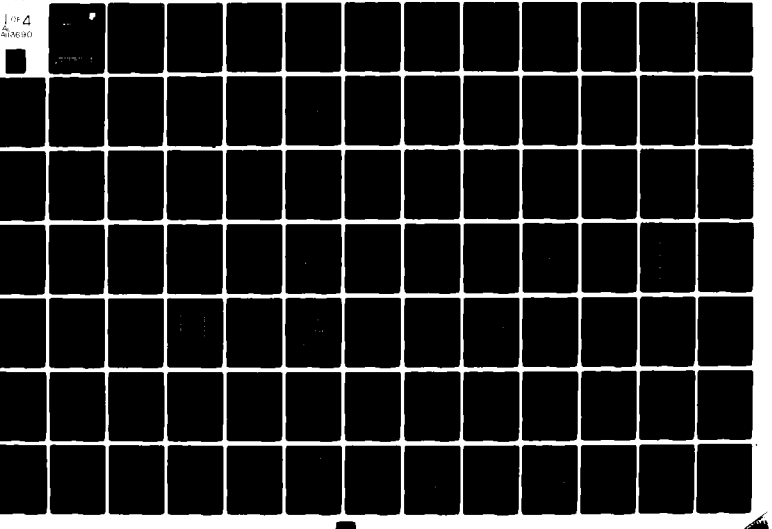
AD-A113 690 HARRIS CORP MELBOURNE FL GOVERNMENT ELECTRONIC SYSTE--ETC F/G 9/2
SECURE DBMS.(U)
FEB 82 T D WORMINGTON, C E GIESLER F30602-80-C-0235
UNCLASSIFIED RADC-TR-81-304 NL



(12)

RADC-TR-81-394
Final Technical Report
February 1982

AD A113690

SECURE DBMS

HARRIS CORPORATION

T.D. Wormington
C.E. Giesler

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DTIC
ELECTE
APR 19 1982
E

82 04 19 001

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-394 has been reviewed and is approved for publication.

APPROVED: Sandra L. Posma

SANDRA L. POSMA
Project Engineer

APPROVED: Alan R Barnum

ALAN R. BARNUM, Ass't Chief
Command and Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COEA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-394	2. GOVT ACCESSION NO. AD-A113 694	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SECURE DBMS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report July 1980 - July 1981
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) T.D. Wormington C.E. Giesler		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0235
9. PERFORMING ORGANIZATION NAME AND ADDRESS HARRIS CORPORATION Government Electronic Systems Division Melbourne FL 32901		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55812131
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEA) Griffiss AFB NY 13441		12. REPORT DATE February 1981 13. NUMBER OF PAGES 330
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Thomas C. Darr, Maj, USAF

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multi-Level Security
Distributed Computer Hardware
Secure Data Bases
Computer Security
Static Encryption
Security Filter
DoD Security Model

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This study effort evaluates the feasibility of employing a distributed computer system architecture to support the secure data base management activities of the Air Force. Various distributed system architecture and the means of implementing the required security enforcing mechanisms are described. The basic approach places a security filter between a group of independent single-level user data base management processors and a common shared, multi-level data base. This security filter is capable of (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

enforcing a different DoD non-discretionary security policy (consisting of read and write access controls, security classification and compartment) for each DBMS processor through provable hardware means. In addition, the security filter can either provide or support enforcement of discretionary security (need-to-know) and integrity protection (data Quality) through software (or firmware) external to the DBMS processors. This approach isolates such security related software from user control. Consequently, there is less need for software certification; that is, trusted software may be adequate for discretionary and integrity security.

The study concludes that the use of a distributed architecture does make it feasible to provide provable multi-level security for data base operations. Such an approach offers many potential advantages. In particular, the security filter can be designed so as to appear transparent to the DBMS processors, thus permitting the protection of a multi-level data base while using commercial processors and non-certified DBMS operating systems. This transparency to the DBMS permits retrofit upgrades to existing data base systems as well as development of future systems. Finally, the relatively low development cost (compared to the development and certification costs of secure operating systems), the great flexibility, and the total provability of the security enforcement mechanisms provided by the described architectures make this a desirable as well as feasible approach to computer multi-level security.

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

DTIC COPY INSPECTED 2

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF FIGURES (CONT'D)

LIST OF FIGURES (CONT'D)

1.0 INTRODUCTION

This report presents the results of a study performed for the Rome Air Development Center under Contract F30602-80-C-0235. The study was performed by the staff of the Harris Corporation Government Electronic Systems Division for one year, terminating 16 July 1981. This report documents a feasibility study of the use of a distributed architecture to meet multi-level security requirements for secure data base management. The described architectures are capable of enforcing DoD non-discretionary and discretionary security and an integrity (correctness) policy. (Figure 2.1-3 defines classical secure processing modes and terminology). Included in this effort were: a math model of a secure multi-level relational DBMS, a locally distributed implementation study, functional descriptions and sizing of the resulting systems, and a benefit trade-off analysis of the architectural approach.

1.1 Approach

Conceptually, the architectural approach is to place a hardware security filter (HSF) between numerous Data Base Management System (DBMS) processors and a central data base. This HSF is capable of enforcing a separate access policy for each processor as shown in Figure 1.1-1. By dedicating each DBMS processor to supporting a group of users at a common access category (eg. TS/CNWDI, S/CRYPTO, etc) with appropriate nondiscretionary access control enforced externally by the HSF, a multi-level mode of operation can be achieved without a secure operating system (OS) or trusted software.

The HSF acts as a central data base entrance and exit guard, separating data by security category and controlling which categories may be stored and/or retrieved by each DBMS processor. Four separation techniques were considered: multi-key static encryption, physical security tags, central data base file separation (address tags), and directory tags. In addition, the following concepts were studied: several multi-level relational DBMS approaches, discretionary security control requirements, integrity control requirements, and audit and security monitoring.

1.2 Conclusions

Many implementations of this architecture are possible using current technology (see Section 5). The primary benefits are:

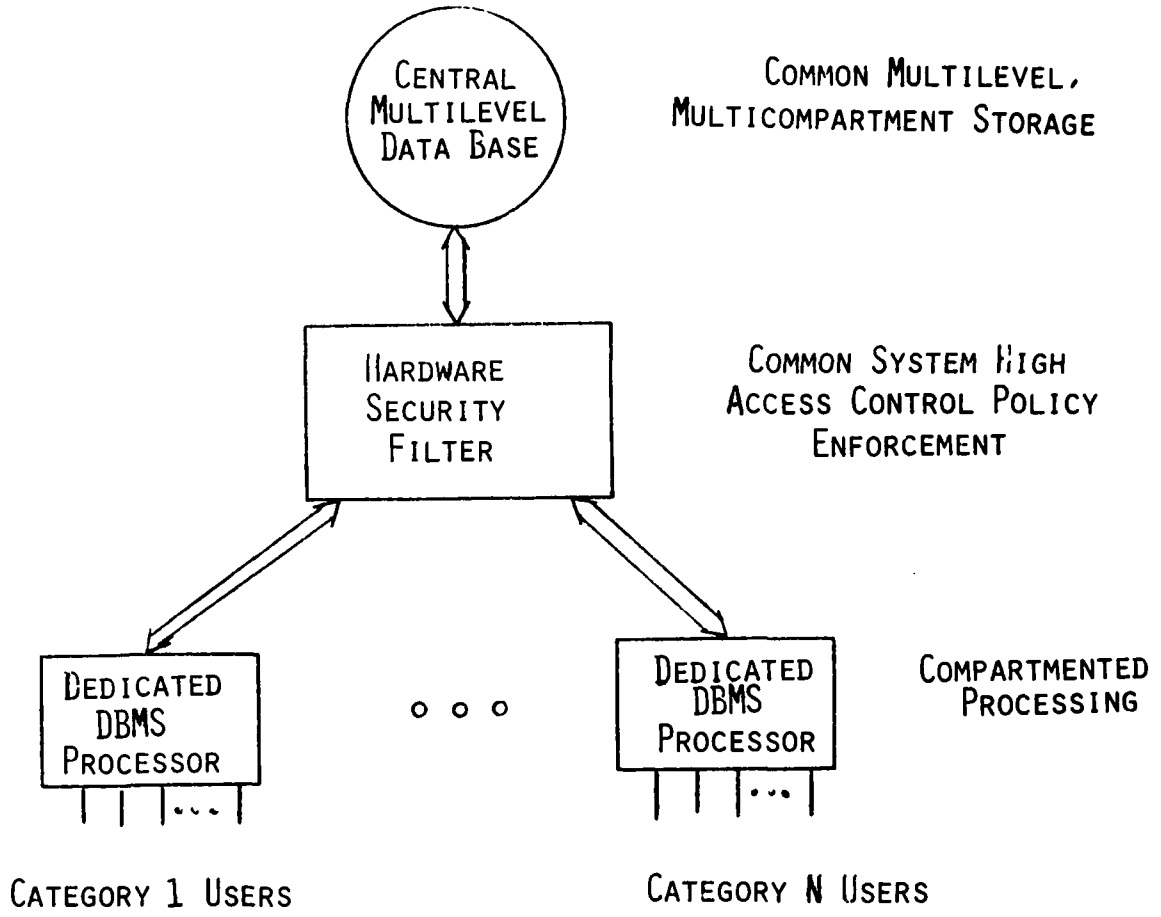


Figure 1.1-1 Secure DBMS Conceptual Architecture

- o Secure Multi-Level DBMS Operation
- o Local Dedicated Processing Equipment
- o Provable Hardware Enforcement of Non-Discretionary Security Without Certified or Trusted Software

Several desirable features are also available from the architectures described in this report. The architecture can be implemented so as to be compatible with existing systems and be easily adaptable to different DBMS processors. The use of existing commercial fiber optic links will allow the DBMS processors to be located up to several hundred meters from the HSF. By replicating portions of the HSF and using standard dual ported disk drives it is possible to nearly double central data base throughput capacity and provide fault tolerance. This architecture provides general multi-level security and is not restricted to support of a particular DBMS type. However, a relational DBMS may be preferable from a human factors perspective and because of its inherent flexibility.

Based on preliminary sizing it is evident that the cost of the multiple dedicated DBMS processors required by this architecture presently dominates system acquisition costs and is currently the primary limiting factor for very large multi-compartment applications (eg. 16 or more categories of dedicated processing). However, the cost of suitable processors has been decreasing steadily and it is very likely that the cost of a suitable DBMS processor subsystem will drop below $40K ($10K hardware) within 5 years, thus alleviating this factor. (See Section 7.) The primary limiting technical factor is the sharing of a single central data base, resulting in dilution of data base throughput per DBMS processor. Analysis indicates that the use of DBMS processors with virtual memory and local disk storage can largely compensate for this effect by drastically reducing central data base loading. (See Section 6.)

Because of the need for downgrading information, at least one trusted multi-level processor is required for this operation. The trusted software in this processor and its trusted users are the primary non-discretionary security threat. Secondary security advantages offered by this architecture for downgrading operations are: all downgrading occurs on an isolated multi-level processor with limited access, and a central security monitor/audit trail to alert the security officer to all downgrading activity can be placed in a separate processor isolated from users and their code.

Discretionary and integrity protection (correctness) may

be entrusted to the DBMS operating system (either commercial or secure) software. Alternately it may be externally enforced by the use of end-to-end encryption (E^3) in a back end DBMS machine architecture with certified firmware in front end I/O (input and output) processors, and certified code in a central Access Control Subsystem (ACS). This second approach requires development of a custom multi-level DBMS and security related software to support E^3 and the architecture. It also requires that there be no user or application code in the DBMS processor, i.e., the user interface is a query language. Multi-level security requirements place some inherent limitations on multi-level relational data structures. In particular, the primary key, schema, and related access structures must be classified at the lowest level of the relation.

In conclusion, the HSF approach to adding multi-level security is far superior to a secure OS approach for a large number of reasons. In particular, the HSF approach

- facilitates system certification because validation relies primarily on hardware flow control analysis
- results in a lower development cost, through the use of hardware to eliminate custom software, thus exploiting the decrease in hardware cost
- permits the use of standard commercial software, including standard operating systems.
- is adaptable to different DBMS processors in a transparent fashion, thus permitting the upgrading of existing systems and the development of future systems. This resolves the dilemma of retaining an obsolete system or development of a new secure O S system by permitting an evolutionary (staged) modular design that supports planned upgrades and standard options.

Such upgrades might be for planned fault tolerance modifications, standard expansion options etc. Section 8 describes the flexibility achievable by applying the HSF to existing systems through a planned program of system upgrades.

2.0 SYSTEM SECURITY REQUIREMENTS

This section discusses the security requirements for a Secure DBMS system. These requirements include discussions of the security policy to be enforced, threat identification, and data protection level. The security policy describes the access rules for data read or modification in terms of a DoD security policy of discretionary and non-discretionary security and a data quality (integrity) policy. The threats discussed in this section are general in nature and are used in the design of a system as guides. The actual security analyses of a system are presented following the system descriptions. The access control policies can be applied to data at several levels. The data protection level discusses these levels in terms of DBMS impact.

The following nine security related sections are presented below:

2.1 Problem Description and General Approach
2.2 Formal Security Policy Description
2.3 General Threat Descriptions
2.4 Covert Channels
2.5 Trusted Entities
2.6 Security Monitors and Audit Trails
2.7 Multi-Level Security
2.8 Protection Granularity Discussions
2.9 Proof of Correctness Requirements

2.1 Problem Description and General Approach

The use of automated data processing equipment has become wide spread throughout industry and the government because it permits the handling and storage of vast amounts of information at affordable costs. The use of computers has enabled countless opportunities for industrial growth, new applications, great savings in labor and improvements in the quality of decisions. Together with all of the advantages that computer technology has brought to man, the computer age has also created a totally new and nearly unsolvable problem: information privacy. In governmental and government - related applications, this problem often escalates to that of SECURITY: Any computer system handling DoD classified information must

be controlled and operated within strict guidelines set forth by the cognizant agencies.

This study addresses the military secure data processing requirements such as those encountered by the Air Force command, control, and intelligence applications. However, the technologies discussed are equally applicable to any data processing application requiring secure data access and user interaction control. The basic problem is illustrated in Figure 2.1-1. Here users and data, at various security levels, desire simultaneous access to the machine resources. Data of all security levels is to be present on the system. Users with the proper security credentials are granted data accesses. DoD security policy (See Figure 2.1-2) requires that an individual possess the required non-discretionary and discretionary privileges before being granted access to the information. Non-discretionary security requires that the individual have a security clearance of higher or equal level than the level of the information requested. Discretionary security requires that the individual possess a proper need-to-know for the requested information

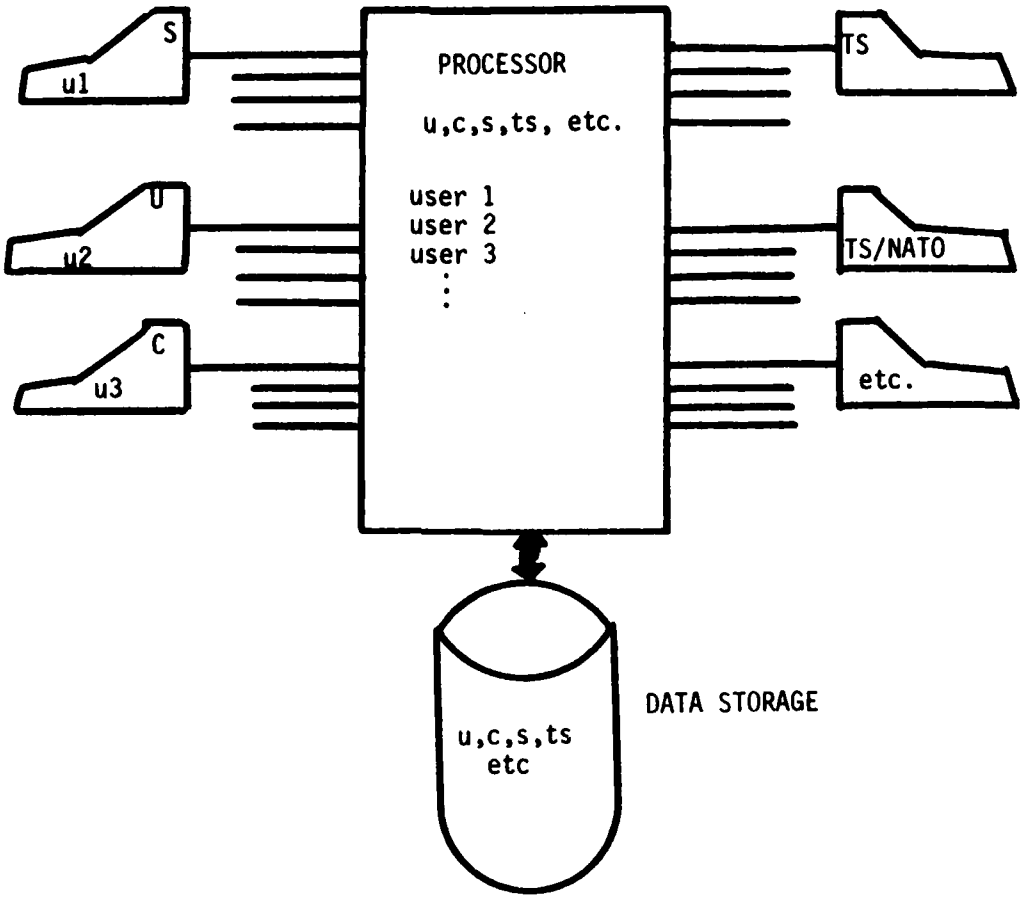
There are several classical approaches to the handling of DoD classified information in computer systems. Figure 2.1-3 describes four basic techniques:

- o Dedicated Security Mode
- o System High Security Mode
- o Multi-Level Security Mode
- o Controlled Security Mode.

An additional technique, period processing, is often used with applications employing the dedicated mode. Period processing is a technique wherein a specific security mode is assigned to a system for a designated time.

The above security techniques have several shortcomings, the principle one being cost. These disadvantages are as follows:

- o obtaining the security clearance for all the users in a system high processor is costly and can often require considerable time.
- o The dedicated mode lacks flexibility and often fails to efficiently utilize the available machine resources. Use of periods processing increases the system utilization only



- o STORAGE FACILITY: MULTI-LEVEL
- o USER PROCESSING: CONCURRENT MULTI-USER,MULTI-LEVEL

Figure 2.1-1 Problem Description

DoD Security Policy. The complete body of law, regulations and policy concerning the safeguarding of Defense sensitive information. DoD security policy includes all the espionage laws, the DoD regulations, and DoD authorized commercial classification for handling and access to information concerning national defense. The basic policy sets four levels and several categories on non-discretionary information control and requires that anyone accessing classified information have a "need-to-know" for the particular information in question.

Security Level. In the context of formal security modeling, the fundamental security attribute of subjects and objects. Security levels combine a level (e.g., Unclassified, Confidential, Secret, Top Secret) and a set of need-to-know categories. A derived partial ordering of security levels is defined using a combination of the "less than" order on levels and the "subset" relation on category set. Thus (Secret, NATO) is less than both (Top Secret, NATO) and (Secret, (NATO, CRYPTO)). The security levels (Confidential, NATO) and (Top Secret, CRYPTO) are incomparable. This derived partial ordering on security levels is the basis on which all subject-to-object access is determined.

Classification. The level of protection that must be afforded information. It is the information counterpart of clearance in DoD security policy. It applies to such system objects as buffers and files, as well as to "real-world" security-related documents.

Clearance. An authorization allowing a person access to classified information. This is a "real-world" term used in connection with DoD policy, whose mathematical counterpart is security level (q.v.). A clearance typically consists of a level (unclassified through top secret) and a need-to-know category or categories.

Non-discretionary Security. The aspect of DoD security policy which restricts access on the basis of security levels. A security level is composed of a level and a category set restriction. To access an item of information, a user must have a clearance level greater than or equal to the classification of the information, and also have a category clearance which is a superset of the access categories specified for the information. See discretionary access controls, security level.

Discretionary Access Controls. Access controls to an object that may be changed by the creator of the object. More generally, mechanisms that allow each subject, at its own discretion,

Figure 2.1-2 DoD Security Policy Definitions

to decide which of its own access rights are to be given to any other subject on a need-to-know basis.

Need-To-Know. A job-related requirement for access to specific information. Need-to-know implies discretionary control of information - even though potential accessors may have the necessary clearance.

Integrity Control. In a formal security model, integrity is used to provide protection against unauthorized modification or destruction of information and is a dual of Non-discretionary Access control. An integrity level is comprised of a level and a category set. To modify an object a user must have an assigned integrity level greater than or equal to that of level assigned to the object, and also must have the integrity category which is a superset of that of the object. Integrity is used to prevent dilution of the quality of information by creating a hierarchical partial ordering in the modify accesses permitted for an object.

Compartment or Category. The unit into which security information is partitioned, corresponding roughly to an interest group or topic area, e.g., NATO, CRYPTO. Category information can exist at many levels, unclassified through top secret. A subject must be cleared to an adequate level and have access to the proper category or set of categories before access to classified data can be given.

Figure 2.1-2 (Cont'd) DoD Security Policy Definitions

Security Mode. A Department of Defense term for "Authorized variations in the security environments and methods of operating ADP systems that handle classified data." DoD ADP security policy (DoD Directive 5200.28) defines four modes: dedicated, system high, controlled and multilevel security modes.

Dedicated Security Mode. In government installations, a mode of operation in which the computer system, its connected peripheral devices and remote terminals are exclusively used and controlled by specific users or groups of users who have a security clearance and need-to-know for all categories and types of classified material contained in the computing system.

System High Security Mode. The mode of operation in which the computer system and all of its connected peripheral devices and remote terminals are protected in accordance with the requirements for the highest security level of material contained in the system at the time. All personnel having computer system access have the security clearance and need-to-know for all material then contained in the system.

Multilevel Security Mode. A mode of system operation permitting data at various security levels to be concurrently stored and processed in a computer system where at least some users have neither the clearance nor the need-to-know for all classified material contained on the system. Separation of personnel and material on the basis of security level is accomplished by the operating system and associated system software.

Controlled Security Mode. A mode of system operation in which there are users who have legitimate access to the system but have neither a security clearance nor a need-to-know for all classified material contained in the system. Internal hardware and software must be provided and approved for maintaining separation of data and users with different classifications and clearances. No more than three adjacent security levels can be supported concurrently, and specific approval by the designated approval authority is needed for this mode.

Period Processing. In computer installations, a mode of processing in which a specific security mode is temporarily established during a specific time interval for processing sensitive information. For example, the computer system could process secret information in the dedicated security mode during one period, and unclassified material in a second period. The computer system must be purged of all information before transitioning from one period to the next whenever there will be new users who do not have clearance and need-to-know for information processed during the previous period.

Figure 2.1-3 Classical Data Processing Security Modes

at the expense of performing inter-level "sanitizations" wherein the system is cleared of all information of a previous higher security level.

- o secure operating systems such as those used in the multi-level security mode, are extremely costly to develop and, at present, not able to be validated except in very special applications and then only with some risk.
- o with the exception of the multi-level and controlled security mode, none of the secure operating modes permits simultaneous handling of different levels and categories of classified information.

This report presents a feasibility study on the use of a distributed architecture to support data base management and other processing requirements for DoD classified information. The system created operates basically in a controlled security mode whereby all information requests are passed through a security filter to determine if the requestor possesses the proper security credentials for the request. Thus, users are separated from a multi-level central data base by this security filter (Figure 2.1-4). In addition, users of various security levels are separated from one another through the use of separate host DBMS processor for each security level. It is therefore the task of the security filter to pass only that information to a host DBMS processor for which the established security policy is met. The mechanisms by which the security filter achieves this task are described in Sections 3 through 7. The means of access control for these systems is through a hardware security filter (HSF) (Section 4.2) whose access control can be certified through flow control analyses.

These systems also provide discretionary access control, although not at the hardware level. All data requests are screened by an Access Control Subsystem processor (Section 4.3) which compares the user's identification to those of the access control list for the requested object. If the discretionary access is granted, the information is sent to the user at his DBMS processor. At this point, only the protection mechanisms of the operating system of the DBMS provides any isolation between the users. This is a much weaker security than that provided for security level control by the HSF but is adequate for discretionary protection at almost all applications and represents a compromise in the cost of providing each user with a dedicated processor.

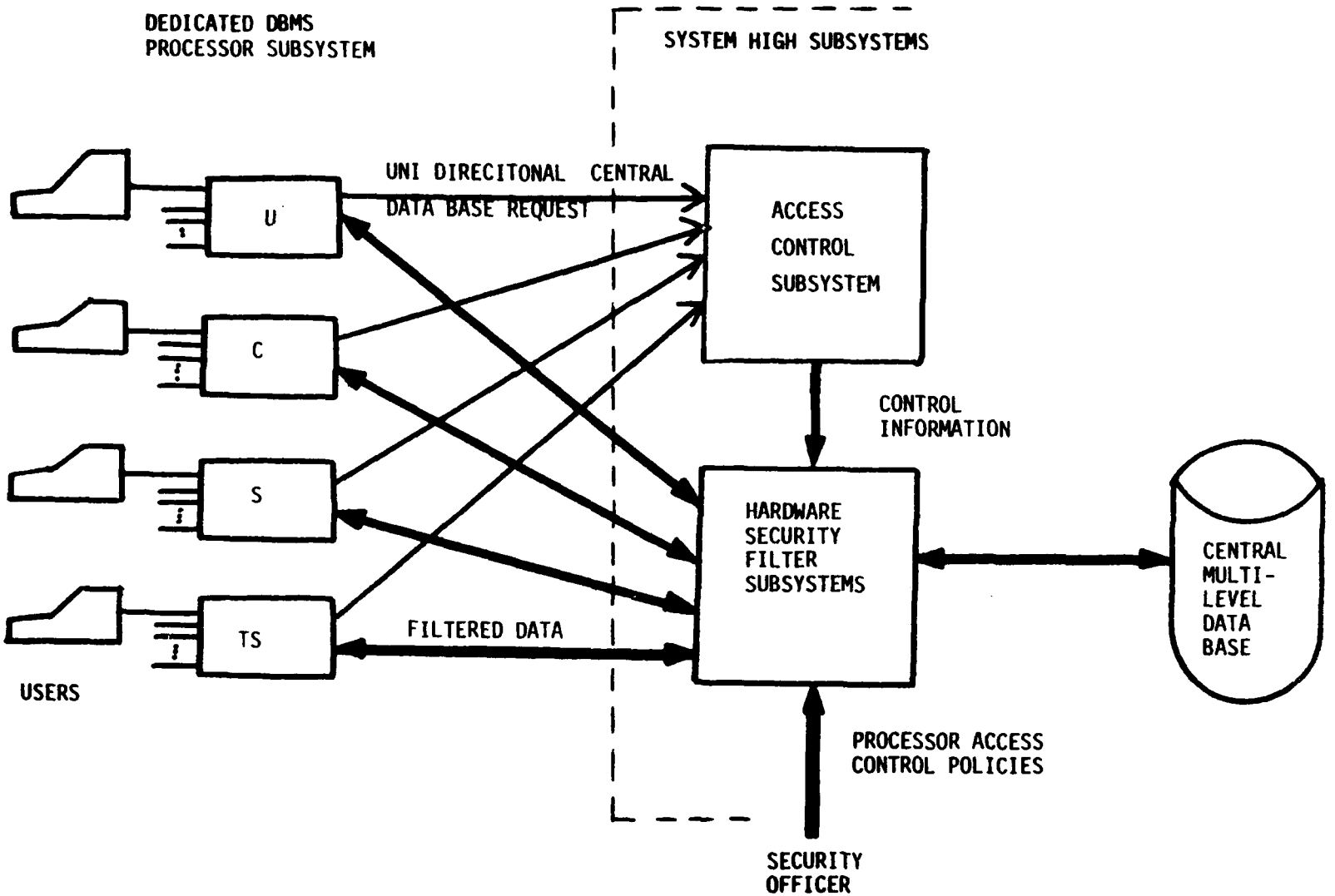


Figure 2.1-4 System Concept Block Diagram

A third entity of the security policy is termed integrity. This is not the "correctness" integrity provided by many data base systems but rather a security related "integrity". This integrity is designed to preserve the "quality" of existing data and prevent any reduction in the quality value through inclusion or replacement of information by data from a lower integrity source.

The requirements of non-discretionary, discretionary, and integrity access control for the enforcement of security are detailed in the discussion of the Formal Security Policy Description of the following section.

2.2 Formal Security Policy Description

The security requirements for a secure system architecture are defined through a formal security policy. This section describes the security policy developed for use in this study and implemented in the hardware and software protection mechanisms. The model is formally detailed in Appendix A and therefore this section will only summarize the requirements resulting from the formal proofs. The approach used in the formal security model definition and proof follows that of Bell and LaPadula[1] (Figure 2.2-1) with modifications and extensions made to the model. One such extension, an integrity rule designed to preserve data "quality" is an adaptation of an approach described by Grohn[2]. The security requirements are proven to be correct through inductive proofs, using relational algebra and ordinary set theory. Appendix A painstakingly proceeds to describe the requirements for a secure system by theorem and formal proof.

These proofs are presented in four major groupings:

I. Requirements of Actions for System Security Preservation

II. Properties of Secure State Preserving Rules

III. Requirements for Secure State Preservation During Additions to the Current Access Set.

IV. Security preservation during actions of rules which do not add access, do not change protection or do not remove access permissions.

[1] all references are presented at the end of this report.

Bell-LaPadula Security Model. An "access control" type of security model based on state-machine concepts; sometimes called the MITRE Model. In this model, the entities in a computer system are abstractly divided into sets of subjects (active entities such as processes) and objects (information containers). The notion of a secure state is defined, and an inductive proof of system security can be given: the initial system state is shown to be secure, and every state transition is shown to preserve this property.

A system state is defined to be "secure" if the only permitted accesses of subjects to objects are in accordance with specified security level restrictions. For example, a subject is permitted to read data at its own level or at a lower level (simple security condition), and to write data at its own level or at a higher level (*-property). State transitions perserve the "secure state" property in accordance with tranquility, erasure and activity principles.

Grohn Security Model. An extension of the Bell-LaPadula model results from the addition of an integrity model. This model is also known as the I.P. Sharp Security Model. Integrity is essentially the mathematical dual of the security model, incorporating a "simple integrity principle" and integrity *-property".

Figure 2.2-1 Security Model Descriptions

Group I Theorems (Theorems 1-3) present the conditions necessary to restrict the set of actions to those actions which maintain a secure system.

Group II Theorems (Theorems 4-6) prove that an action set limited by a set of secure state preserving rules forces the system to be a secure system. As a result, these three theorems prove that security enforcement lies in the design and implementation of the rules themselves.

The Group III Theorems (Theorems 7-9) present the set of conditions which must be met to maintain a secure state under additions of subject, object, access tuples to the current access set.

The Group IV Theorems present the conditions for the actions of a rule to be secure state preserving in the special cases where protection levels do not change, accesses are not removed, and permissions are not added.

The Group I theorems result in a Corollary which describes the necessary conditions for preservation of a secure system: the simple protection property; the *-protection property; and the discretionary protection property. These properties will be discussed in terms of their constituents, security and integrity in Section 2.2.1 in detail.

In addition to the formal math security model, security flow analysis is used to verify the functional security of hardware subsystems. This too is detailed in a following section (Section 2.9).

2.2.1 Security Requirements

The formal security math model details the requirements for a secure system in terms of current access sets, system states, user privileges, etc. In this section, a more simplified description of the basic security requirements will be given.

2.2.1.1 SUBJECTS, OBJECTS, and ACCESSES

The security model defines the security requirements for access of objects by subjects in a computer system. (Figure 2.2-2 defines these and other

Access Control. A strategy for protecting objects from unauthorized access.

Access. The ability and the means necessary to store or retrieve data, to communicate with (i.e., provide input to or receive output from), or otherwise make use of any resource in a computer system.

Access Mode. A distinct operation recognized by the protection mechanisms as a possible operation on an object. Read, write and append are possible modes of access to a file, while execute is an additional mode of access to a program. See security mode in Figure 2.1-3.

Subject. An active user of a computer system together with any other entity acting on behalf of a user or on behalf of the system; for example, processes, jobs, and procedures may all be considered subjects. Certain subjects may also be considered to be objects of the system. See Bell-LaPadula security model, (Figure 2.2-1) and object (below).

Object. In a formal security model, an identifiable resource, data container or related entity of the system; the counterpart of subject. Software-created entities such as files, programs and directories are objects, as well as hardware resources such as memory blocks, disk tracks, terminals, and tapes.

Simple Security Condition. A security model rule allowing a subject read-access to an object only if the security level of the object is the same or less than the security level of the subject and the category set of the object is a subset of the subject's current access category set.

Security *-Property. A security model rule allowing a subject write-access to an object only if the security level of the object is the same or higher than the security level of the subject and the category set of the object is a superset of the current category set of the subject.

Simple Integrity Condition. A security model rule allowing a subject write-access to an object only if the integrity level of the object is the same or less than the integrity level of the subject and the integrity category set of the object is a superset of the current integrity category set of the subject.

Integrity *-Property. A security model rule allowing a subject read-access to an object only if the integrity level of the object is the same or higher than the integrity level of the

Figure 2.2-2 Selected Security Model Definitions

subject and the integrity category set of the object is a subset of the current integrity category set of the subject.

Authentication. The act of identifying or verifying the eligibility of a station, originator or individual to access specific categories of information. The process by which the Government determines concurrence with specifications.

Access Control List. A list of subjects which are authorized to have access to some object. See subject, object.

Security Flow Analysis. A type of security analysis performed on a non-procedural formal system specification which locates potential flows of information between system variables. By assigning security levels to system variables, many indirect information channels can be identified. Security flow analysis defines a security model similar to the access control model (Bell-LaPadula) but with a finer protection granularity.

Figure 2.2-2 (Cont'd)

security related concepts.) Basically, a subject is any active process in the system,including users, user programs, and any other processes acting in the user's behalf. An object is any inactive system entity and includes data files, inactive programs (code, directories, etc). An access is the linkage between the subject and object. Therefore, computer security can be defined in terms of providing an access control mechanism capable of enforcing the desired security policy. A DoD non-discretionary and a discretionary security policy, an integrity policy, and a tranquility principle are enforced in this protection plan.

An access can be of several forms. The recognized accesses in this study are:

r read
w write
a append
d delete

An execute access is often used but is omitted from this model due to the fact that execute access in the proposed protection architectures is processed identically to a read access.

2.2.1.2 SECURITY, INTEGRITY, and DISCRETIONARY PROTECTION

Access control rules employed in the security policy specify exact requirements for access in terms of classification and clearances (non-discretionary access), need-to-know (discretionary access), and integrity levels. (These concepts were defined in Figure 2.1-2).

Appendix A defines the security requirements for various access modes in terms of protection requirements. Protection is defined in terms of security and integrity as presented in Figure 2.2-3. This figure also presents these requirements in terms of the simple security rule, *-security rule, simple integrity rule, and *-integrity rules. This dissected form of protection is presented because the proposed implementations provide protection in a distributed form: the simple security and *-security rules are enforced in hardware; (in the Hardware Security Filter - See Section 4.2) the simple integrity and *-integrity rules are enforced in software (in the Access Control Processor see Section 4.3).

READ ACCESS REQUIREMENTS: $P_c(S) \rightharpoonup P_o(O)$

$\underline{r} \in M_{ij}$ for Subject i, and Object j

This implies:

$S(s) \geq S(o)$	$I(s) \leq I(o)$
$C(s) \supseteq C(o)$	$K(s) \subseteq K(o)$
(Simple Security Rule)	(*-Integrity Rule)

APPEND ACCESS REQUIREMENTS: $P_o(O) \rightharpoonup P_c(S)$

(write without read) $\underline{w}\ M_{ij}$ for Subject i, and Object j

This implies:

$S(s) \leq S(o)$	$I(s) \geq I(o)$
$C(s) \subseteq C(o)$	$K(s) \supseteq K(o)$
(*-Security Rule)	(Simple Integrity Rule)

MODIFY ACCESS REQUIREMENTS
(requires read before write) $P_c(S) = P_o(O)$

This implies

$S(s) = S(o)$	$I(s) = I(o)$
$C(s) = C(o)$	$K(s) = K(o)$
(Simple and *-Security rules)	(Simple and *-Integrity rules)

NOTATION:

- $S(\bullet)$ Security Level Operator - returns security level of argument
- $C(\bullet)$ Security Category Operator-returns security categories of argument
- $I(\bullet)$ Integrity Level Operator-returns integrity level of argument
- $K(\bullet)$ Integrity Category Operator-returns integrity categories of argument.

Figure 2.2-3 Summary of Security Rule Requirements

2.2.1.3 Discretionary Access

Discretionary access refers to access permissions granted on an individual basis to persons who exhibit a need to the information. It is granted by the owner (or other responsible person, such as the security officer) to a particular subject in a specific mode or set of modes. The decision to grant such access is judgemental on the part of the owner and is normally made based upon a "need-to-know". The access granted can be any subset of the set of valid accesses.

Discretionary access is often described in terms of an access matrix as described in Appendix A. Alternately, discretionary access permissions can be described in terms of an access control list (ACL) formed from such a matrix by separating the matrix into vectors from columns or rows of the matrix. Thus, such an ACL can be keyed by subject or object identifiers. For example, an ACL keyed by object name would have a list of subjects and their accesses for each named object. For example:

OBJ1: (SUB1={R,W}, SUB3={R}, SUB4={R,W,D})
OBJ2: (SUB1={R,A}, SUB2={R,W,A,D}, SUB3={R})
etc.

2.2.1.4 Tranquility Principle

The tranquility principle states that the current protection level of a subject or object never changes.

This rule is enforced to prevent direct reassignment of objects to new protection levels. Should it be possible for an object to be assigned a lower protection level, a direct violation of the security policy would result. A relaxation of this rule could permit assignment of an object to a higher protection (that is, a higher DoD classification or category or a lower integrity level or category) without a security rule violation. Such a relaxation requires a policy decision by the security administrator of the installation. Actually, the security policy already permits this process indirectly through a "copy" procedure whereby a low level object can be legally copied to a new object at a higher level.

It is absolutely necessary that subjects not be permitted to lower

their protection during a given session without release of all resources and objects. If this tranquility principle for subjects was not enforced, a subject would be able to access a high level object, reduce his current protection level, and store the object at the new, lower level - a direct violation of DoD security. Similar to the tranquility principle for objects, the tranquility principle could be relaxed for the subject and permit a subject to increase his current protection level, up to his maximum level. In practice this will be difficult because the envisioned security architecture will not recognize a user port to be but at a single level as assigned by the security officer. This is due to the hardware implementation of the security filter. However, the subject could reduce his integrity level in this architecture, provided that a mechanism is available to recognize this procedure.

One major relaxation of the tranquility principle is allowed for in the secure DBMS design. Because of the need to permit lower level summaries of highly sensitive information, the DBMS architectures permit violations of the tranquility principle by trusted subjects operating on trusted DBMS processors (See Section 2.6). Here subjects are totally outside the security restrictions of the security policy. Consequently these subjects must be limited in number and must be carefully screened. Such trusted operations are permitted only after the security officer has properly established the DBMS processor as a special policy-excluded machine. Extensive security monitoring and audit trail recording is used by the security officer to scrutinize all operations performed on the trusted processor and detect any abuses of the procedure.

2.2.1.5 Requirements for Read Access

Read access is defined as an access to an object permitting its observation with no modification. The security model requirements for read access, from Figure 2.2-3, states that the simple security rule, the *-integrity rules, and discretionary access must be enforced. These rules are now defined verbally:

Simple Security Rule: The simple security rule requires that for read access the subject must:

1) have a security clearance level higher than or equal to the classification level of the object; and

2) possess a superset of the categories assigned to the object.

*-Integrity Rule: The *-integrity rule requires that for read access the subject must:

1) be at an integrity level which is lower than or equal to that of the object; and

2) possess a subset of the integrity categories of the object.

The simple security rule merely states that the subject cannot view an object at a higher level. The *-integrity rule might seem overly restrictive. Its intent is to prevent the integrity level of the subject from being reduced by giving him information which is at a lower integrity level than his level. If the integrity level of the subject was permitted to be reduced by allowing read accesses to low integrity level objects, any subsequent writes to the data base by that subject would have to be conducted at the lower level. Thus, the *-integrity rule prevents a lower integrity level object from reducing the integrity level of the subject.

2.2.1.6 Requirements For Append Access

Append access is defined as a write operation which does not require a prior read of the object (or portion of the object) being written. The security model access requirements, from Figure 2.2-3, state that the *-security and the simple integrity rules must be preserved and that the subject must have write discretionary permission. These rules are now verbally defined;

*-Security Rule: The *-security rule states that for write access the subject must :

1) be at a security level lower than or equal to that of the object; and

2) possess a superset of the categories of the accessed object.

Simple Integrity Rule: The simple integrity rule requies that for write access the subject must:

1) be at an integrity level greater than or equal to that of the object; and

2) possess a subset of the integrity categories of the object.

The *-security rule thus states that the subject cannot write to a security level lower than his own current level. The current level of a subject is that level by which he is currently recognized. (A user may be cleared to a specific maximum level. However, this does not require that he be recognized at this maximum level. Instead he may choose a lower level as his current level for processing purposes). The simple integrity rule states that the subject's integrity must be at least as great as that of the objects being written so as to prevent dilution of the integrity of the object.

2.2.1.7 Requirements for WRITE Access

When an object is modified, it must be read prior to being written. Thus the subject's access requirements are the intersection of those described above for read and append. Specifically, the subject and the object must be at the same security and integrity levels and the subject must have been granted both read and write discretionary permissions.

Write access for multi-level objects is somewhat more complex. A multi-level object is one which contains identifiable parts which are at different protection levels. For example, it is possible for an object to have some fields at an unclassified level and other fields at the secret level. If it is desired, a security policy can be devised (without violating any DoD security policy with respect to the object)such that blind updates to an object can be performed. In this case, an unclassified user could blindly update the secret field. This would be done by using (reading) the unclassified field as a "key" to locate the data base entry to be changed and subsequently writing to the secret area without first reading it. The only weakness of this approach is that potential secret material is entered through an unclassified machine. However, this procedure could be an asset in those instances wherein unclassified objects achieve a higher classification when viewed in aggregate. Nonetheless, there is a more suitable technique of achieving the same result whereby the information is stored in the data base at the lower level and "viewed" by personnel at a higher level. In this instance, the higher level aggregate is formed only by the personnel cleared

for that level. The lower level subjects merely collect and report low level information.

A weather collection node is an example of such an unclassified process. When the weather data base is merged with the target base, the weather entries thus merged become sensitive.

2.2.1.8 Requirements for Execute Access

Because of the manner in which the proposed architectures enforce security, an execute access requires that the desired object be read by the subject's processing equipment. As the applications processor contains no validated protection mechanisms, it is impossible to prevent the subject from viewing an object accessed in such a manner with any certainty. Therefore, the requirements for execute access are identically those for a read access.

2.2.1.9 Delete Access

A delete access is a destructive write process. Normally, an object must be viewed (read) prior to deleting (writing). Thus, a delete access has the same requirements as that of a write access.

2.3 General Threat Description

Threats to the operation are of several types. Appendix B discusses many of these in detail. This section will briefly summarize the discussions of Appendix B.

Threats can be categorized in several groupings. One such catalog describes threats as one of the following types (See Figure 2.3-1):

- o Physical Threats - physical attacks to the system hardware and facilities; e.g., sabotage, system failure.
- o System Threats - Attacks to the system operational integrity e.g., software or hardware flow.
- o Data Threats - attacks designed to require information through invalid means; e.g., impersonation, browsing
- o Direct Security Attacks - attack upon the system security system; e.g., covert channels, illegal downgrades.

Many of these threats cannot be addressed by a secure DBMS development, for example, development of a Secure DBMS is not intended to prevent physical threats. Many of the system threats and data threats are countered through normal good engineering practices.

Several of the data threats and direct security threats are addressable by the secure DBMS design. Through the use of single level processors in a multi-level system, the threats of a non-discretionary disclosure through browsing and artifice are eradicated. The discretionary access control, (within the dedicated user DBMS processors and in the Secure DBMS Access Control Subsystem) enhances protection against illegal data access.

There are several threats which are most difficult to alleviate, even within the Secure DBMS architecture presented here. These threats are described in Figure 2.3-2. The trusted subject threat cannot be removed if downgrade processing is to be provided (See Section 2.5). This threat is only reduced through personnel screening, limiting this privileged mode to a minimum number of personnel, and proper security auditing of the privilege processes.

Trap door software in trusted processes cannot be eliminated short of validating the software - an extremely costly process in all but the smallest

PHYSICAL THREATS

FIRE	THEFT	ENVIRONMENT
EXPLOSION	SABOTAGE	COMMUNICATIONS FAILURE

SYSTEM THREATS

SOFTWARE TRAP DOORS	INFORMATION HANDLING PRACTICES
HARDWARE FLAWS	O/S UPDATE PROTECTION

DATA THREATS

IMPERSONATION	ARTIFICE	TAPPING	RADIATION
FOIBLE	BROWSING	DATA DESTRUCTION	

DIRECT SECURITY THREATS

COVERT CHANNELS	ILLEGAL RE-CLASSIFICATION OF OBJECTS
TRUSTED SUBJECTS, PROCESSES	ILLEGAL ACCESS TO OBJECTS

FIGURE 2.3-1 SUMMARY OF DATA BASE SYSTEM THREATS

TRUSTED SUBJECTS - SPECIAL PRIVILEGES ARE REQUIRED TO PERMIT SECURITY DOWNGRADE OPERATIONS. THIS IS DONE IN A SPECIALLY CLEARED AND DESIGNATED DBMS PROCESSOR UNDER THE SCRUTINY OF THE SYSTEM SECURITY OFFICER WHO IS MONITORING SYSTEM OPERATIONS.

TRAP DOOR SOFTWARE- TRAPDOOR SOFTWARE IN THE DOWNGRADE DBMS PROCESSOR IS THE MOST SEVERE THREAT BECAUSE IT IS MOST DIFFICULT TO DETECT UNTIL THE VIOLATION HAS OCCURRED. GOOD SECURITY MONITORING PRACTICES ARE A MUST.

COVERT CHANNELS - COVERT CHANNELS PROVIDE A MEANS OF MOVING CLASSIFIED INFORMATION FROM A HIGH LEVEL TO A LOWER LEVEL. THIS REQUIRES AT LEAST TWO COOPERATIVE PROCESSES. GOOD DESIGN PRACTICES AND VALIDATION THROUGH DATA FLOW ANALYSES ARE THE PRIMARY MEANS OF ELIMINATION. A GOOD SECURITY MONITOR PROCESS MIGHT ALSO PERMIT IDENTIFICATION OF THE USE OF THE DIFFICULT "TIMING CHANNELS"

FIGURE 2.3-2 SECURE DBMS PRINCIPLE THREATS

process. Again, a good security monitoring process and regular system audits can facilitate the detection of the use of such software. Section 2.6 discusses security monitors and audit trails in detail.

Covert channels are a design issue and are addressed by the Secure DBMS architecutre. Section 2.4 discusses Covert channels in greater detail.

2.4 Covert Channels

A covert channel is a serious threat to the secure DBMS. "Indirect" or "covert" channels are communication paths not explicitly intended for data transfer, but which can pass information through the selective use of system resources - that is, through "leakage" or "signaling". Indirect channels are generally categorized as "storage" and "timing" channels. Timing channels are those that exploit the system clock or system performance characteristics to pass information and are difficult to identify in a non-procedural system specification, while most storage channels can be identified by flow analysis techniques. Both types of indirect channels are exploitable only through interprocess cooperation.

It is necessary that all covert channels in the designed architecture be eliminated or otherwise restricted to a sufficiently low bandwidth so as to reduce their utility. Data flow analysis will be used to identify any potential storage channel. Timing channels are somewhat more difficult to prevent. Methods of reducing the predictability of system processes will tend to reduce the effectiveness of a timing channel. For example:

1) Assign each DBMS process a specific time slice in which to use a resource - a Time Division Multiplexing. In this manner, the activities of a user cannot be used to modulate the response time of the system
2) Periodically, randomly alter the request queue order. This will scramble the response times and render them unpredictable
3) Add random delays to the channel response time - again removing the predictability of the system response.

Both methods 1 and 3 decrease system throughput. Method 1 can waste much time when there are channels with no pending requests. Method 3 wastes time through the addition of delays. Method 2 may not directly wastetime, but it is only remotely possible that a particular user could remain in the queue an inordinant amount of

time. This would occur only if his request was frequently selected to be swapped to a lower queue position.

It is necessary to note that when a system has a number of users in addition to the two cooperating in the covert channel, the predictability of response time is greatly reduced because these other users will tend to randomize the predictability of the response required by the cooperative covert channel users. This basically forces the covert channel to a lower bandwidth, reducing its usefulness. A single terminal simulator issuing DBMS requests at random can act as an alternative to many users if desired.

Only the Time Division multiplexing technique (Method 1) is absolute in its destruction of this timing channel. It may be easier and less costly to permit the channel to exist and devise a technique for the security monitor to detect its exploitation and alert security of the fact.

2.5 Trusted Entities

Trusted entities are those items within a system upon which the enforcement of security depends but which cannot be proven to be secure. In addition, some processes are actually desired, such as a downgrade option. Trusted entities can be of any three types:

personnel; processes; equipment

It is the goal of the secure DBMS design to minimize the quantity of these trusted entities.

2.5.1 Trusted Personnel

Some personnel within the secure DBMS must be accepted as trusted. There is no method to insure that an individual will not violate security. There are 3 basic groups of trusted personnel identifiable:

1) The security administration and operational security officer staff. Figure 2.5-1 describes the activities of the security officer.
2) The Data Base Administrator (DBA). Figure 2.5-2 describes the responsibilities of the DBA.
3) System users. System users are trusted at two levels as described in Figure 2.5-3.

The trusted operations associated with the security downgrading process is probably the greatest system security threat.

- o general security maintenance
- o selection of the desired security access control policy for a given DBMS channel
- o *validation of the selected access control policy*
- o ensuring that period processed equipment is sanitized (cleared of all sensitive information) between level changes
- o monitor the security audit trails to ensure that no system failure has occurred and to screen all attempted assaults upon the security system
- o perform data back-ups of all data bases
- o initiate the scheduled change of cryptographic keys for systems employing data encryption.

Figure 2.5-1 Summary of Responsibilities of the DBMS Security Officer

- o Specification of data bases and their structures
- o Coordination of all data base efforts to ensure consistency in their procedures
- o Coordinate data protection with security personnel. Includes security levels, categories, integrity levels and categories, and discretionary access requirements
- o Review of the data base system performance and determination of change required to optimize the data base performance.
- o Specification of access paths between user data bases.

Figure 2.5-2 Summary of Responsibilities of the DBMS Data Base Administrator

NORMAL USER MODE - users are expected to properly handle all classified information, using standard DoD procedures
- users are expected to properly classify all information which they handle according to security directives.

SPECIAL TRUSTED ACCESSES - users can be granted special DBMS privileges which violate the security policies in order to perform special processing, such as downgrading operations. Such trusted subjects can do so only from specially designated DBMS processors which the security officer has selected for temporary System High operation. The security officer selects the access control policy for the DBMS processor. In conjunction with this mode of operation, an accurate security audit is conducted in real time and viewed by the security officer as each DBMS downgrade access occurs.

Figure 2.5-3 System Users and Thier Roles as Trusted Subjects

2.5.2 Trusted Processes

A trusted process is an application program or system process which cannot be certified or one for which certification is either not desirable or perhaps not required for security. In the architecture design for the secure DBMS, some trusted software may be used. For example:

- o the operating systems of the user host computers may be trusted to provide user separation (a form of discretionary access)
- o the processes used by the subject on a DBMS processor to reduce the security level of an object will probably have to be trusted (albeit closely monitored)
- o many of the processes of the Access Control Subsystem (ACS), (See Section 4.3), will be trusted. (These processes may be such things as discretionary access control, integrity control, audit trail maintenance. The threat in this processor is relatively low because it is not accessable to the user for program storage and processing. Should certain activities of the ACS be considered critical, validation of that activity can be conducted).

The trusted processes are minimized in the secure DBMS designs. There are tradeoffs possible between the cost of validation and the amount of trusted code permitted. Some applications may incur only a marginal risk by trusting a process while for other applications the risk might be intolerable. For example, trusting user processing separation to a DBMS processor incurs a (perhaps) tolerable risk in many applications, particularly if one of the improved, "secure"operating systems is used. When the risk cannot be tolerated, users might be assigned to personal DBMS processors - at great expense due to the currently high cost of processor replication.

2.5.3 Trusted Equipment

Much of the equipment in a computer system must be trusted. In those security applications where the hardware cannot be trusted, multiple redundancy is used to reduce the risk.

An example of trusted hardware is a microprocessor. Such a system has so many states that correctness of the microprocessor cannot be totally tested. Therefore, only the most likely system states can be tested. Physical failure of a microprocessor in a security critical path can be reduced by running redundant

processors as "reference monitors" and comparing the results of each processor for agreement.

Many security paths are of such a critical nature that redundancy is required to insure that no faults develop, despite the fact that the equipment is known to be good. In the secure DBMS design, the hardware security filter (Section 4.2) is such a device wherein a redundant system is employed.

2.6 Security Monitors and Audit Trails

Two essential security violation detection mechanisms in a computer system handling classified information are the security monitor and the audit trail.

2.6.1 Security Monitors

A security monitor is a process with the following tasks:

- o logs all security sensitive system activities in the audit trail
- o presents security relative information to the security officer, such as
  - current users & their current protection levels
  - protection privileges permitted for a given individual
  - all current accessed objects and their modes
  - protection level, and discretionary access levels of all stored objects
  - all modifications to protection levels or discretionary access permissions
- o sounds an alarm whenever an attemped security violation is detected.

Many of these actions are available to the security officer in real time. Certainly all of these actions should be included in the audit trail.

Security monitors can be of two basic types: embedded and isolated.

An embedded security monitor (Figure 2.6-1) is a part of operating system software and creates a trace of all commands processed. In a standard system, an embedded security monitor is located in the same processor as are the user application programs. Therefore, such a monitor is susceptible to tampering by a computer thief. In the secure DBMS architectures, an embedded monitor would

be placed in the access control subsystem, a relatively secure location because user access to this processor is restricted to DBMS queries. In a sense, this location of the monitor is not strictly embedded as it is isolated from the host (DBMS processor) equipment and user programmability.

The second type of monitor, the isolated monitor (Figure 2.6-2), resides in a totally separate processor which serves as a "listener" of all data base transactions. Because the logging operations are relatively simple, the software can be kept relatively small, facilitating validation (if desired). In the secure DBMS architecture, the security monitor would be attached to the access control processor. Because so much of the information being logged is acquired from the access control processor, many of the security advantages of the isolated monitor over those of the embedded monitor at this site are lost due to relatively tight system couplings. Therefore, an embedded monitor may adequately serve the secure DBMS in all but the most critical cases.

2.6.2 Audit Trails

The security audit trail records all security sensitive actions of the system. These activities can be described in terms of four system logs:

- o System Access Log - describes all subject log-on and log-off activities (Figure 2.6-3)
- o Protected Object Log - describes all accesses, successful or not, to protected objects (Figure 2.6-4)
- o DBMS Log - describes all DBMS alteration attempts (creates, deletes, permits, etc), successful or not. (Figure 2.6-5)
- o Suspected Violation Log - describes conditions surrounding a suspected security violation attempt and complete system state at time of attempt. (Figure 2.6-6)

The above logs should provide sufficient information for system analysis and for system security audits.

2.7 Multi-Level Security

Multi-level security is separated into two areas: subject and object.

The DBMS must support concurrent use by subjects of several levels.

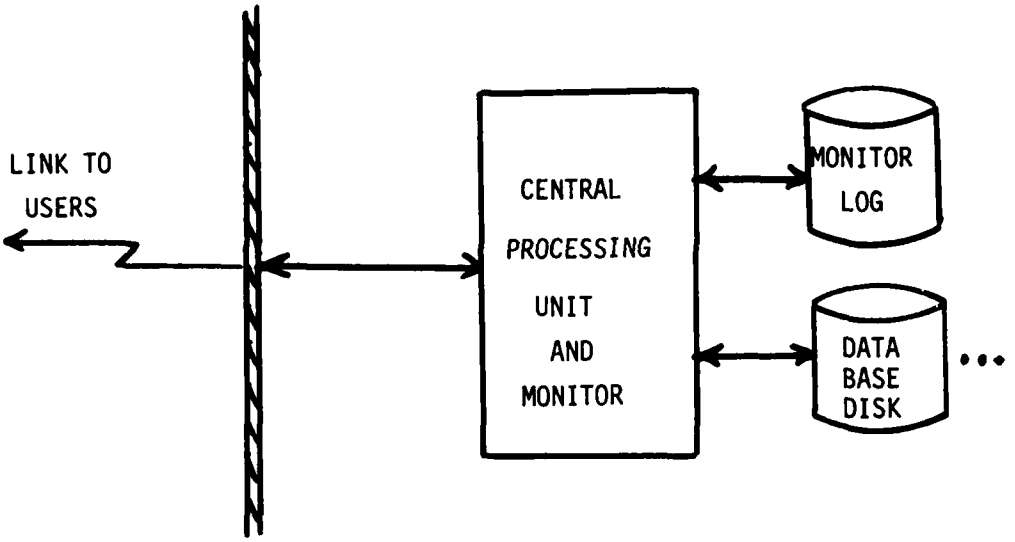


Figure 2.6-1 An Embedded Security Monitor

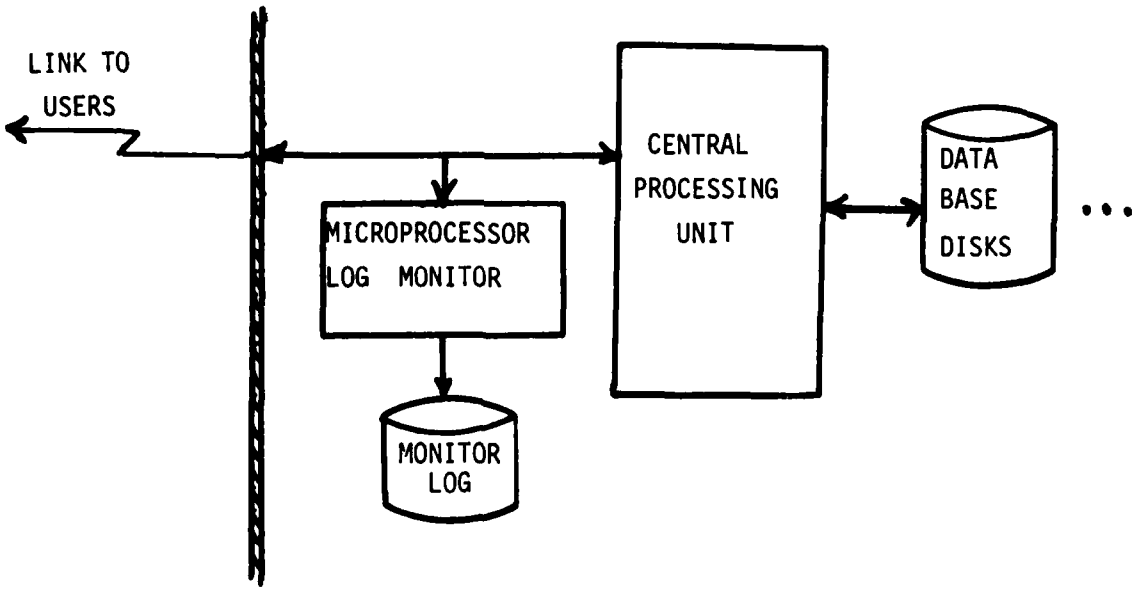


Figure 2.6-2 An Isolated Security Monitor

- o User identifier
- o Date/Time of log-on
- o Data/Time of log-off
- o DBMS Processor used identifier
- o Any resources used

Figure 2.6-3 System Access Log Description

- o Record of file opens and closes
- o User identification for all file accesses
- o Object access node (read, write, modify, etc.)
- o Identification of DBMS processor from which access was achieved
- o Protection level of subject
- o Protection level of object accessed
- o Date/Time of access

Figure 2.6-4 Protected Object Log Description

- o DBMS directory changes (CREATES, DELETES, PERMITS, RESCINDS, etc)
- o Subject identification for DBMS directory alteration
- o Protection level of subject at time of alteration
- o Protection level of object affected by directory change
- o Identification of DBMS processor from which change was initiated
- o Protection level of object after change (if level was changed)

Figure 2.6-5 DBMS Access Log Description

- o Date/Time of suspected violation
- o Identifications of suspected DBMS processor and subject generating the suspected violation
- o Type of suspected violation (illegal directory modification, object reclassification, access beyond user permission level, etc)
- o Complete system state dump (system operation is suspended upon encountering suspected violation except in the case of a trusted process downgrade operation)
  - dump of all active DBMS processor identifiers
  - dump of subjects currently active on each DBMS processor
  - dump of current accesses possessed by each subejct
  - dump of protection levels of each subject
  - a DBMS protection system activity including current access request in process and all pending system requests
  - short DBMS directory dump describing all objects on system, their protection levels, and their discretionary access permissions. If the DBMS system maintains last access data, such as last user to access, time, date, mode, etc, this too would be presented.

Figure 2.6-6 Suspected Violations Log Description

Although it is desirable that the subject not be burdened by a "single security level per session", multiple levels may not be practical: enforcement of the security policy with user changes in level is difficult, if not impossible, due to the requirement to sanitize the DBMS system prior to permitting a subject to go to lower security levels. Any security enforcement device, such as a system "sanitizer", residing in the user work area, such as the DBMS front-end host processors or terminals, cannot be ensured against user tampering and consequently cannot be totally validated. Consequently a system can be designated to permit processing by a number of users at different levels but only in a single level per session, single level per processor environment.

The DBMS may store objects of many different security levels. A subject must therefore, be able to read and write to objects at more than one level. The formal security policy permits this action within certain constraints. Basically, no non-trusted user can be permitted to access an object at one level, and subsequently write the accessed object at a lower level. Therefore the secure DBMS permits multi-level views of information to be read in such a manner consistent with the security policy and write only to those parts of the view consistent with the security policy.

2.8 Protection Granularity Discussions

Data can be protected at several levels of granularity. This section presents these levels of protection and some of the implications of their use.

Data Base

The classification of the entire data base to a single level is not a solution to the multi-level data base problem. The result of such a protection policy would be to create a system high machine.

Record, Row, or Tuple Protection

In a tabular representation of a data base, a record is a row of data, (Figure 2.8-1). Typically, a record contains information of many types. It is normally more profitable to assign like security levels to items of the same type as opposed to the related items of a record. For example, a data base might contain several unclassified entries, such as airfield name, number of personnel,

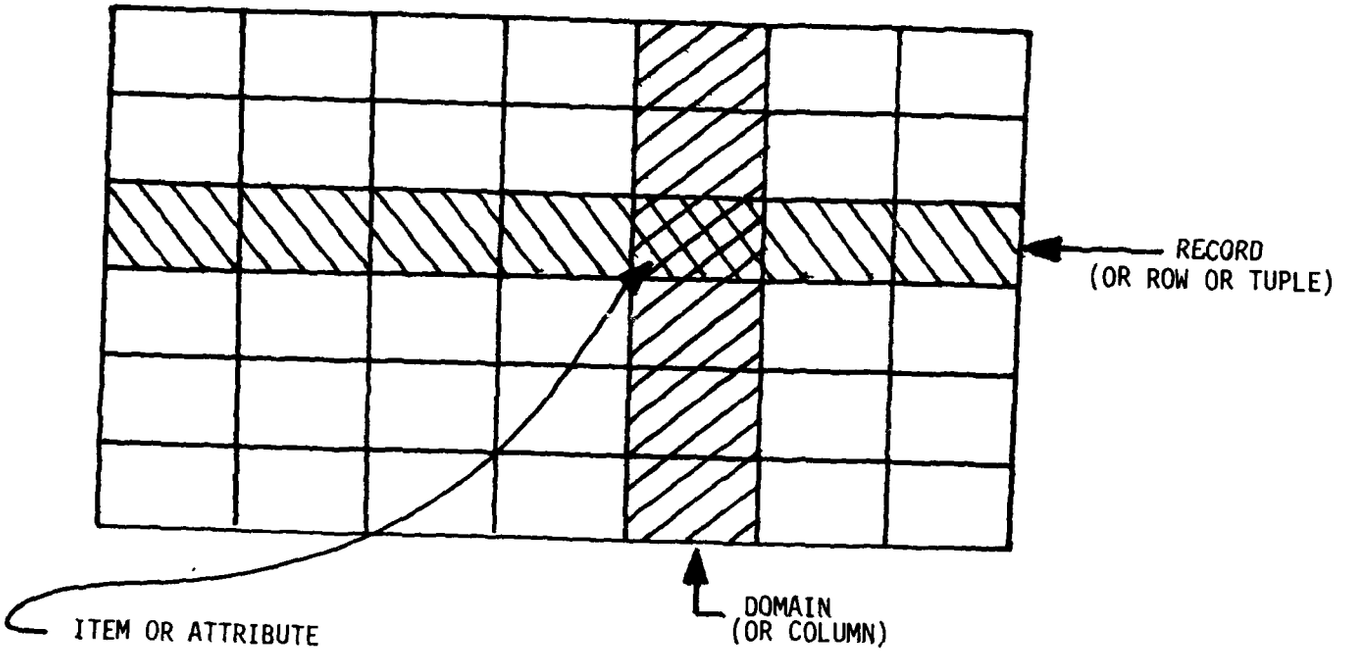


Figure 2.8-1 Relational Data Base Terminology

number of aircraft, and type of missiles deployed at that sight, as illustrated in Figure 2.8-2. It might be desired that the missile types be Top Secret. If the missile type is contained in each record, then all records must be protected at this same level, resulting in a single level data base.

Domain or Column Protection

In a tabular representation of a data base, a domain is a column of data (Figure 2.8-1). A column usually contains the same type of information and is a good choice for a single level of protection. Protection can be provided through a variety of mechanisms, including domain masks for a relation, item tags, or directory tags. See Section 4.1 for details of multi-level data protection.

Item Protection

An item or attribute is the intersection of a domain and a record in the tabular representation of a data base (Figure 2.8-1). The only practical means of providing item protection is through the use of data tags - security tags attached to every data base time. Care must be maintained in the method of maintaining the data tags. If a security label, i.e., a security level, can be altered through an upgrade process by a high level subject, the resulting security tag modulation can be observed from a low level, thus forming a covert channel. (An upgrade is the act of raising the protection level of an object). Figure 2.8-3 illustrates this process. Thus, the security tags must conform to the tranquility principle or must themselves be protected by a security rule. One such rule permits security tags to be set only from the lowest level of the protection level of the object, and once set, can be altered only from the lowest protection level of the object.

Other Granularities

While it is possible to protect data at a level lower than the item level by tagging subfields within the item or even words within the item, it is not practical. For example, if each word was protected and there were two bytes required to specify the protection level (security level and category; integrity level and category), an overhead of two protection bytes per word would be incurred!

RELATION: AIRFIELD DATA

DOMAINS	AIRFIELD NAME	NUMBER AVAILABLE PERSONNEL	NUMBER READY AIRCRAFT	MISSILE TYPES DEPLOYED
	AF1	700	50	
	AF2	3000	100	
	AF3	1010	70	
		1762	60	

DBMS KEY

U C S TS

CLASSIFICATIONS

DBMS KEY

Figure 2.8-2 Example Multi-Level Relation

ORIGINAL RECORD SECURITY LEVELS	U	U	U	U	U	U	U	U	U	U	U	U	U	U
RECORD SECURITY LEVEL AFTER MODULATION BY HIGH LEVEL SUBJECT	U	S	U	S	S	S	U	U	S	S	U	S	S	S
FIELDS VISIBLE TO LOW LEVEL	X		X				X	X			X			
MESSAGE CONVEYED	0	1	0	1	1	1	0	0	1	1	0	1	1	1

Figure 2.8-3 Illustration of a Covert Channel Through Data Tag Modulation

Summary

Figure 2.8-4 summarizes some of the implications of the various protection granularities.

For the purpose of the secure DBMS, the domain protection level was chosen as a practical level for implementation: cost is reasonable, protection granularity is moderate; implementation is feasible.

Details of methods of providing data separation in a multi-level data base and implementation implications are detailed later in Section 4.

2.9 Proof of Correctness Requirements

In order to prove that the secure DBMS is capable of providing the protection required by the formal security policy it must be validated through a security analysis. For the secure DBMS, flow analysis will be utilized to prove that sensitive information can pass only through the desired paths and that no covert channels exist. To do this, all system and subsystem interfaces must be fully described.

In addition to the system validation, the security policy must be shown to be sufficient and necessary for maintenance of system security. For this purpose, the formal mathematical proof of the security policy is provided in Appendix A and was summarized in Section 2.2.

In relation to system validation, Figure 2.9-1 defines terms used in validation and verification of systems.

PROTECTION GRANULARITY	COST			FLEXIBILITY	FEASIBILITY	PRACTICALITY (usefulness)	PROTECTION TECHNIQUE AVAILABLE
	S/W	H/W	STOR.				
DATA BASE	V.LOW	V.LOW	V.LOW	V. LOW	YES	V. LOW	NOT MULTI-LEVEL — SYSTEM HIGH MODE SECURE DBMS
RECORD	MOD	MOD	MOD	LOW	GOOD	LOW	DATA TAGS
DOMAIN	MOD/ HIGH	MOD/ HIGH	MOD	MOD	V. GOOD	V. GOOD	DATA TAGS, DIRECTORY TAGS, MULTI-FILE VIEW
ITEM	HIGH	V.HIGH	MOD to HIGH	V. HIGH	GOOD	V.GOOD	DATA TAGS
SUB ITEM	HIGH	V.HIGH	V.HIGH	V. HIGH	V. GOOD	V. GOOD (OVERKILL?)	DATA TAGS

Figure 2.8-4 Summary of Protection Granularity Techniques

Accreditation. The final acceptance of a system for operation in a specific environment. This is an administrative activity.

Certification. The application of policy doctrine and technical evidence to a system to determine the prudence of its use in a particular secure application. This is a technical and political activity.

Certification refers to the "user" agreement, at the conclusion of the OT&E (operational test and evaluation) phase of a contract, that the acquired system satisfies its intended operational requirements.

Correctness. Formally, the property of a system that is determined through formal verification activities. Correctness is not an absolute property of a system, rather it implies the mutual consistency of a specification and its implementation. See verification.

Correctness Proof. A mathematical proof of consistency between a specification and its implementation. It may apply at the security model-to-formal specification level, at the formal specification-to-HOL code level, at the compiler level, or at the hardware level. For example, if a system has a verified design and implementation, then its overall correctness rests with the correctness of the compiler and hardware.

Once a system is proved correct, it can be expected to perform as specified, but not necessarily as anticipated if the specifications are incomplete or inappropriate.

Implementation Verification. The use of verification techniques, usually computer-assisted, to demonstrate a mathematical correspondence between a formal specification and its implementation in program code. See verification.

Validation. The collection of evaluation, integration and test activities carried out at the system level to ensure that the system being developed satisfies the requirements of the system specification.

Verification. Informally, a clear and convincing demonstration that software is correct with respect to well-defined criteria, such as a security model. In a formal context, verification refers to the mathematical demonstration of consistency between a formal specification and a security model (design verification) or between the formal specification and its program implementation (implementation verification). The phrase "formally verified" is now beginning to imply that computer-assisted techniques have been employed in the verification effort. See correctness.

In testing, verification refers to the iterative process of determining whether the product of selected steps of the CPCI-development process meets the requirements levied by the previous step.

Figure 2.9-1 Terminology for System Accreditation

3.0 SYSTEM ARCHITECTURE AND ELEMENTS

This section describes the general architecture and subsystems needed to implement a distributed secure multi-level DBMS system: This DBMS system is capable of supporting groups of users at a common access level on dedicated processors sharing a common multi-level central data base. This section has the following subsections:

3.1 Concept
3.2 Implementation Architecture
3.3 Subsystem Descriptions
3.4 Relational DBMS Approach
3.5 Alternate Approach

3.1 Concept

The enforcement of non-discretionary security between multiple dedicated processors and a common multi-level data base can conceptually be thought of as a crossbar switch (as shown in Figure 3.1-1) with the connectivity limited by the simple and *-security rules. Note that data in this multi-level data base must be logically separable by security classification to enforce non-discretionary security.

For some multi-level, multi-compartment applications the large number of dedicated processors required may economically limit the application of this architectural approach. As this approach is compatible with period processing, the number of processors attached to the central data base can be limited to the number of concurrent access categories required with the potential connectivity appropriately changed between periods.

Ideally, the enforcement of non-discretionary security would be done by a certifiable hardware security filter (HSF). Such an HSF would greatly aid system certification as non-discretionary security enforcement validation would rely on hardware flow control analysis. That is, a multi-level mode of operation could be achieved without requiring a secure operating system or certified/trusted software.

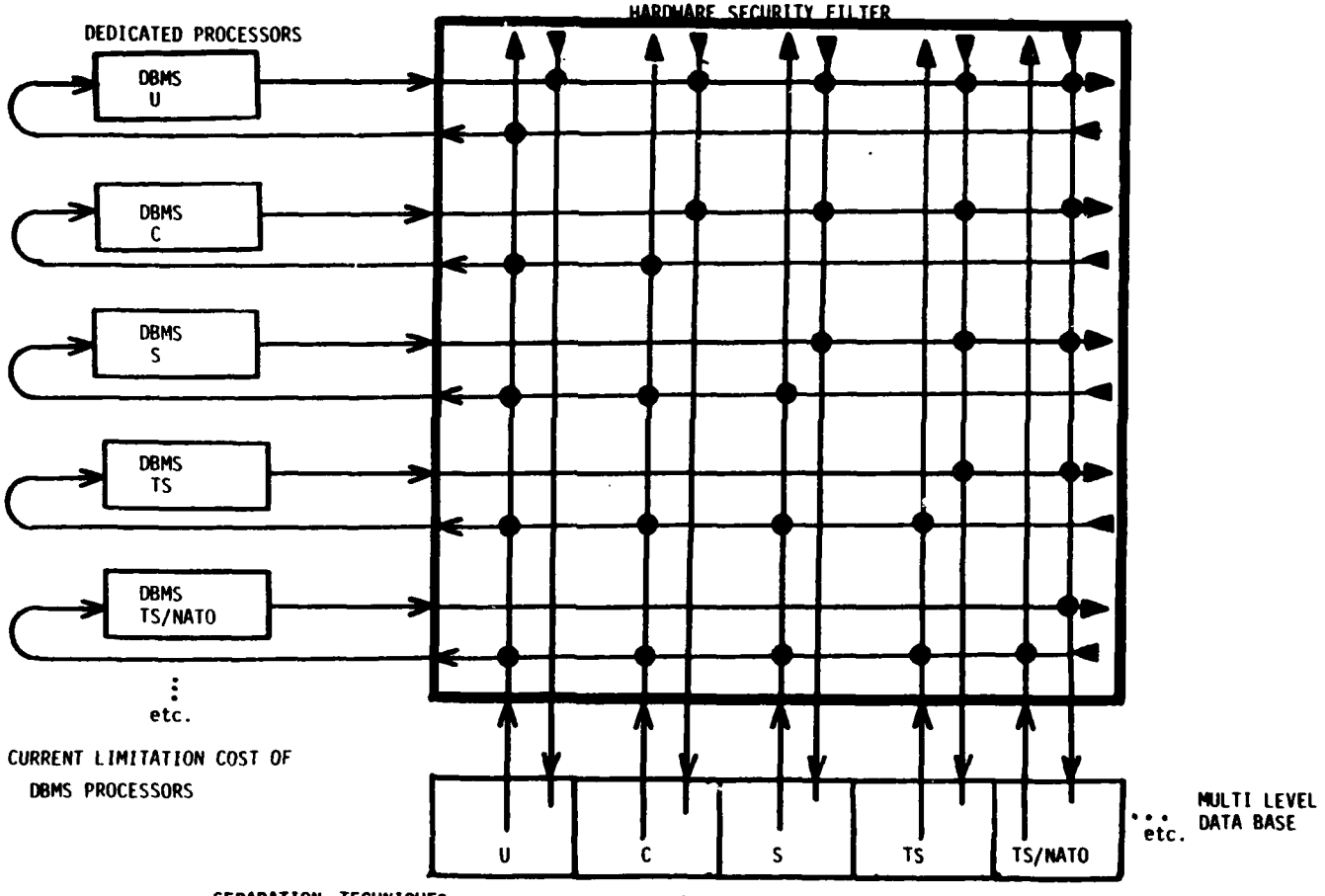


Figure 3.1-1 Cross Bar Switch Concept

3.2 Implementation Architecture

The architectural implementation of this concept is shown in Figure 3.2-1. This block diagram shows multiple dedicated DBMS processors subsystems connected to a central, multi-level data base subsystem. The DBMS processor and central data base are separated by an intervening hardware security filter (HSF) subsystem that is front-ended by an access control subsystem (ACS). The ACS handles the contention among the DBMS subsystems for central data base accesses and other central processing functions. Because the ACS control outputs are screened by the HSF against processor access control policies (ACP) supplied by the security officer and its other interfaces are receive-only request lines from the DBMS subsystem, it acts as a system high observation node. Hence, no trusted or certified ACS software is required.

3.3 Subsystem Descriptions

The central multi-level data base resides in mass storage. The data stored within this central data base must be logically separable by classification (security level and compartment). The central data base is connected to the HSF.

The HSF acts as a central data base entrance and exit guard. It receives central data base access requests through the ACS. It receives an ACP for each processor from the security officer. It enforces this ACP for each DBMS processor by separating data by security category and by controlling which categories may be stored and/or retrieved by each DBMS processor.

The ACS handles central data base access contention among the DBMS subsystem, queuing those requests which cannot be immediately serviced. Because it is the only central system high observation node it performs a large number of central processing tasks. The ACS performs the central audit trail and security monitor functions. The ACS also handles the central data base file management (storage allocation), i.e., it implements directory commands. Because this allocation must occur at a system high level that cannot be viewed by low level DBMS processors, this subsystem must receive logical central data base access requests from the DBMS subsystems and convert them to physical central data base requests for the HSF.

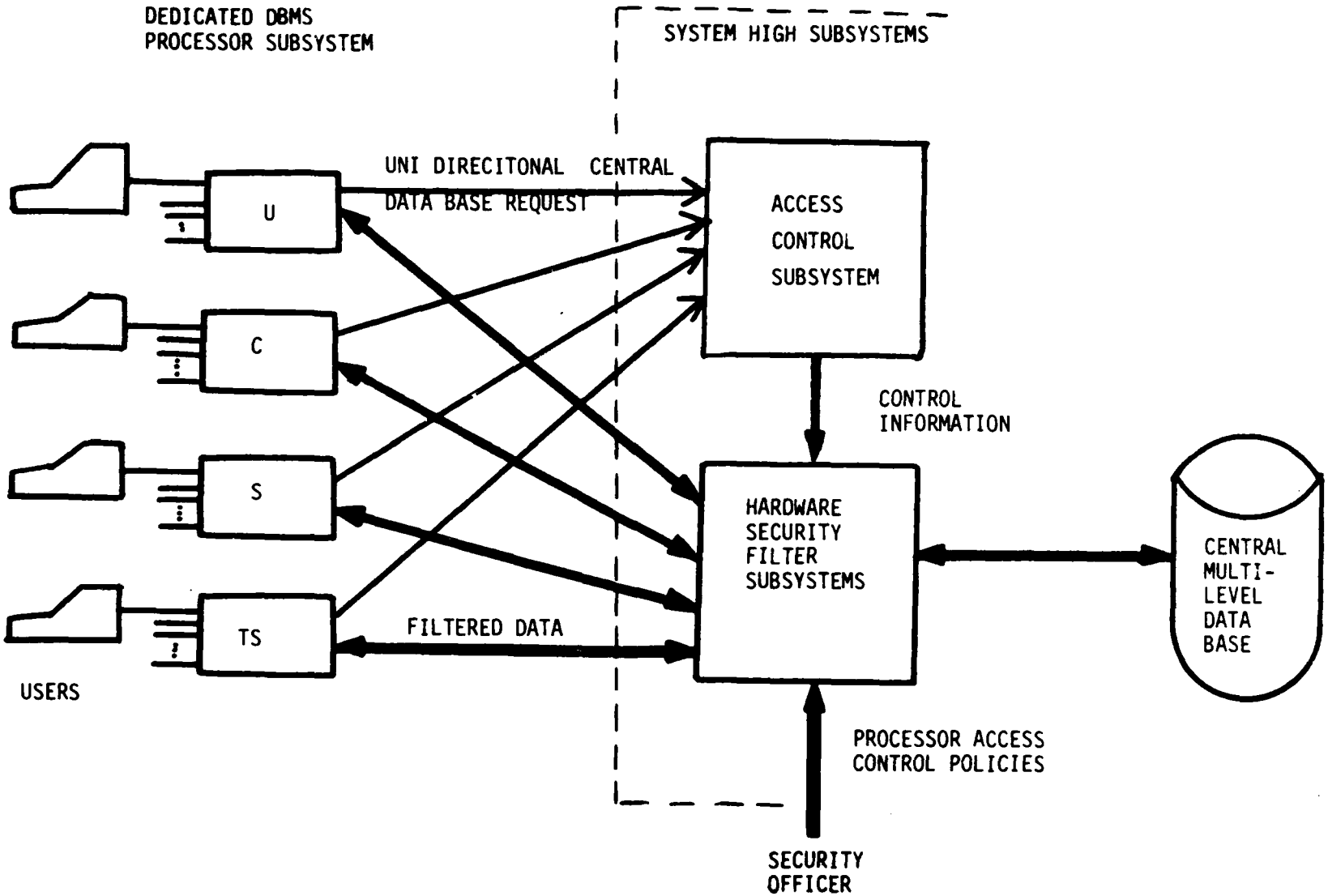


Figure 3.2-1 System Block Diagram

The ACS is also the preferred location for integrity and discretionary access control software as it is a central location which cannot be bypassed and it provides physical isolation from potentially hostile user code. It is also desirable to locate some maintenance software in this subsystem to perform fault detection/isolation and DBMS maintenance support. A more detailed functional partitioning diagram of this subsystem is shown in Figure 3.3-1.

The DBMS subsystems may either be comprised of general purpose processing systems and/or back-end DBMS machines. The general purpose processing systems will directly support user application programs. The back-end DBMS processors provide an interactive query language user interface. In the back-end DBMS configuration, another (host) processor is therefore required to run application programs.

3.4 Relational DBMS Approach

For this study a relational DBMS approach was selected for detailed study. While other DBMS approaches, e.g. Hierarchical or CODASYL, are not incompatible with this architecture, the relational approach has the advantage of requiring relatively little direct data coupling. Because of this feature it is easier to segment the data base by security classifications.

Figure 3.4-1 shows some of the features of a relational DBMS. In this type of DBMS the data base is broken into units called relations which can be visualized as tables with rows called records, columns called domains, and the intersection of a row and column called a data item or attribute. In the example shown there are two relations: an employee relation and a department relation. The employee relation contains domains called: Name, Sex, Age, Salary, Job Title, and Department Number. The employee's name is the key to this relation with the other domains dependent upon that key domain. The department relation has multiple keys with the department number as the primary key and department title as a secondary key. Note that an employee's record contains his department number, a key to the dependent relation. This foreign key allows the DBMS to construct a joint view of multiple relations. For example, a list of employees and their directors can be created as a user view.

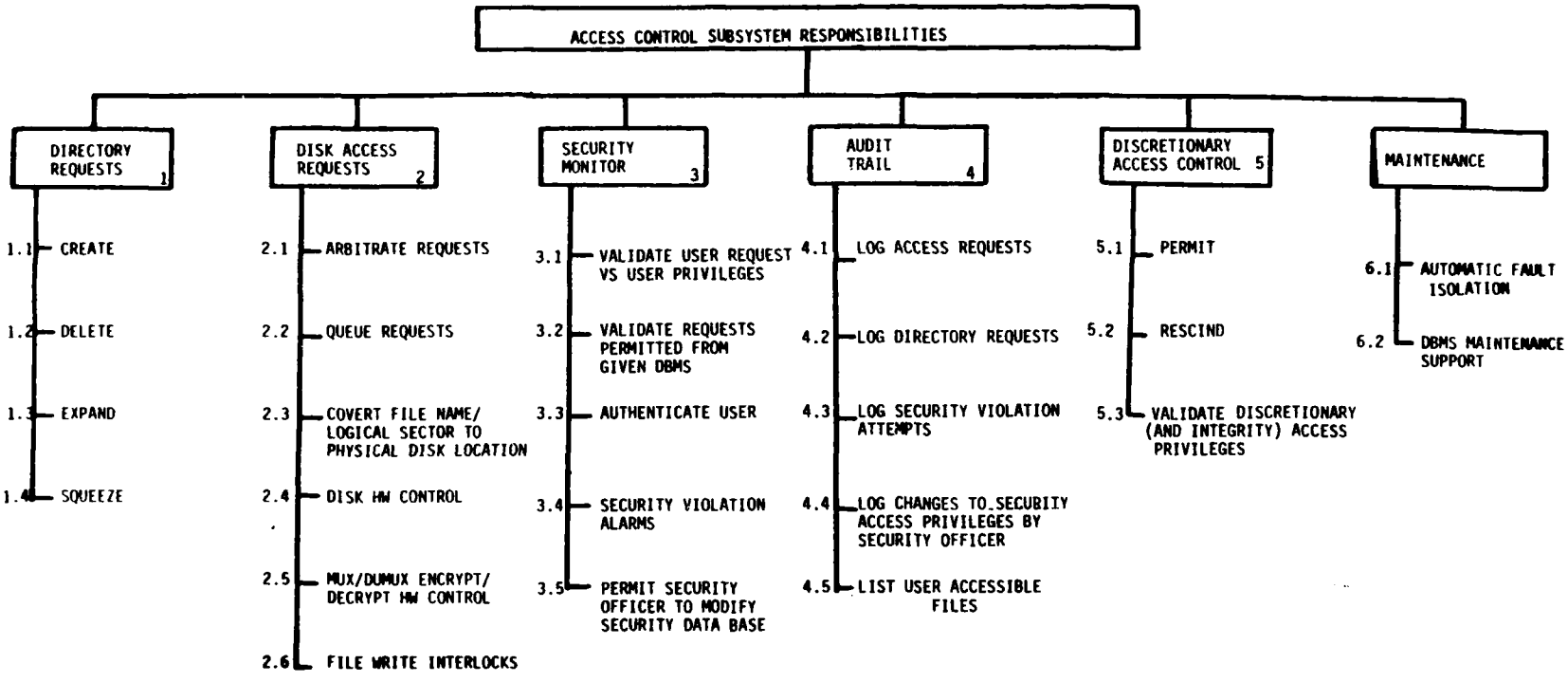


Figure 3.3-1 Access Control Subsystem Functional Partitioning

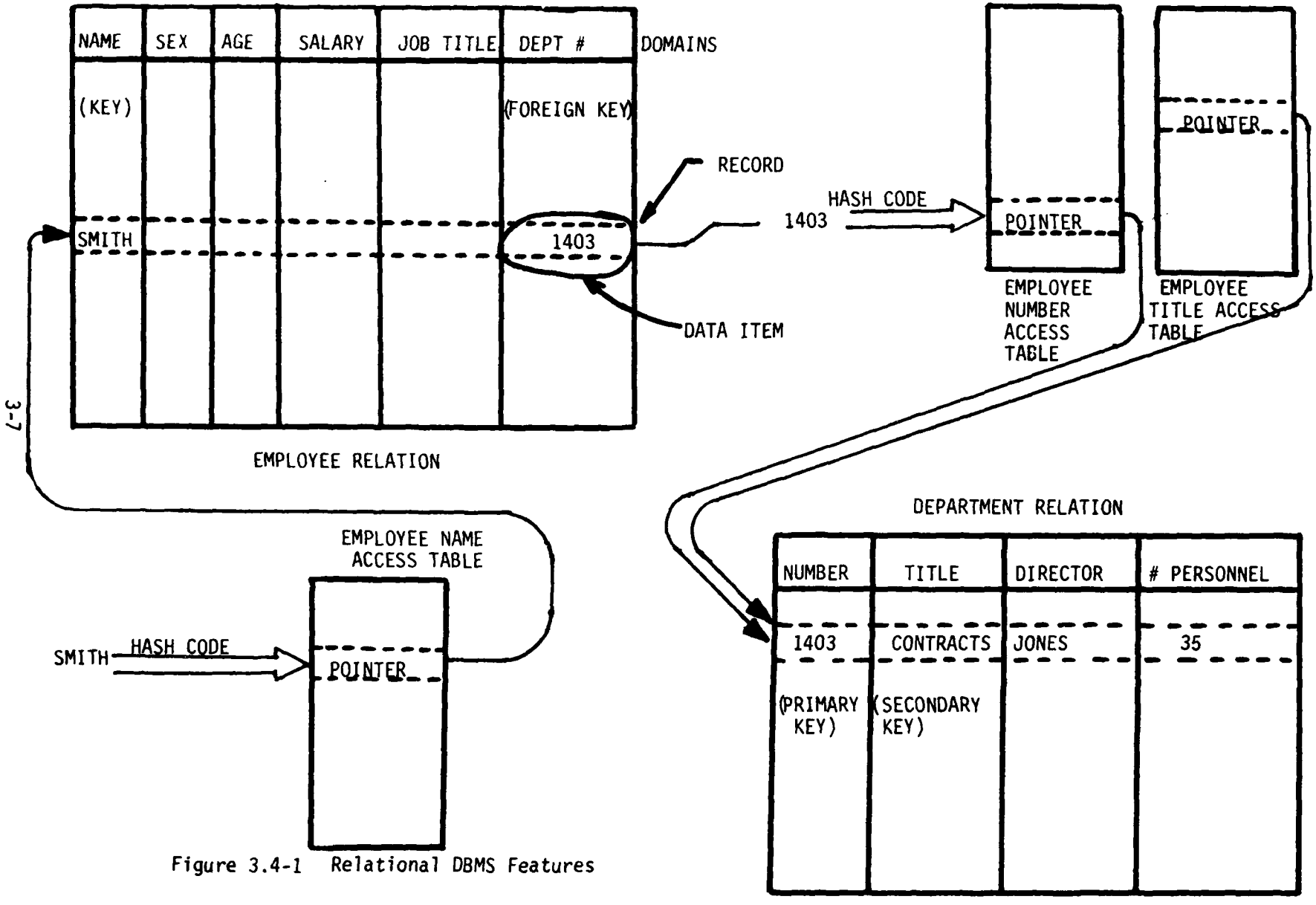


Figure 3.4-1 Relational DBMS Features

To decrease access time, a hierarchical, partially inverted access structure is typically used to allow indirect random access of records by key data items. For example, when Smith is hired his employee record is added to the employee relation. At the same time, his name is hash coded to form an index and a pointer to his record is stored at that indexed location in the employee name access table. Thus "What is the name of Smith's department?" query could be processed by the DBMS as follows: the name Smith is hash coded and used to index his record pointer. His record is then retrieved and his department number removed. His department number, 1403, is then hash coded to index the pointer to his department's record. His department's record is subsequently retrieved and the name of his department found.

In contrast, "List the name(s) of all directors under age 45" requires a combination of sequential and indirect random access record retrievals. In this case each department record is sequentially accessed, the director's name is removed and used to index his employee record containing his age.

Actual processing would be somewhat more complicated. In the above example, the director's date of birth would actually be stored, with age computed using the current date. Another typical DBMS complication occurred when Smith was hired by department 1403 thereby increasing the number of personnel in that department. Thus updates are required to the department data base as well as the employee data base when employees are added, deleted or change departments. In essence, an addition, deletion or change to the department domain in the employee relation should trigger an associated update in the department relation. The actual mechanics associated with relational DBMS access structures can be quite involved and are beyond the scope of this discussion. The above relational data base descriptions are presented only as an introduction to the data base process. The subject is covered amply in the literature.[3, 4, 5]

3.5 Alternate Approach

An alternate distributed architecture approach to multi-level security is to place the security filter between user terminals and the central data base. This security filter then enforces security at the user (terminal) level rather than the processor level. The DBMS functions are handled by an isolated back-end processor. This approach provides secure multi-level DBMS processing for interactive query users. A host processor is required to support any application programs. If such a host processor is shared concurrently by many users, then the security provided by the distributed architecture reverts to the processor level.

3.5.1 Concept

The key to this alternate approach to a distributed secure multi-level DBMS architecture is to isolate the DBMS functions from the users. This is accomplished by forcing direct data transfers to be made between the user terminal and data base through an intervening security filter. Figure 3.5-1 shows this conceptual architecture. A back-end DBMS processor is used to process queries and create access structures. However, this DBMS processor is isolated from the users and may only make indirect responses to user's request. The DBMS acts as a translater converting user queries into central data base requests which are processed in turn by the security filter. Furthermore, all data generated by the DBMS processor and stored in the central data base such, as access structures, is stored in a special security compartment that is inaccessable to users. This isolates DBMS generated data from the users and prevents use of the central data base as a covert channel from the system high DBMS processor to the users. Thus, in essence, the back-end DBMS processor is a system high observation node and therefore does not require certified or trusted software.

3.5.2 Implementation Architecture

Figure 3.5-2 shows the implementation of this alternate architecture. A hardware security filter subsystem is still used to enforce non-discretionary security. However, it now connects a single system high back-end DBMS subsystem and multiple single-level front-end I/O processors to the central multi-level data base. Another difference is the addition of a multi-key end-to-end encryption (E^3) system between the classified front-end I/O processors and the HSF. The user key for the E^3 device is selected to enforce discretionary security. Similarly, single

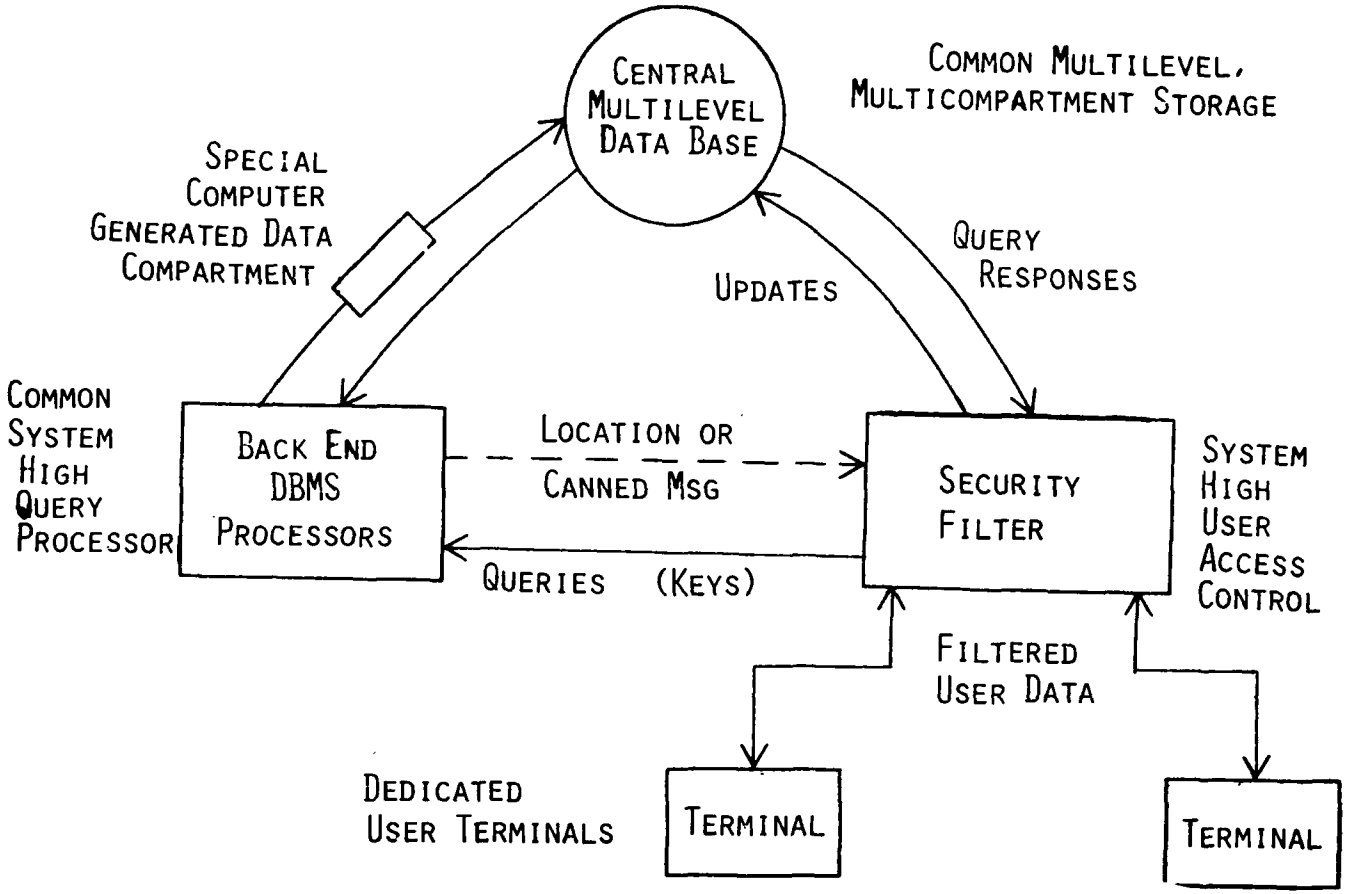


Figure 3.5-1 Alternate Architecture Concept

key E^3 systems are located within users terminals, thus providing hard encryption protection of all user data passed between the users and the HSF.

Clear text (unencrypted) queries are routed from a user's terminal, bypassing the E^3 system, through the front-end I/O processor to the system high back-end DBMS subsystem. The DBMS subsystem finds the central data base location associated with the request which it passes to the front-end I/O processor. This specified location may contain an unclassified canned message, e.g., "item not found". This processor then relays the access requests to the access control subsystem where discretionary security is enforced by encrypting the query response in the user's key.

A similar approach is used for updates, except that the data generated by the DBMS is placed by the HSF into a separate security compartment. This special compartment is made inaccessible to the front-end I/O processor by the HSF access control policy, thereby denying direct user access to the result of the modification. To prevent a covert channel between the DBMS to the end user, certified or trusted code is required in the front-end I/O processor. However, note that these processors cannot compromise discretionary or non-discretionary security by themselves even with trap door software.

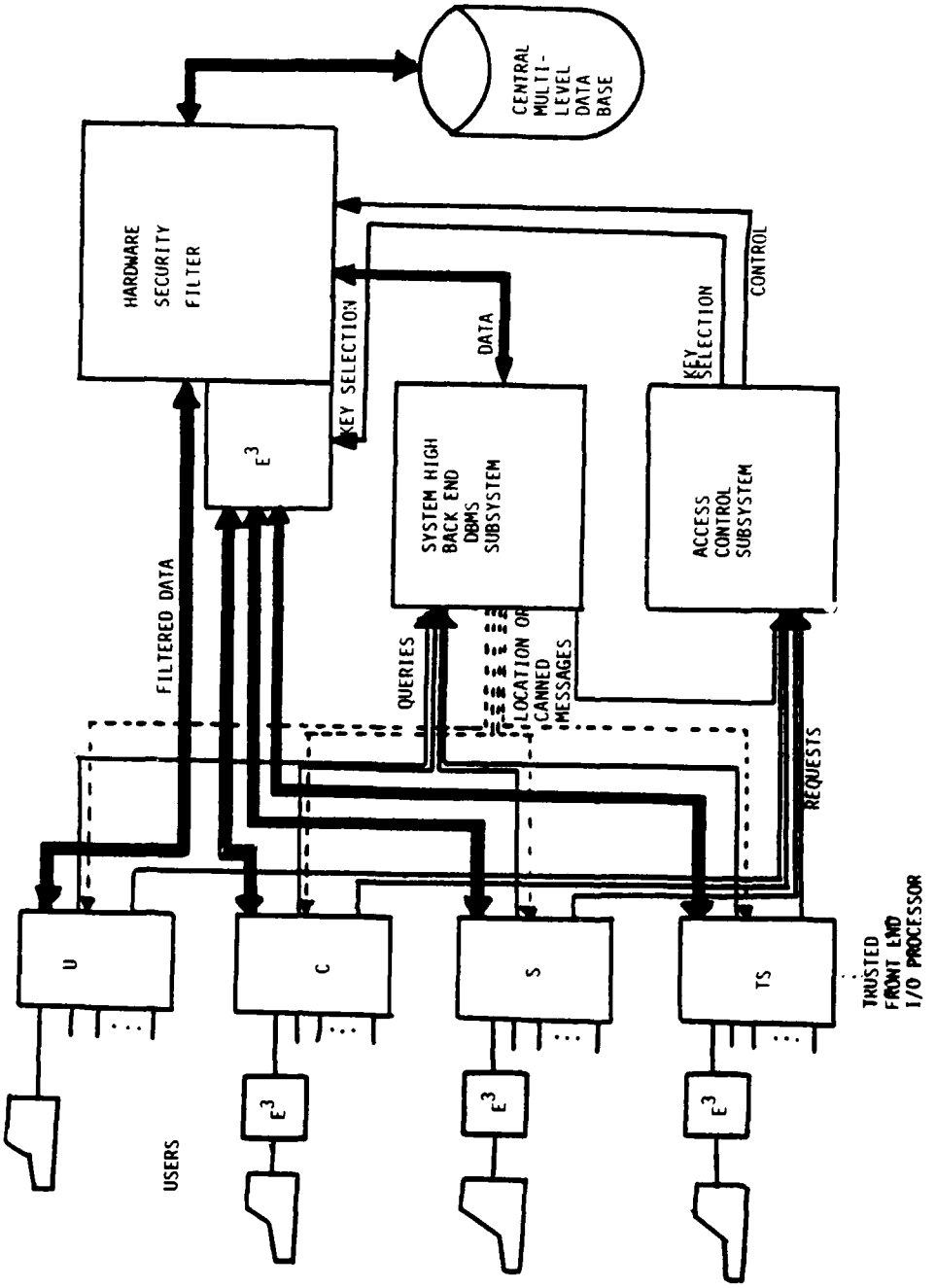


Figure 3.5-2 Alternate Secure DBMS Architecture Block Diagram

4.0 IMPLEMENTATION TECHNOLOGIES

This section describes the technologies capable of implementing a multi-user, multi-level secure DBMS. In particular, the following major areas are discussed:

Section 4.1 Relational Approaches to Multi-Level DBMS with Domain Level Protection

Section 4.2 Hardware Security Subsystem

Section 4.3 Access Control Subsystem

Section 4.4 Encryption

Many of these implementation technologies are integrated in the architecture designs of the following section, Section 5, Architectural Approaches.

4.1 Relational Approaches to Multi-Level DBMS with Domain-Level Protection

In section 2.8 the granularity level of protection was discussed and the use of domain level granularity was shown to be a good choice. This section details the methods of providing multi-level security in a relational DBMS.

4.1.1 Approaches

There are at least four ways of handling multi-level relations within a distributed secure DBMS architecture of single level DBMS machines:

- o Domain Separation
- o Black Bypass
- o Internal Bypass
- o Multi-Relation View

The Domain Separation technique is a software only approach. It places domains with different protection levels in different files, maintaining access control at the file level (see Figure 4.1-1) This approach requires custom DBMS software as a relation's domains must now potentially reside on multiple files, some of which may be unaccessable at some security levels.

A second approach is the Black Bypass technique (Figure 4.1-2) This approach requires physically tagging data with its security classification. The Hardware Security Filter Subsystem screens these tags on the fly against the access control policy. A black bypass to the Hardware Security Filter is used to allow access to any level of classified data in an encrypted form. This capability

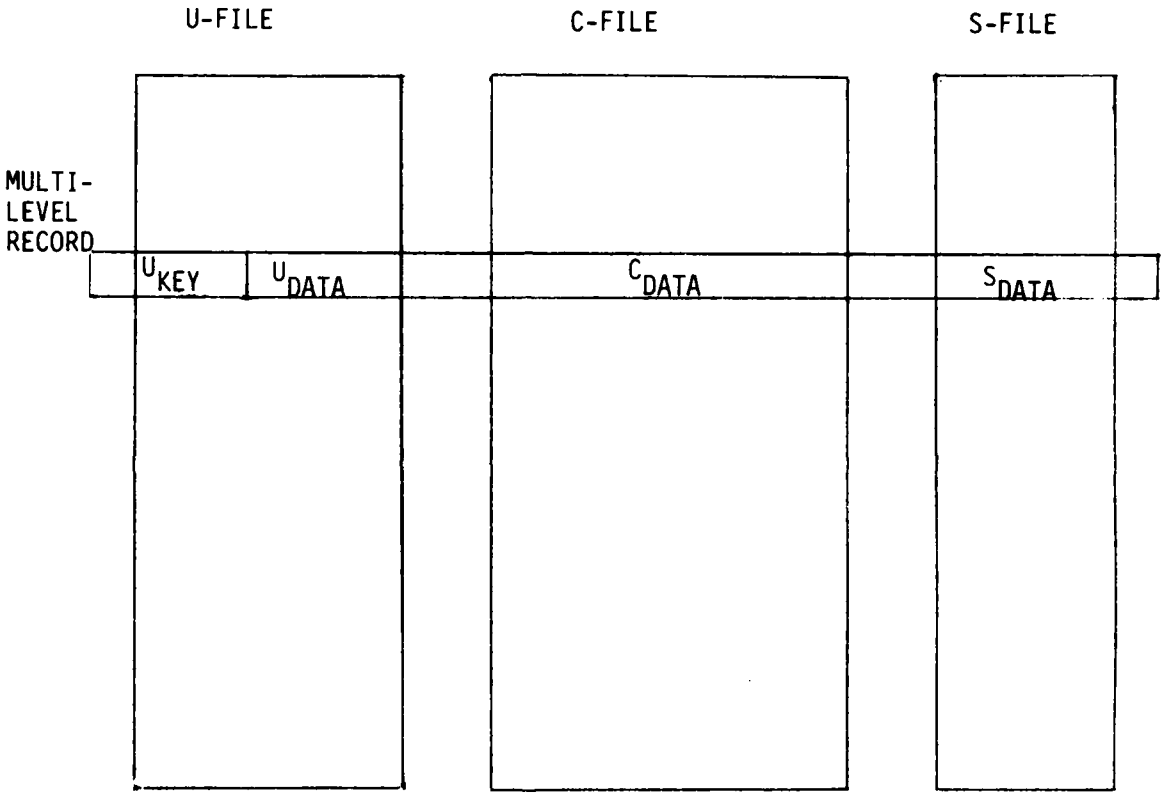


Figure 4.1-1 Domain Separation for Multi-Level DBMS

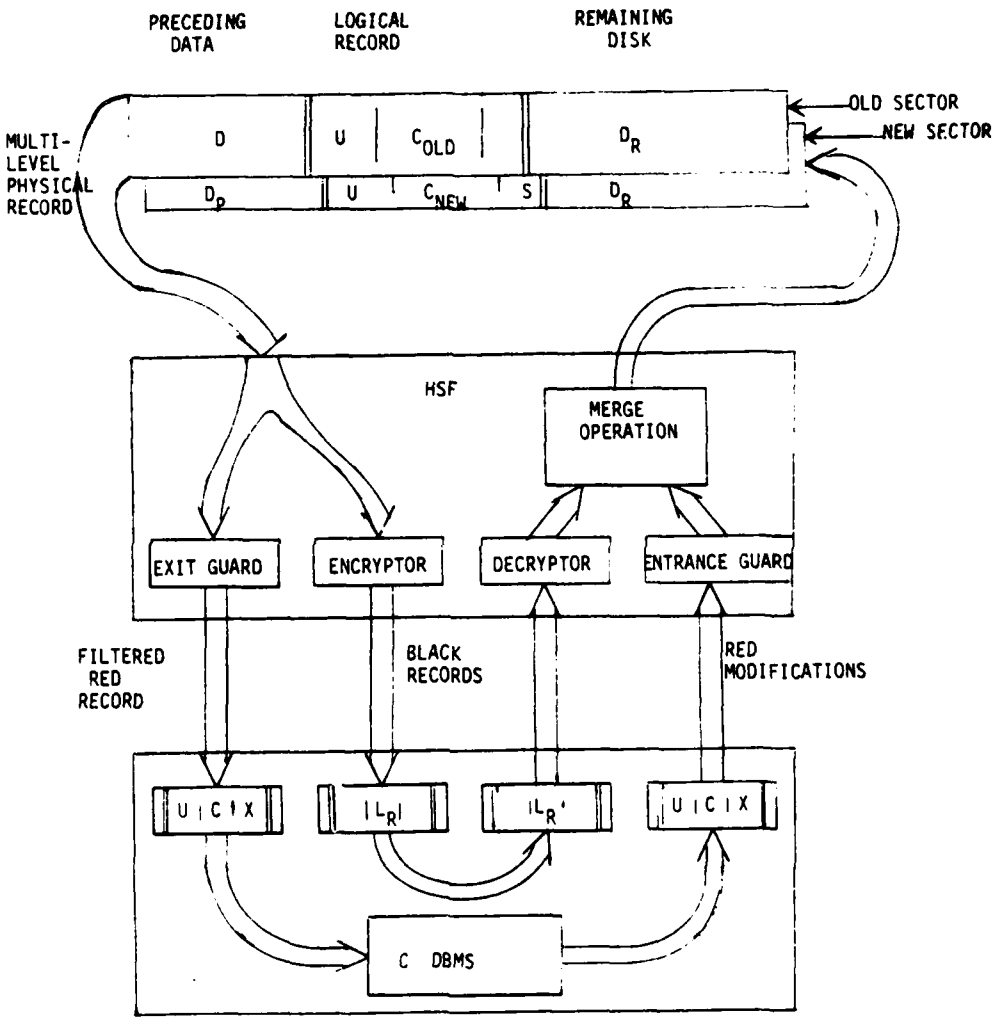


Figure 4.1-2 Black Bypass Approach for Multi-Level DBMS

permits the Hardware Security Filter to take mixed red (unencrypted) and black (encrypted) data from a DBMS processor with only the red data security classification screened against access control policy on the fly. This approach allows a DBMS processor to make changes at its security level to a multi-level physical record without compromise to lower or higher level data in this physical record. This technique requires an extensive increase in Hardware Security Filter Subsystem requirements and extensive modification to the DBMS software to handle security tags, reading red or black physical records, and merging red changes into black physical records.

The Internal Bypass technique is the third approach (Figure 4.1-3). This is a variation on the Bypass approach technique that does not require routing black records througha DBMS. Instead, a physical bypass is placed in the Hardware Security Filter Subsystem that allows a record to be read and stored within the Hardware Security Filter Subsystem. That portion of the record permitted to be accessed is subsequently read by the DBMS processor which can then mark changes to be merged with the old record temporarily stored in the Hardware Security Filter Subsystem. Only these changes are then screened against access control policy. The advantage of the internal bypass is the removal of tne complex encryptor/decryptor bypass mechanism in the Hardware Security Filter Subsystem. This technique does require changes to the Hardware Security Filter Subsystem to manage the bypass buffer and to the DBMS software and OS to handle security tags and to mark changes.

The fourth separation technique is that of the Multi-Relation View approach (Figure 4.1-4). This approach divides multi-level relations into multiple, single-level, single compartment relations with the key(s) replicated in each relation. A view is then used to manipulate the total super-relation. This approach requires no system operation modification. However, it does have application impacts on the DBMS scheme and on the addition or deletion of records. The largest disadvantage is the potential loss of data correctness i.e., the potential of having a partial record due to not deleting or adding a subrecord (identified by a key) in <u>all</u> subrelations of the super-relation.

Besides the additional complexity required by most of these approaches, there are also DBMS performance penalties associated with them. In particular:

- The Domain Separation and Multi-Relation View approaches require reading multiple files for retrieval of a single multi-level record.

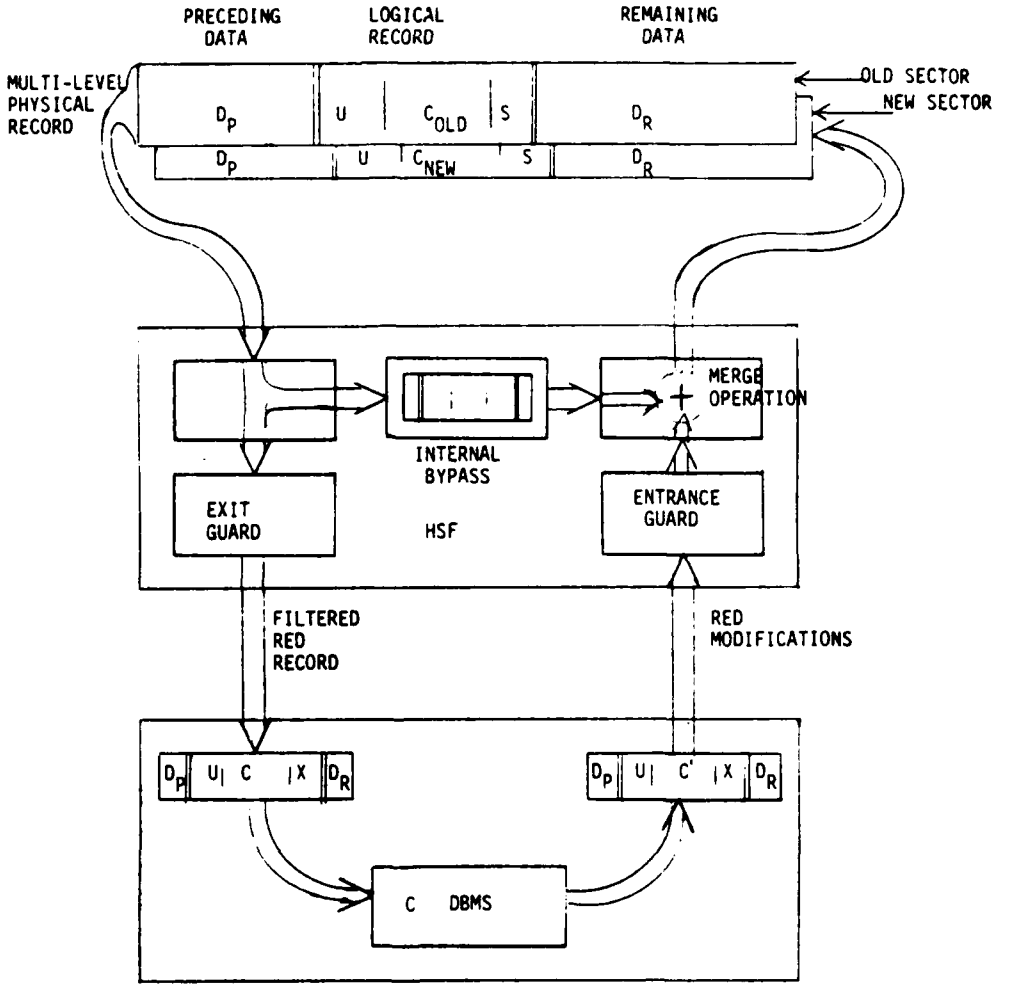


Figure 4.1-3 Internal Bypass Approach For Multi-Level DBMS

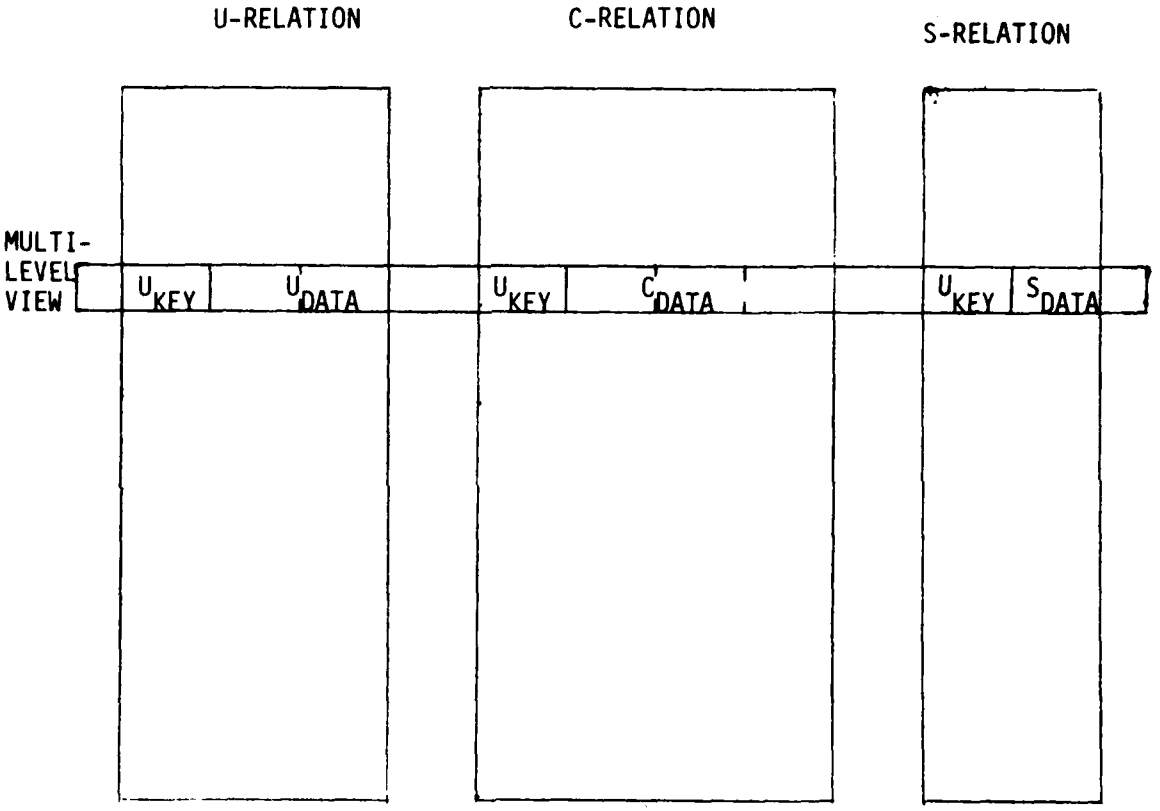


Figure 4.1-4 Multi-Relation View Approach For Multi-Level DBMS

- o The Black Bypass approach requires reading each multi-level record twice (once for the black data transmitted and once for the transmitted red data) prior to an update.
- o The Black Bypass and Internal Bypass approaches both require adding physical security tags thus reducing effective central data base throughput and packing density.
- o The Black Bypass approach can create a high speed covert channel between DBMS processors. In particular, an unclassified DBMS processor can create an unclassified file which a higher level processor can subsequently modify the data items, upgrading selected items in a pattern determined from the bit patterns of a high level object. For example, an upgraded field can represent a "one" bit, a non-upgraded field can represent a "zero" bit. The lower level processor can then read the modulated security flags. (See Discussion in Section 2.4)

In addition to these DBMS performance penalties to effective central data base access times and capacity, there are large DBMS processor penalties in terms of the additional time to handle each request.

4.1.2 Functional Restrictions

The requirement for handling multi-level relations results in some functional restrictions on the record access structure of the DBMS.

The primary concern is, "What is the level of computer generated structures that go along with the user generated structures (e.g., schema and records)?" In particular, relational DBMS typically uses a hash code table to index records by key with records stored in a linked list format. This approach allows fast random access by key through a hierarchical partially inverted structure, sequential access through a linked list, and the capability of relatively fast record addition or deletion without expanding or contracting the other records through use of an empty record queue (linked list).

From a practical standpoint, these computer generated structures (e.g., hash code tables, free queue pointers, link list pointers, etc.) must be at the lowest level of the relation. If not, the DBMS will be unable to locate logical records from that security level, effectively denying access. For this same reason, the schema (logical record structure) must also be at the lowest level of a relation. Finally, the DBMS key(s) must also be at the lowest level of a multi-level relation. This is because the key(s) are the effective index used for record deletion and addition. In particular, for additions, a computer hash code table entry that contains information about the key(s), must be generated at the lowest level of the relation.

Because the keys are maintained at the lowest level, some additional restrictions on DBMS processing result. Those operations which must create or delete keys and pointer structures must be performed at the same level as the DBMS structure, i.e., the lowest level of the DBMS. Therefore, additions and deletions must be made from the lowest level of the relation. This process requires the ability of the subject to write information to higher protection levels if multi-level relations are to be used or use blind writes (updates).

There is an exception to the above requirement in the case of the multi-relation view approach of multi-level record handling. In this case, the same key as used in the lowest level sub-relation is replicated in the higher level sub-relations, thereby creating the necessary access paths for additions or deletions.

One final restriction exists if higher level additions and deletions are to be permitted by a lower level user to higher level objects. In order for the low level pointer structures to permit proper location of the item to be processed, the higher level items must have a fixed field size.

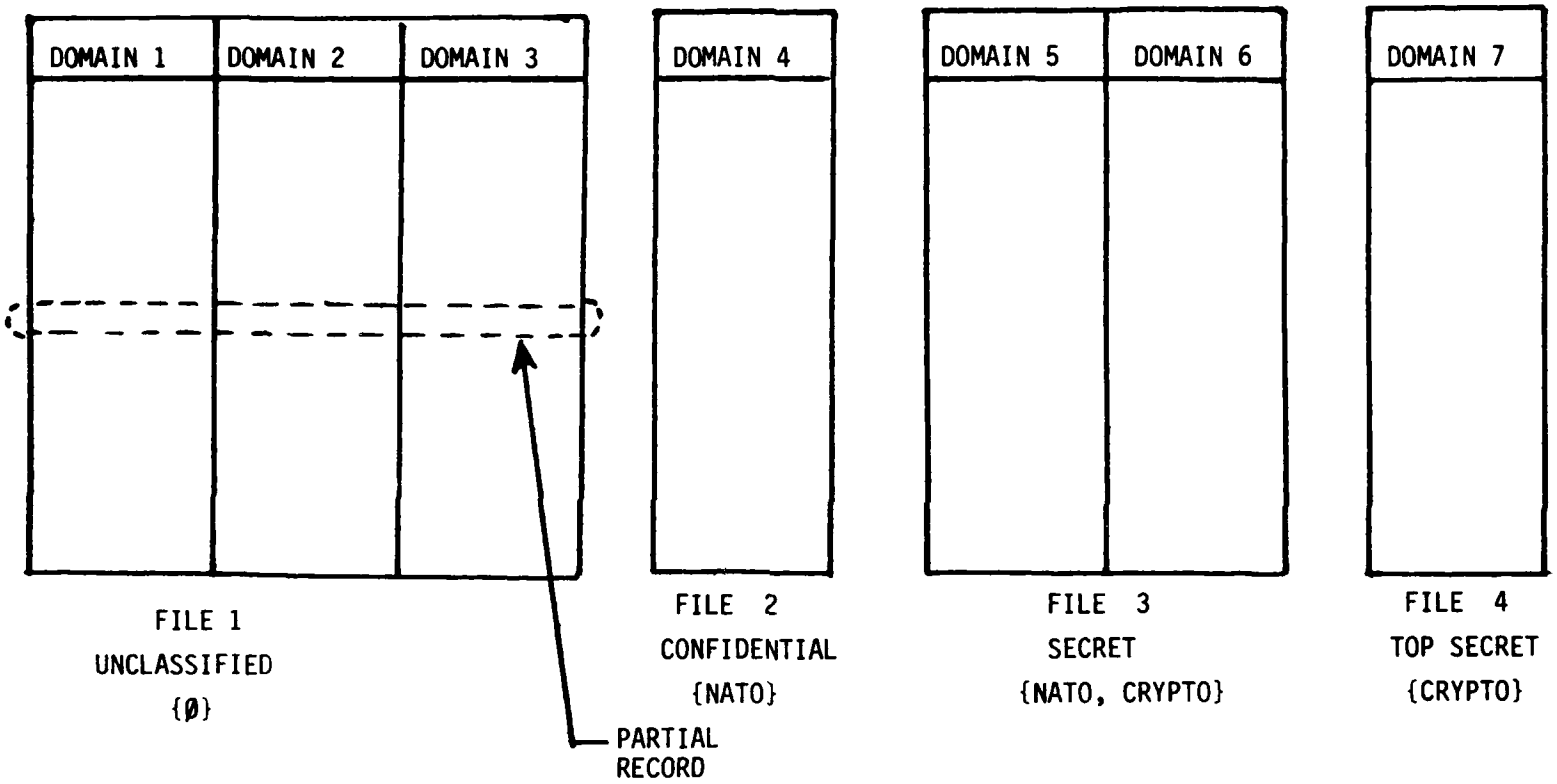
4.1.3 Domain Separation Approach Description

One approach to handling secure multi-level domain relations is to develop a specialized DBMS that segments data by the protection level of its domain and stores each segment in a separate file. (Figure 4.1-5). This technique has a minimal impact on any of the distributed architectures and provides good security. However, this approach requires development of a DBMS customized to handle multi-level security and has an adverse performance impact due to requiring multiple files to be read for multi-level record access.

DBMS Impact

Basically, the Domain Separation method requires that the schema be modified to permit a security clearance to be established for each data domain. Files at the lowest level and compartment formed from the intersection of all of the compartments (hence referred to as the Common Compartment of the relation) of the domains must contain the keys and associated computer access structures, the relation's parameters (e.g., hash code table lengths, free queue pointer, etc.), and the schema. There must be a separate file for partial record storage of each set of domains with a different clearance. These "higher level" partial records must be directly or indirectly accessed through the computer access structures at the "lowest level". If they are to be directly accessable to lower level subjects for appends or deletes, these higher level partial records must be stored in a flat fixed-length manner. If "higher level" variable length fields are desired, these partial records must be indirectly accessed through a flat pointer table and an empty partial record queue maintained at that higher clearance level.

There are three methods of handling relation initialization (creation) and record additions and deletions for Domain Separated multi-level relations. The first technique requires that these operations be done at the lowest level and at the common compartment of the domains of the relation. This is to permit access to write (add or delete) to the key structure of the relation. Initialization and deletion should blank (destroy) previous record data at all levels. This can be done from the lowest level if a physical record (disk sector) can only contain (all or part of) one partial logical record and "higher-level" partial records are



DOMAINS ARE SEGMENTED INTO FILES ACCORDING TO PROTECTION LEVEL. FOUR ACCESSES ARE REQUIRED TO RETRIEVE A SINGLE MULTI-LEVEL RECORD FOR THIS EXAMPLE.

Figure 4.1-5 Domain Separation Technique For Multi-Level Relations

are stored in a flat fashion, thus permitting blind updates. However, the use of flat records does not allow records to be packed on disk and therefore degrades disk packing density and throughput.

A second approach for initialization and deletion is to relay the command to a higher level machine which performs the higher level portion of the activity. This approach requires upward communication among the DBMS processors. However, if this communication path is provided, it would be more desirable to modify an existing DBMS system and split these multi-level relations into multiple single clearance relations with the key(s) replicated. (See Multi-relation View Approach.)

A third approach is to ignore the "higher level" partial record data. The drawback to this approach is that when a record is created at the "lowest level," the "higher level" partial record will contain old data and is thus an undesirable approach.

Hardware Impact

Hardware impact is minimal for this technique.

Software Impact

A custom DBMS must be created to provide this type of multi-level data separation.

Security

This technique provides good security without any covert channels. In addition, because a custom DBMS must be written, integrity control can be developed as part of the DBMS software.

Cost

Costs will be high for this approach due to the development of a custom DBMS system. System maintenance and systems personnel training for this non-standard system will add to the cost.

4.1.4 Black Bypass Approach Description

Another approach to handling secure multi-level domain relations is to provide a black key generator (KG) bypass of the Hardware Security Filter Subsystem with each data item physically tagged with its security clearance. This

approach requires an extensive upgrading of the Hardware Security Filter Subsystem and the development of a specialized DBMS that supports the KG bypass and physical security tags.

The Black Bypass approach allows a single level DBMS to modify a portion of a physical disk record containing multi-level data. The DBMS must first read a red (decrypted) version of a physical record with the hardware security filter blanking higher level data as determined from the data security tags. The DBMS then reads a black (encrypted) version of the physical record, overlays portions of this black version with red modifications, and writes the resulting mixed (red and black) version back to disk through the Hardware Security Filter Subsystem. The Hardware Security Filter checks the data tags of the red modified portion of the record against access control policy and merges these with the remainder of the record obtained by decrypting the black data.

DBMS Impact

The specialized secure DBMS requirements are similar to that for domain separation. The schema must be modified to require a security clearance for each data domain. The keys, associated computer access structures, parameters, and schema must be stored at the lowest level and common compartment of the domains. If variable length higher-level files are used, these partial records must be indirectly accessed through a flat pointer table, a separate file used to contain these variable length fields at each separate clearance, and an empty partial record queue maintained at these clearance levels. In addition, the DBMS must handle physical security tags, potentially on a fixed format basis. Also the DBMS must be capable of reading both red and black records and overlaying red changes on the black record in order to perform updates.

One advantage of this approach is that all levels of a multi-level relation can be read with a single disk access thus supporting relatively efficient queries of multi-level relations. However, there is an overhead associated with the physical data tags which impacts the system throughput and disk packing density.

Software Impact

The Black Bypass technique requires modifications to the DBMS to handle red/black data and the data tags.

Hardware Impact

The hardware impact is significant for this technique, particularly in the Hardware Security Filter (HSF).

One immediate impact upon the HSF results from the requirement of individual physical security tags. These security tags require a fixed storage format for data and clearance tag. When a physical record (disk sector) is subsequently read, the security tag data must be stripped out (deformatted), compared to access control policy, and higher level data is blanked.

The interface between the Hardware Security Filter Subsystem and the DBMS Processor Subsystem must also be redefined for this approach. Various options are available. The simplest option is to use the same fixed format used to store data and clearance tags on disk, thereby keeping transfer rates at both interfaces the same and allowing use of synchronous links.

Sending the data in both the encrypted (black) and decrypted (red) data to the DBMS processor impacts the HSF data handling requirements. Basically, there are three ways to handle black reads: re-read the physical record; double the link rate providing the filtered red version on one channel and the unfiltered black version simultaneously on a second channel; or provide a buffer in the Hardware Security, sending the filtered red version first and the black version later. The last two approaches are preferable as they require only one central data base read access whereas the first approach requires two central data base read accesses for an update.

The write channel in the Hardware Security System is the most complex of all. Additional formatting will be required to differentiate red and black fields in the mixed records. The black fields will have to be decrypted while the red fields have the security tags stripped out and checked against write access control. After processing, these fields must be re-merged into a red physical record in the central data base.

Another hardware impact is the requirement for a high speed KG and bypass mechanism. The KG should be dynamic rather than static; otherwise, changes to higher level data will be evident to lower level DBMS processors. Dynamic encryption synchronization will require either adding crypto sync data to the front of each black record or maintaining the same sequence for write operations as used for read operations.

Security

This approach has many security problems. The two worst are that it provides a high speed covert channel between DBMS processors and that it permits a successful clear text attack of the crypto system in a low level DBMS processor, thus compromising higher level data. The high speed covert channel requires the lower level DBMS machine to write a file of low level data, the high level DBMS machine then modifies the file using security clearance tags to encode high level data. The lower level machine can then read these modulated tags by determining which fields remain viewable. The other security risk is in giving a low level machine encrypted high level data and simultaneously providing clear text and encrypted versions of low level data. This processor gives a subject the means to totally break the cryptographic system. Because of these security impacts, this technique cannot be recommended.

Cost

This is an extremely costly system. Major factors are:

- o High speed KG equipment
- o HSF enhancements to handle read and black data read/write/merge operations
- o Modification required to the DBMS to permit red/black data and tag handling
- o Associated costs of non-standard DBMS in maintenance and training
- o An increase in storage costs due to the data tag requirement. However, this cost is partially offset by the reduction in access time through a reduction in the number of physical disk read operations.

This is, therefore, not only a non-secure system, but is also quite costly.

4.1.5 Internal Bypass Approach Description

The third multi-level domain separation technique is the internal bypass technique. This is a data tag technique whereby an exit guard filters all data sent to a subject and merges the altered fields with an internal copy of the record. This approach allows a single level DBMS to modify permitted portions of a multi-level physical disk record by merging allowed changes with the previous version of the physical record. The primary advantage of this approach is that

all levels permitted by access control policy, can be read with a single disk access thus supporting relatively efficient queries of multi-level relations.

When a DBMS reads a filtered record, an unfiltered copy is stored in a buffer internal to the Hardware Security Filter Subsystem. If the DBMS performs an update, it marks the fields to be changed and sends the update to Hardware Security Filter Subsystem which merges the change with the stored version of the record. The Hardware Security Filter screens the tags of the update against access control policy. If the clearance of the DBMS is not the same as the lowest level and common compartment of the relation, then the HSF also screens the tags of the update against the previous tags. This tag screening insures that security tags can only be changed from the lowest level to prevent the use of the tag level as a covert channel from a higher level DBMS machine to a lower level DBMS machine. If a tag or access violation is not present, the HSF then merges the changed fields with the version of the record previously stored within the HSF.

DBMS Impact

The Internal Bypass technique requires special modifications to the DBMS software very similar to that of the Black Bypass technique of Section 4.1.4. The schema must be modified to require a security clearance for each data domain. The keys, associated computer access structures, parameters, and schema must be stored at the lowest level and common compartment of the domains. If variable length higher-level files are used, these partial records must be indirectly accessed through a flat pointer table, (a separate file used to contain these variable length fields at each separate clearance) and an empty partial record queue maintained at these clearance levels. In addition, the DBMS must handle physical security tags, potentially on a fixed format basis. Additionally, the DBMS must be capable of reading both red and black records and overlaying red changes on the black record.

One advantage of this approach is that all levels of a multi-level relation can be read with a single disk access thus supporting relatively efficient queries of multi-level relations. However, there is an overhead associated with the physical data tags which impacts the system throughput and disk packing density.

Software Impact

Like the Black Bypass technique, this approach requires modification to the DBMS to handle the physical data tags.

Hardware Impacts

The primary hardware impact of this approach is in the Hardware Security Filter Subsystem. One immediate impact of individual physical security tags is the requirement of a fixed format for data and clearance tag storage in the central data base. When a physical record (disk sector) is subsequently read, the record is stored in a buffer internal to the Hardware Security Filter Subsystem. Simultaneously, the security tag data must be stripped out (deformatted)and compared to access control policy. Then the accessible (filtered) data is sent to the DBMS processor. If the access is for a query, the internal buffer is subsequently overwritten by a later read access. If an update is made, the changes are sent back from the DBMS processor and merged with the stored record.

Security tags may only be changed from the lowest level of the relations, i.e., the Hardware Security Filter Subsystem must enforce two levels of access control, one at the file level (e.g., using address security tags) and one at the data item level using physical tags. The Hardware Security Filter/DBMS Processor Subsystems interface must also be redefined for this approach to support physical security tags, and changed fields.

Security

This approach provides good multi-level security and performance. No direct covert channels are known to exist in this architecture.

Cost

This is a moderately costly multi-level relation separation technique. The major cost factors are:

- o DBMS Software modifications to support physical data tag generation and handling.
- o HSF must be capable of processing the data tags as an exit guard, must be able to use the data tags to control merges of modified data, and must provide the internal bypass buffer and control logic.
- o There is an additional overhead in storage due to the presence of the physical data tags which must be stored with each item. However, this cost is somewhat offset by the reduction in the access time resulting from the fewer disk accesses required per query as compared to other techniques.
- o There are increased costs of software maintenance and staff training associated with the DBMS software modifications.

4.1.6 Multi-Relation View Approach Description

This last approach to handling secure multi-level domain relations is to segment the multi-level relations into several single-level relations, one relation being created for each protection level present. Access keys must be replicated in every level relation. The use of views (logical joining of multiple relations) effectively allows end users to manipulate these multi-level "super" relations as they would a normal relation with similar multi-level security restrictions.

DBMS Impact

The primary operational impact is that instead of laying out a single schema, a schema must be layed out for each clearance level and for overall (multi-level) views. Since the layout of a data base structure is done relatively infrequently, usually by a skilled data base administrator, this is a minor impact. A second DBMS impact is the requirement for the multiple accesses necessary to retrieve each single-level subrelation to be built into the super-relations view. However, if multi-level views are a minority in the data base, this impact will be minor. In any case, the cost of multiple disk accesses per view are easily outweighed by the other cost savings.

A second operational impact is a record must be added and deleted at all clearance levels. When this does not happen, the multi-level view will show a partial record. The problem of multi-level view additions and deletions could be handled in part by passing the view addition and deletion requests from the lowest level to higher level machines at the cost of a modified DBMS and upward (blind) communication links or by trusted subjects on a multi-level machine. Another method for elimination of the partial records resulting from incomplete additions or deletions is the development of a process to aid in the detection of such records so that the condition can be corrected.

Software Impact

The software impact for this technique is minimal as a standard commercial relational DBMS can be used.

Hardware Impact

The Hardware Impact for this technique is minimal. A standard Hardware Security Filter provides the required access control.

Security

This approach provides good security with no obvious covert channels.

Cost

Because of the minimal software and hardware impacts, this architecture has a minimal cost inpact.

4.1.7 Evaluation of Multi-Level Relational Approaches

This section summarizes the merits of the four multi-level relation separation approaches.

Domain-Separation

The domain-file separation approach suffers from the same performance problems as the multi-relation view technique and has the same type of multi-level addition and deletion problem. In addition, it cannot use a standard DBMS, i.e., it requires the development of a specialized DBMS, with its associated higher cost. However, this approach will have some minor multi-level enhancements relative to the multi-relation view approach as a result of the requirement for a specialized DBMS. In particular, multi-level security will be directly incorporated in the schema and file level integrity security can be built in. These two features are also easily implementable for the internal bypass approach. This approach offers little improvement over the multi-relational view approach at the cost of a standard DBMS and is thus a poor compromise choice.

Black Bypass

The black bypass is unacceptable from a security viewpoint due to the existence of a covert channel and the clear text attack threat. In addition, the Internal Bypass approach is functionally equivalent with better performance, security and cost. Finally,the cost of the black bypass approach greatly outdistances all of the other techniques.

Internal Bypass

The internal bypass approach is functionally equivalent to the KG bypass approach, is less costly, and does not have the covert channel and clear text crypto compromise security threats inherent with the KG bypass. The internal bypass approach has superior performance and functionality potential. It is also costly as it requires an extensive upgrade of the Hardware Security Filter Subsystem and the development of a specialized DBMS.

Multi-Relation View

The view of multiple single clearance relations has no development costs and allows the use of a standard DBMS. It has the poorest multi-level performance potential but better single level performance, thus, if multi-level views are a small portion of the DBMS load and these views typically are restricted to a few clearances, the performance differential will not be significant. The largest drawback to this approach is the awkwardness of multi-level view additions and deletion which is not found in the internal bypass mode.

Conclusions

Figure 4.1-6 summarizes the characteristics of these multi-level data separation approaches. The indications are that the internal bypass is the technically superior technique but has a very high development cost. As a compromise, the best choice is the multi-level view approach which meets the requirements using commercial software and thus has the minimal cost.

APPROACH	DBMS IMPACT	SOFTWATE IMPACT	HARDWARE IMPACT	SECURITY	PERFORMANCE	COST
DOMAIN SEPARATION	MODERATE	MODERATE	LOW	HIGH	LOW TO MODERATE	HIGH
BLACK BYPASS	HIGH	HIGH	HIGH	POOR	LOW TO MODERATE	HIGH
INTERNAL BYPASS	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH
MULTI-RELATION VIEW	LOW	VERY LOW	LOW	HIGH	LOW	LOW

Figure 4.1-6 Summary of Multi-Level Data Separation Techniques

4.2 Hardware Security Filter Subsystem

The Hardware Security Filter (HSF) is responsible for enforcing a non-discretionary security access control policy provided by the security officer for each attached processor. Many implementations of an HSF are possible using existing technology. The primary differences among such implementations are the capability to support multi-level files and choice of data separation technique(s).

4.2.1 Overview

The primary HSF requirements driver is the choice of the type file to be protected: single-level or multi-level. The impact of this decision is discussed in detail in Sections 4.2.2 and 4.2.3 respectively. Security Failure Analyses (SFA) and Tempest are other primary HSF requirements drivers. Ideally, the HSF design will provide provable non-discretionary security enforcement independent of other subsystems for single level processors. Another implementation consideration is its adaptability to various central data base and DBMS processor subsystems. One aspect of this adaptability is the degree of transparency of the HSF to the central data base and the DBMS processor. For the DBMS subsystem, measures of the degree of transparency are the modifications required to interface to an OS the amount of support software required, and the HSF impacts to application programs. Ideally, the OS interface would appear as a standard disk driver with no support software required and no application program impact.

There are four primary, non-exclusive, separation techniques that can be utilized to implement an HSF: multi-key static encryption; physical security tags; address security tags; and directory tags. These techniques offer a large range of capability and features that are summarized in Figure 4.2-1. The static encryption approach uses a different key for each security classification to maintain data separation. The encryption and decryption keys available to a given processor are controlled by an access control policy (ACP) supplied by the security officer. The physical tag approach requires physical security tags to be attached to data and physically screens these security tags against ACP. The address tag approach segments disk storage into single level compartments. The disk address of each access request is converted into an associated security classification and is screened against the access control policy. The directory approach uses certified

	SECURITY RANKING	MULTI-LEVEL FILE CAPABILITY	PERFORMANCE PENALTY	DYNAMIC STORAGE ALLOCATION	COMPLEXITY	HARDWARE SIZE	COST
CRYPTO KEY *	1	YES	NO**	YES	4	4	4
PHYSICAL TAG	2.5	YES	YES	YES	3	3	3
ADDRESS TAG	2.5	NO	NO	NO	2	2	2
DIRECTORY TAG	4	NO	NO	YES	1	1	1

Ranking - 1-. BEST; 4- WORST

* Has SFA Benefits
**Indirectly Yes for Multi-level Files - Security Key Format Must Be Provided

- o Directory Tag Required for Audit Trail and Central Discretionary Security and Integrity Firmware
- o Other Techniques Limited to HW Enforcement of Non-Discretionary Security at Processor Level
- o Multi-Level File Capability Also Requires Secondary File Separation to Eliminate Covert Channel
- o Hybrid Approaches Possible and Beneficial

Figure 4.2-1 Hardware Security Data Separation Technique Tradeoffs

or trusted software to maintain the security level of each file in the directory and to control which accesses are permitted by the current ACP.

4.2.2 Hardware Security Filter With Single-Level File Support Sizing

Hardware Security Filters with single-level file support can easily be built. All four data separation techniques can be used individually or in combinations. The multi-key encryption data separation approach is the most secure but the most costly due to the need for KG equipment. The other security tag approaches offer a range of security and size.

Multi-Key Encryption Approach

The multi-key encryption approach shown in Figure 4.2-2 is the most secure HSF approach but requires more hardware than a security tag approach. The design shown here requires 6 line replaceable unit types (LRU), 9 to 15 cards, plus an additional KG sync processor and a single multi-key KG unit (see Section 4.4 for sizing of a potential KG system). This approach has no direct throughput or packing density overhead, but will increase central base latency to some degree due to the need for random access static crypto synchronization. This approach also requires utility software to support re-keying and will also require the ACS to keep track of old key/new key directory data if re-keying is to be performed on-line. The principal advantage of this approach is that central data base failures will not compromise security, i.e., if the wrong data is read it will not be delivered to a user in a decrypted form. The storage of statically encrypted rather than clear text (red) data is an additional security advantage.

The HSF subsystem has four interfaces: a standard disk interface to the central data base; a high speed serial interface to the DBMS subsystem; a man-machine interface (MMI) to the security officer; and a central data base request interface with the Access Control Subsystem (ACS). The encryption approach has the most complex MMI and ACS interfaces. In addition to access control policy entry, the security officer must also enter KG keys and potentially user code offsets. The HSF supplies a busy/idle status to the ACS. The HSF receives disk commands and the following access request data: read/write mode, security level and compartments, and the channel (DBMS) number, and possibly user ID and old/new key selection data.

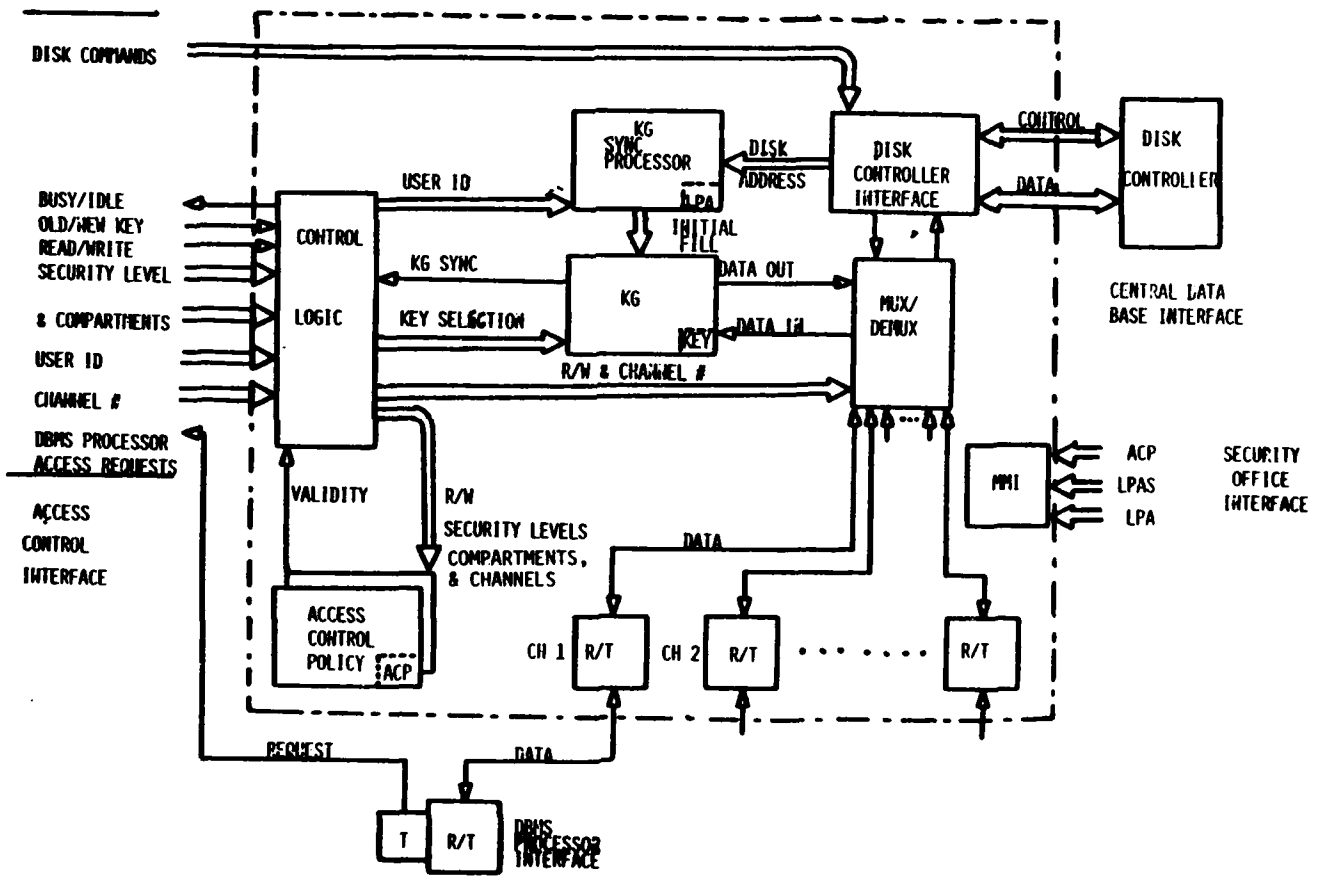


Figure 4.2-2 Hardware Security Filter Subsystem Employing Encryption

Data Tag Approaches

The following HSF subsystem designs rely on data having security tags to perform its security filtering function. The actual implementation can use any of three tag types: a physical tag, the central data base address of the data, or a directory entry. Depending on the tagging method used, various designs result requiring approximately 2 to 6 LRU types and 5 to 15 cards.

The physical tag approach shown in Figure 4.2-3 is the most secure tagging approach. It actually provides security classification granularity at the data item level. It also requires the most hardware of any tagging approaches and requires that the DBMS classify (physically tag) each data time or the HSF to internally add and remove security tags . Besides the possibly high DBMS subsystem overhead this approach also degrades effective disk packing density and throughput.

The central data base address tagging approach (Figure 4.2-4) essentially partitions the central data base storage by security classification. Security granularity is at the file level for this approach and screening consists of mapping the disk address to a security classifcation and verifying that the access control policy allows the requested read or write for that security classification. This approach has no direct throughput or packing density overhead, but the fixed disk partitioning results in a need for additional spare capacity due to the fixed vs dynamic allocation. This approach requires a moderate amount of hardware.

The directory entry approach (Figure 4.2-5) requires the least hardware but it is also the least secure. In this case, this subsystem is merely a secure switching network and the access control subsystem and its software must be totally entrusted with enforcement of access control policy. For all the tagging/non-discretionary approaches SFA concern extends to the central data base design and for this approach to the access control subsystem design as well.

Figure 4.2-6 summarizes the LRU's for the security tag approach.

The ACP Boolean Logic LRU is of particular interest (Figures 4.2-7 and 4.2 -8). This LRU contains the logic which maps the access control policy and the data request parameters into a logical enable/violation signal. The design calls for a pair of these LRU's to be operated in a self-checking mode.

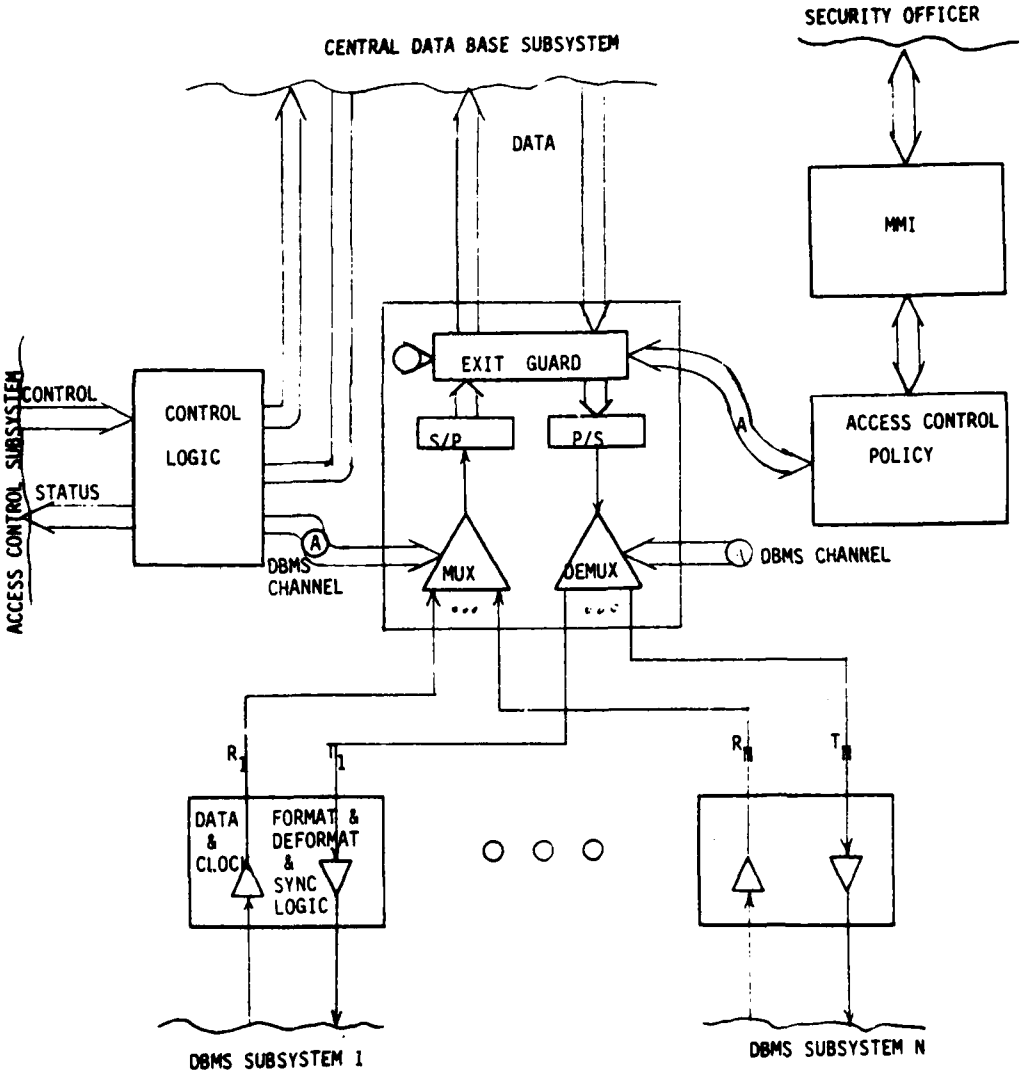


Figure 4.2-3 Hardware Security Filter Subsystem Employing Physical Data Tags

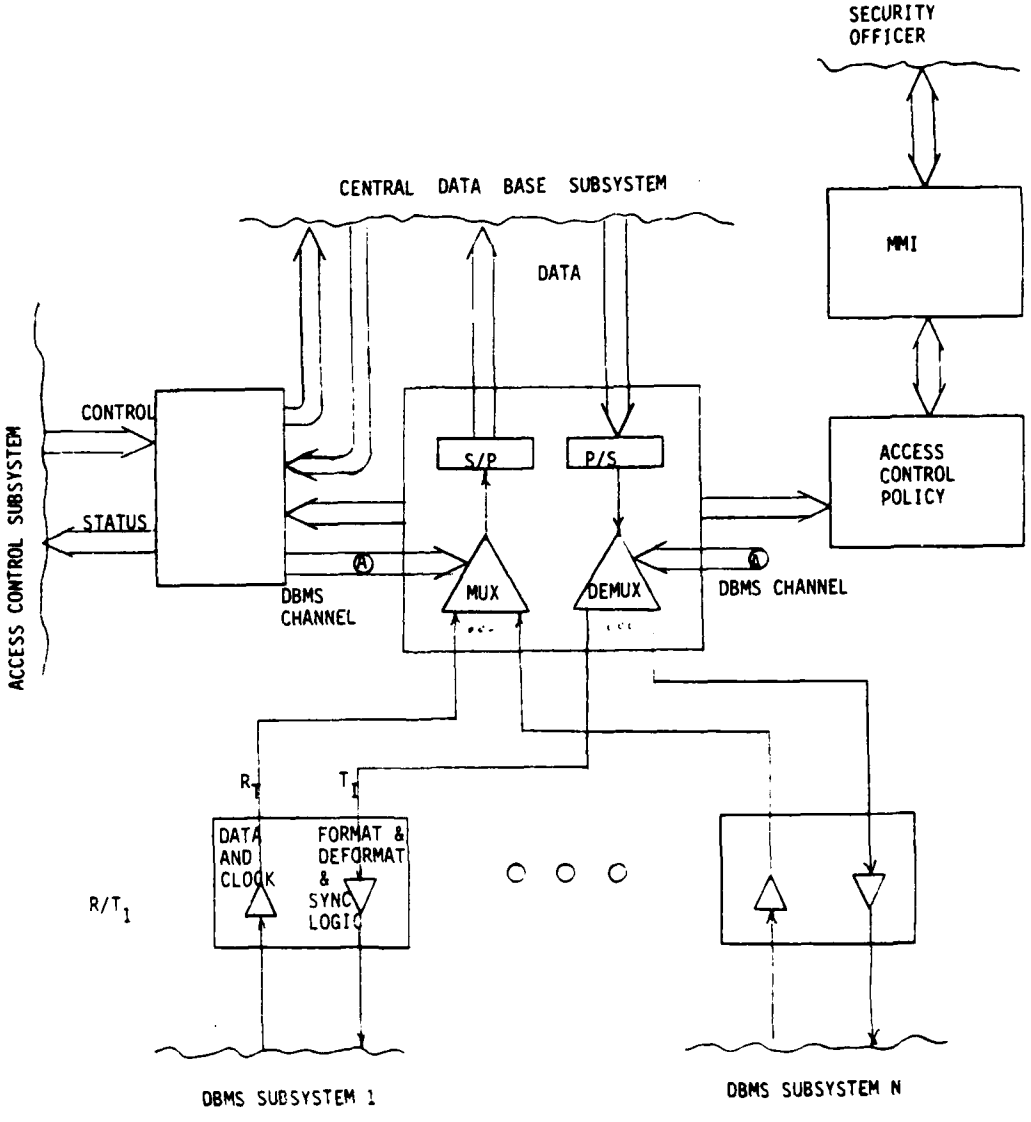
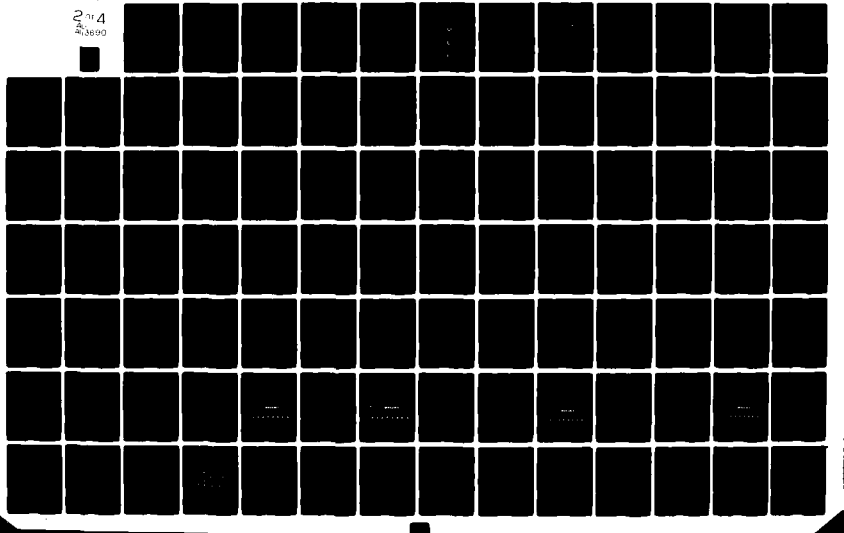


Figure 4.2-4 Hardware Security Filter Subsystem Employing Address Tags

AD-A113 690 HARRIS CORP MELBOURNE FL GOVERNMENT ELECTRONIC SYSTE--ETC F/G 9/2
SECURE DBMS.(U)
FEB 82 T D WORMINGTON, C E GIESLER F30602-80-C-0235
UNCLASSIFIED RADC-TR-81-394 NL



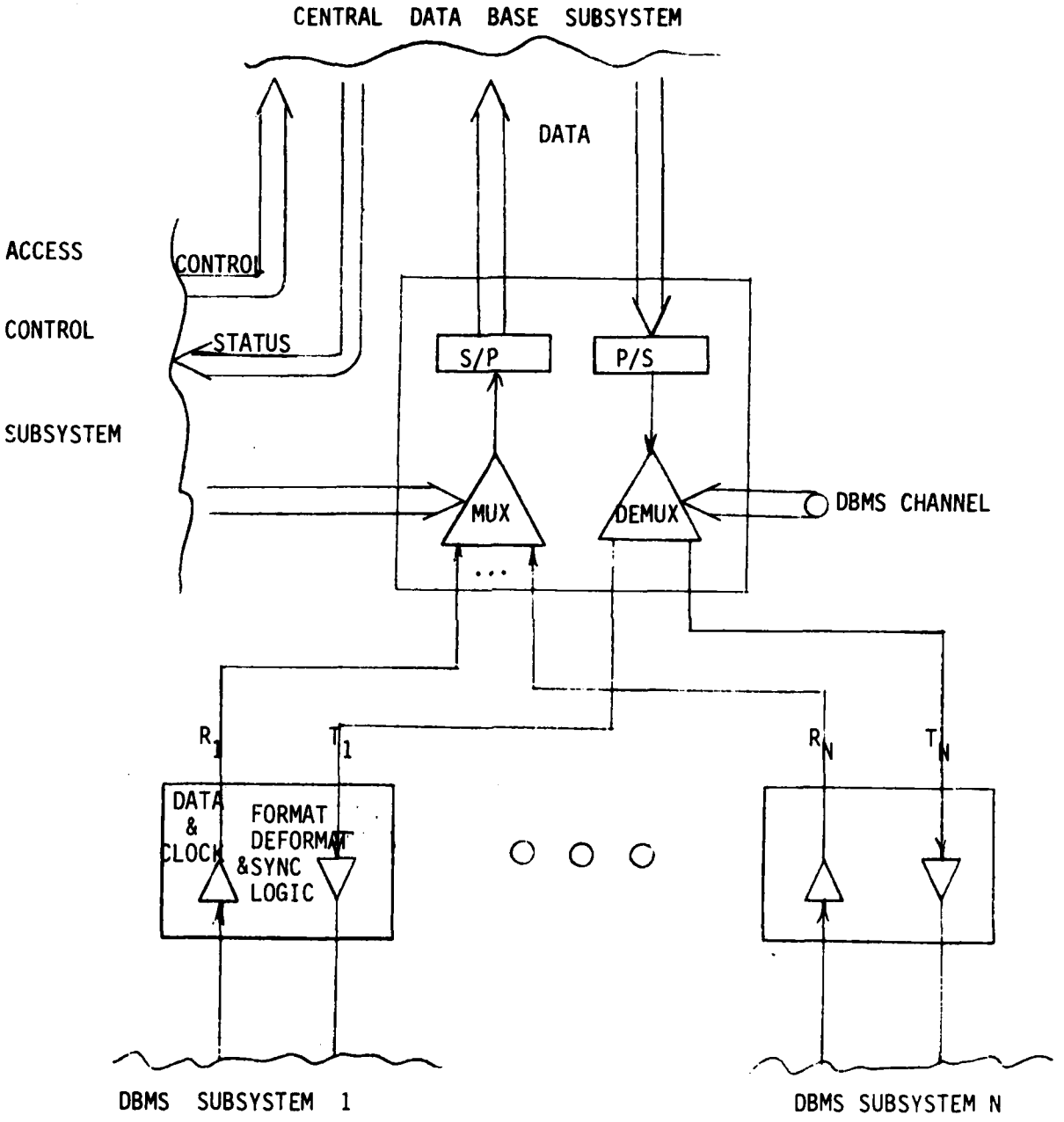


Figure 4.2-5 Hardware Security Filter Subsystem Employing Directory Tags

DBMS Interface

- o 10 Mbd receiver/transmitter
- o sync
- o timing
- o formatting logic

Fault Tolerant Switching Network

- o switches DBMS transmitter and receiver data and clock
- o for physical tag approach, provides entrance and exit guard
- o parallel data interface with the Central Data Base Subsystem

Control Logic Synchronizer

- o provides the ACS interface
- o controls interfaces with the other LRU's

Access Control Policy Boolean Logic

- o operates in dual-mode self-checking mode for fail safe
- o for address tag protection approach contains the address to security tag translation ROM
- o provides Hardware validation of requested access privileges by comparison to security policy established by security officer.

Disk Controller LRU

- o Serial-to-parallel and parallel-to-serial conversion
- o for encryption protection approach, inhibits read and write operations until crypto graphic sync is achieved (can perform seek function during inhibit)

Man-Machine Interface

- o permits selection of ACP by security officer
- o permits security officer to display access control data for any desired DBMS channel
- o permits validation of the selected policy

Figure 4.2-6 Description of LRU Functions for Security Tag HSF Approaches

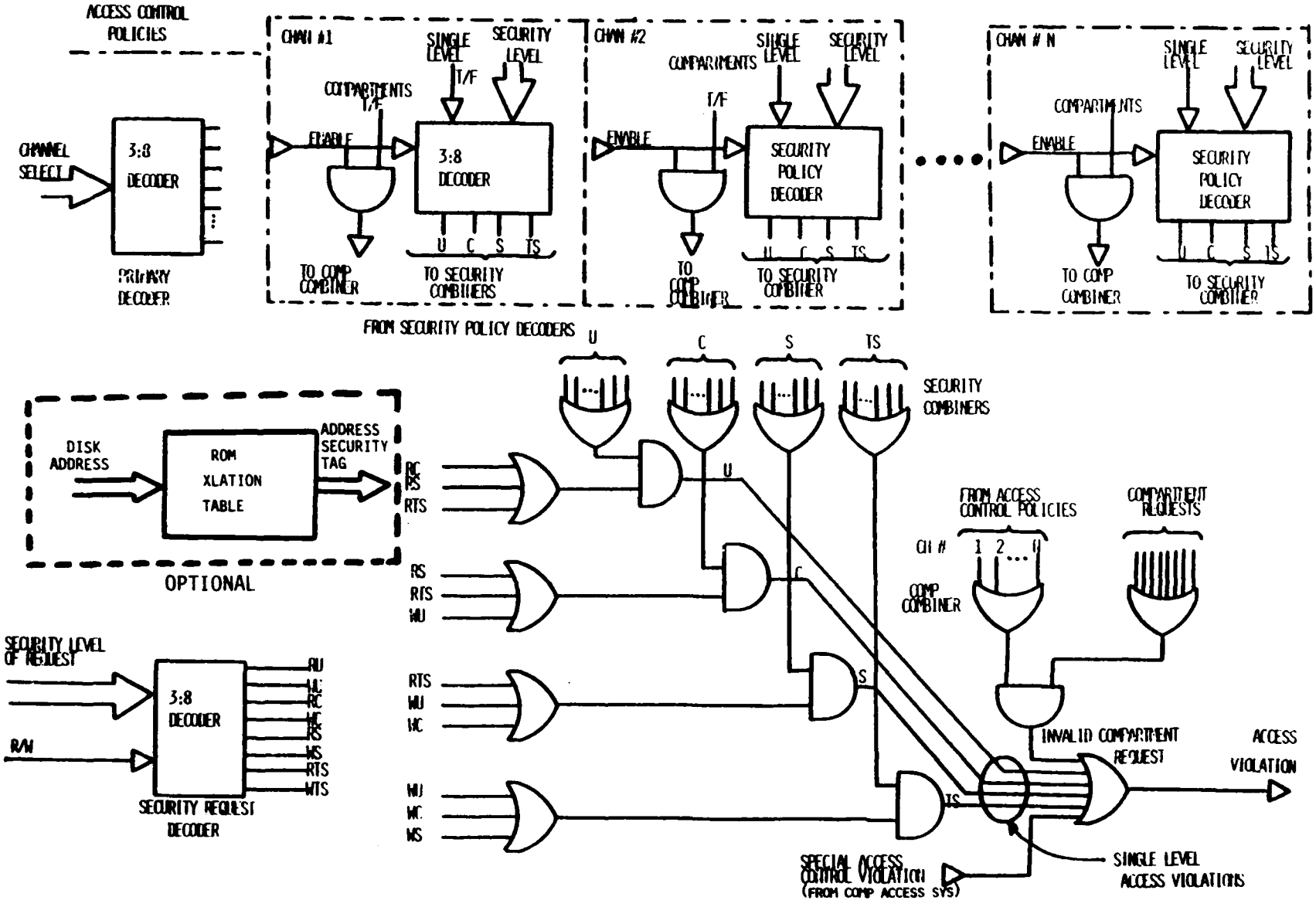


Figure 4.2-7 Access Control Policy Boolean Logic LRU

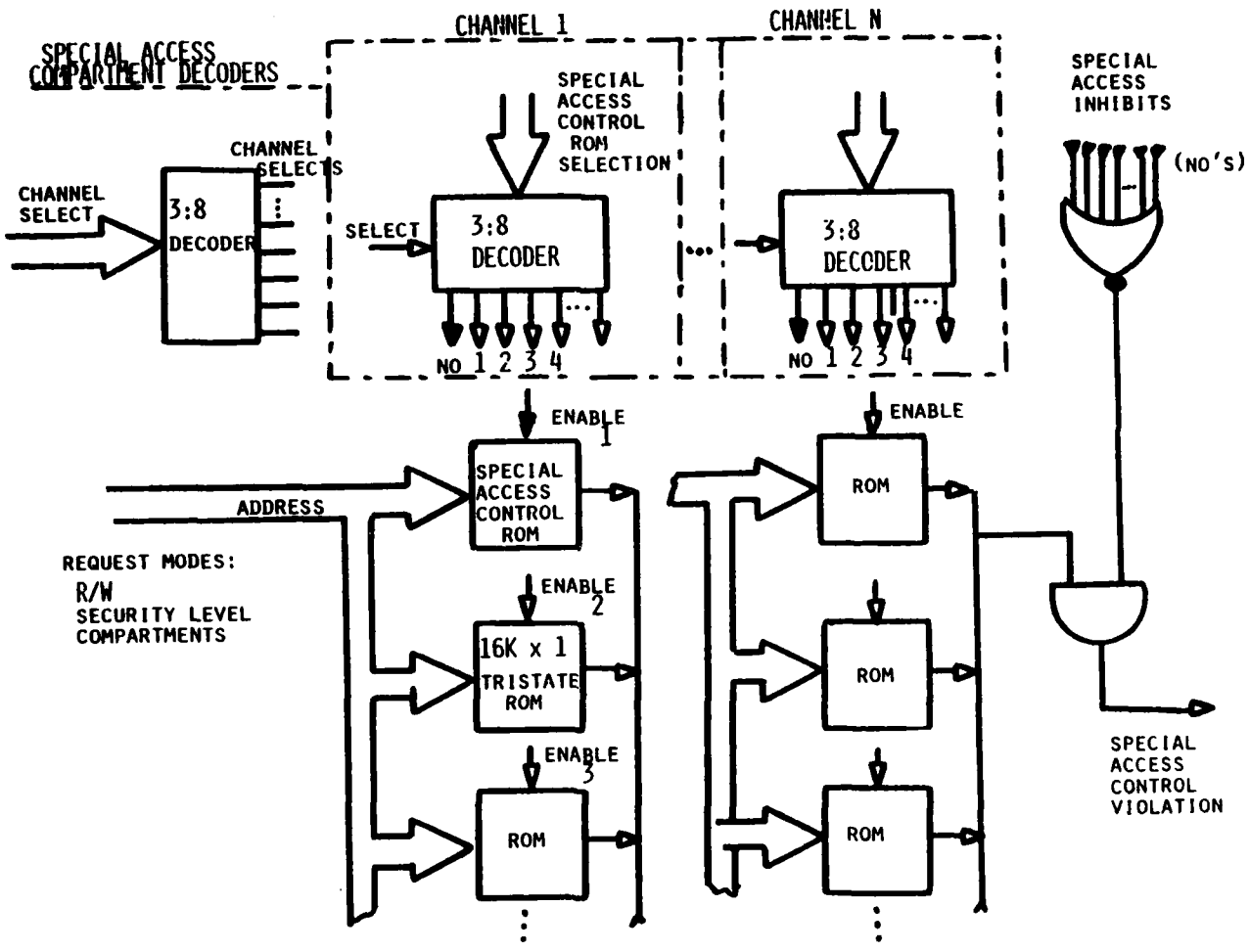


Figure 4.2-8 Hardware Compartment Access Violation Detector Logic

The special access (compartments) ROM's are the primary contributers to size, as there are N times M 16K by 1 ROM used - where N is the number of DBMS channels and M is the number of possible access control policies available. This LRU is not required if the directory approach is used, but may still be desirable.

The man-machine interface LRU is also worthy of comment. The control panel shown in Figure 4.2-9 is used to select the ACP for each DBMS processor. For the selected channel, access control data is displayed. Additional controls allow the security officer to modify the access control policy. This design allows the security officer to select hardwired enforcement of a single level security access policy (a simple security rule and *-security rule), its level, whether any compartment accesses will be allowed, and an optional special access control policy contained in ROM. In addition, this design allows validation of access control policy: In an off-line mode all possible access requests are made and the output sampled. At the end of this test, a display indicates which security levels may be read, which security levels may be written, which compartment(s) may be accessed, and a CRC for the sampled output sequence. Tempest and SFA requirements will impact this subsystem design and size. Further, the small number of LRU types will help reduce sparing cost.

4.2.3 Hardware Security Filter With Multi-level File Support Sizing

Providing a Hardware Security Filter with multi-level file support is much more complex than providing single-level file support. Either multi-key static encryption, physical security tags, or both are required for primary data separation. Data classifications must be checked against access control policy on the fly with higher level data blanked during reads. In addition, a secondary file level data separation technique is required to insure that the internal file security structure can only be created from the lowest level of the file (see Section 4.1). In addition, the HSF must provide a read-modify-write capability. (see Internal Bypass in Section 4.1). Besides the additional hardware needed to upgrade an HSF for this multi-level file support, the complexity of control logic also increases enormously thus making certification much more difficult. At a minimum the control logic required is a rather large, complex state machine and could easily result in the requirement for some certified low level firmware.

ACCESS CONTROL PANEL

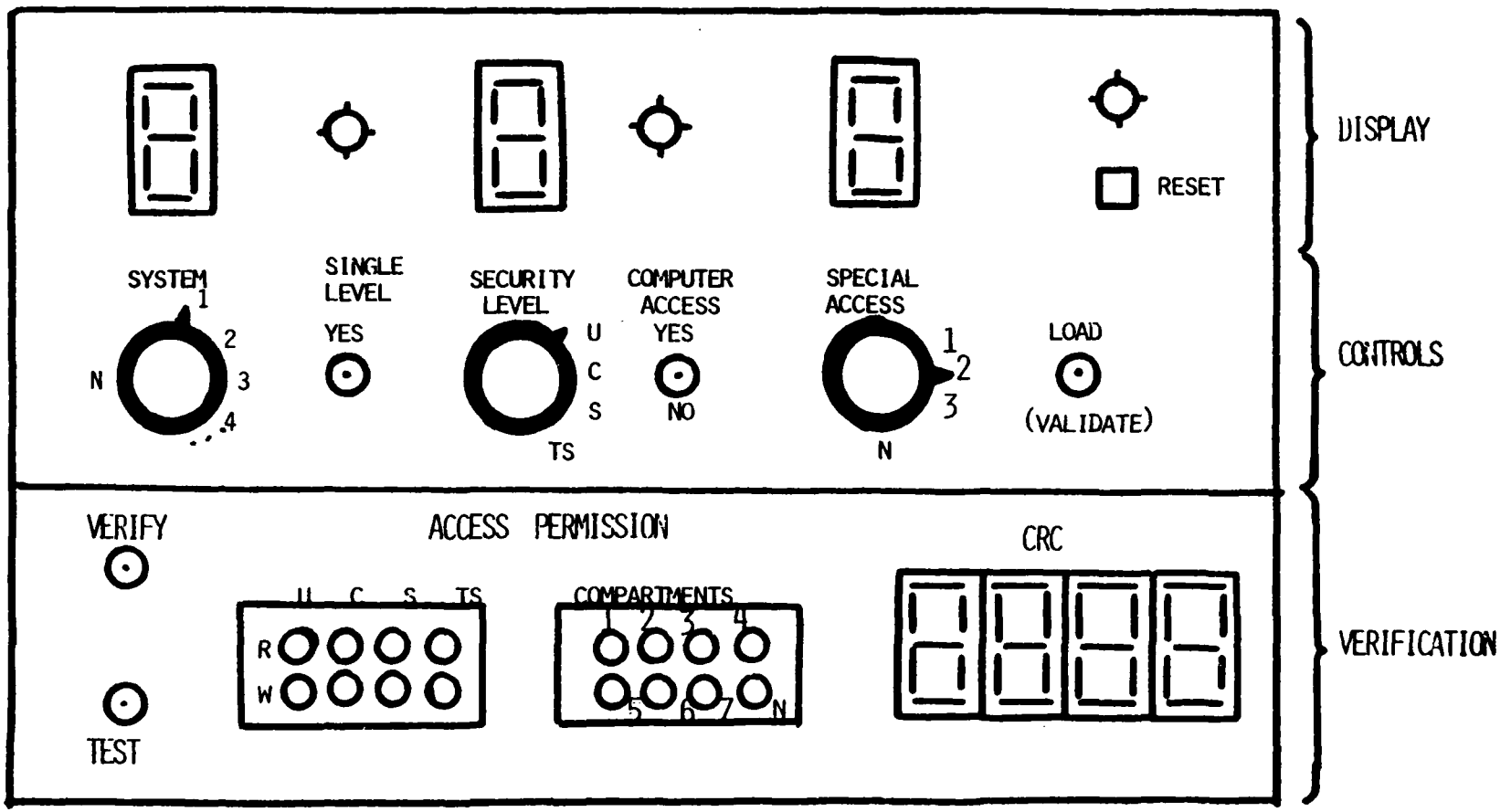


Figure 4.2-9 Access Control Policy MMI

A maximum HSF system design (Figure 4.2-10) uses physical address tags for internal multi-level file security structure, address tags for secondary minimum file security level protection, and static encryption for SFA purposes, and is fault tolerant (dual channel). With the partitioning shown, about 8 LRU's and 18 to 32 cards are required excluding the KG systems. A single MMI, a triple mode redundant majority ACP, and the DBMS receiver/transmitters (R/Ts) are shared by the two central data base channels. The rest of the LRU's for these two channels are independent. The Internal Bypass Buffer (IBB) and Multi-level Security Tag Screener (MSTS) are new LRU types, not found in the single-level file HSF previously described in Section 4.2.2. The control logic LRU is considerably more complex and the switch network and R/T LRUs have more connectivity and routing functions.

The Multi-level Security Tag Screener performs many security related functions. For reads, it deformats the disk output into security tags and data, checks the security tag against access control policy and blanks any data not permitted by ACP. Writes are more complex and require screening of user data prior to the merging of the old data previously stored in the internal bypass buffers with the desired changes. A possible format for the user data is shown in Figure 4.2-11. Note that two extra bits that are not stored are required to indicate whether to use the old security tag and data, merge the old security tag with user supplied data, or replace the old security level of the tag and data with user supplied items. In the event that the security tag is to be altered, the HSF must verify that the request is coming from a DBMS processor at the lowest level of the relation. This is to prevent a covert channel through data tag modulation, similar to the case discussed in Sections 2.4 and 4.1.1.

The Internal Buffer Bypass should hold at least a track of data from disk, typically 17K bytes, for relatively high efficiency. This would allow a read-modify-write operation to be performed on a per track basis and require multi-track operations to be decomposed into multiple single track operations. This technique obviously reduces disk throughput by requiring a read operation before write in addition to any required DBMS read accesses.

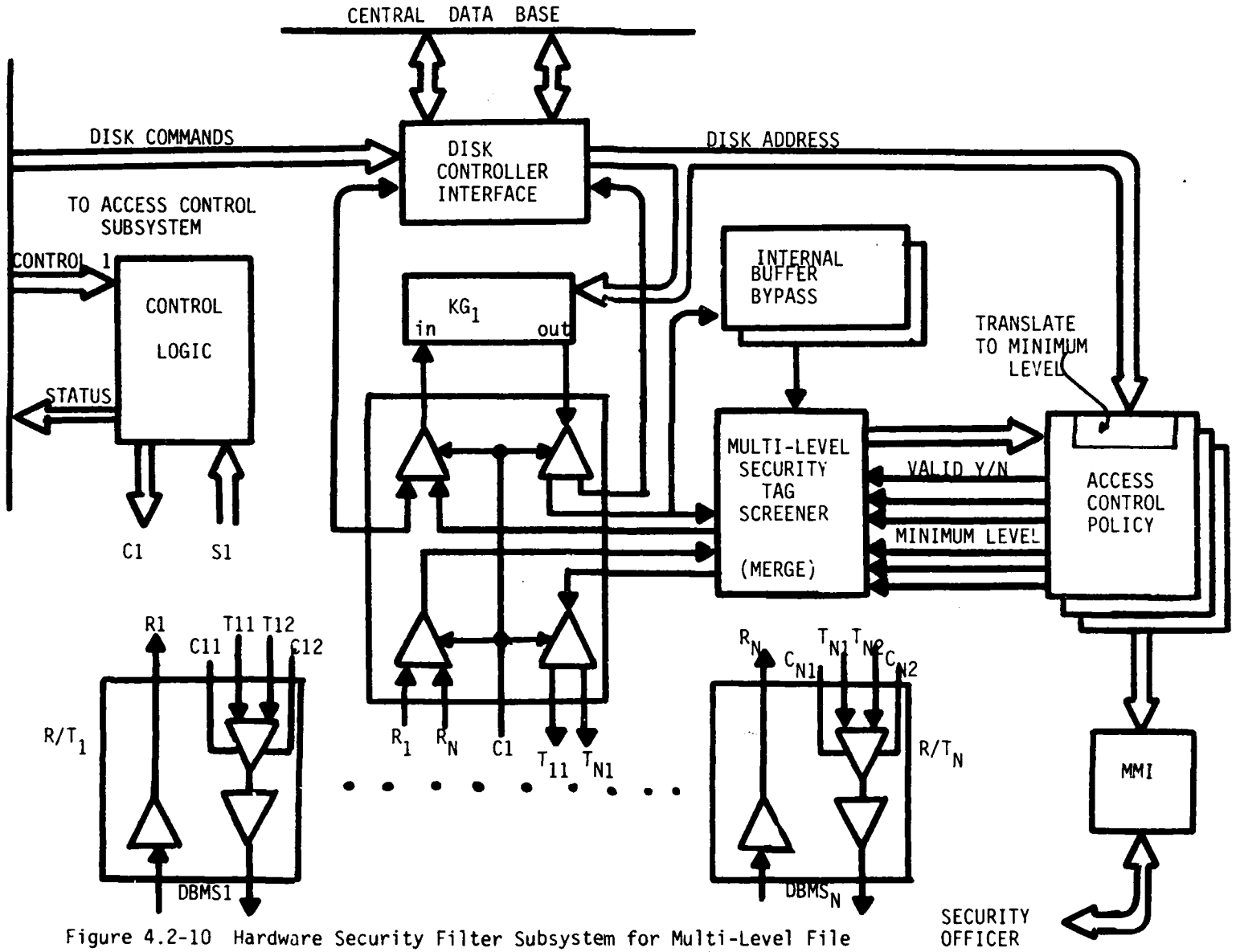
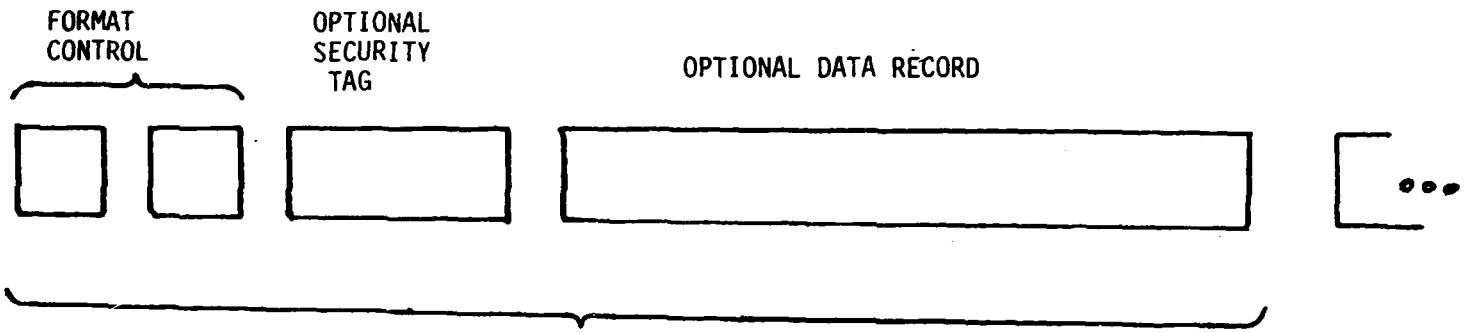


Figure 4.2-10 Hardware Security Filter Subsystem for Multi-Level File Support (Maximum System)



FORMAT CONTROL USE OLD TAG AND DATA
MERGE OLD TAG AND NEW DATA
USE NEW TAG AND DATA

Figure 4.2-11 Possible Write Format

4.3 Access Control Subsystem Implementation

The access control subsystem (ACS) processes the user queries and initiates the required activities to fill the user's request. The ACS performs actions in four major areas:

- o Directory Request Processing
- o Disk Access Requests
- o Security Monitoring
- o Audit Trail Generation

These were functionally described in Section 3.3. This section will describe the implementations of the ACS and their requirements.

4.3.1 Hardware and Software Sizing for the ACS

The software of the Access Control Subsystem (ACS) represents the majority of the trusted and certified software of the Secure DBMS. At least eight interrupt handlers, two real time foreground tasks, one background task, and a multiprocessing executive are required. In addition, boot and diagnostic software is also required for off-line modes. Figure 4.3-1 describes these handlers and a summary of their processing activities.

The DBMS message interrupt handler is tasked with queueing the request for the proper foreground handler and entering the request on the audit trail. The Disk Access Foreground Task processes data base access requests, verifying the credentials of the requesting subject and placing successful requests on the Central Data Base Controller queue. This Central Data Base Controller is responsible for transmitting the requested information to the correct subject. A Directory Request Foreground Task accepts messages requesting access to the DBMS Directory and verifies the requesting subjects privileges before processing the request. The Man Machine Interface (MMI) permits the security officer to communicate with the DBMS and establish access control policies for each processor channel. Finally, the Multi-processing Executive supplies the system handlers for such activities as disk control, real-time clock, etc.

Figure 4.3-2 presents a conceptual block diagram of the Access Control Subsystem.

DBMS Message Interrupt Handler

- o Receipt of DBMS Request by interrupt handler
- o Send message to Audit Trail Buffer
- o Dump audit trail buffer if full
- o Determine request type: central data base access or directory request
- o Place message task into queue for proper process

Disk Access Foreground Task

- o Accepts messages from the DBMS Message Interrupt handler
- o Verifies request against current access policy
- o Verifies request against user's access privileges
- o Verifies the discretionary access requirements
- o Places any detected violations in the audit trail
- o Alerts security officer of any detected violations through the man-machine interface task
- o Converts Valid requests into physical Central Data Base requests
- o Places physical disk access request into disk access queue

Directory Request Foreground Task

- o Accepts messages from DBMS Message Interrupt handler through the Directory Access Queue
- o Verifies request against current access policy
- o Verifies request against user's access privileges
- o Verifies the discretionary access requirements
- o Places any detected violations in the audit trail
- o Alerts security officer of any detected violations
- o Valid requests are processed by modifying the directory
- o Valid requests are placed on the audit trail

Figure 4.3-1 Access Control Subsystem Process Descriptions

Man-Machine Interface (MMI) Background Task

- o Permits security officer to modify the security data base
- o Permits security officer to make queries
- o Permits security officer to request reprots
- o Generates reports of attempted security violations and anomalies in near real-time

Central Data Base Controller

- o Accepts disk access requests from the Disk Access Foreground Task through the disk access queue
- o Processes such requests by retrieving the requested data and sending it to the proper user channel
- o Responsible for requesting the proper I/O channel (user) via MUX controls
- o For systems with static encryption, ensures that a channel with the proper key is available during rekey periods

Multi-Processing Executive

- o Real time multi-tasking disk base executive
- o Disk handler
- o Real-time clock handler
- o System boot firmware
- o High order language support library
- o Diagnostic software

Figure 4.3-1 (cont'd) Access Control Subsystem Process Descriptions

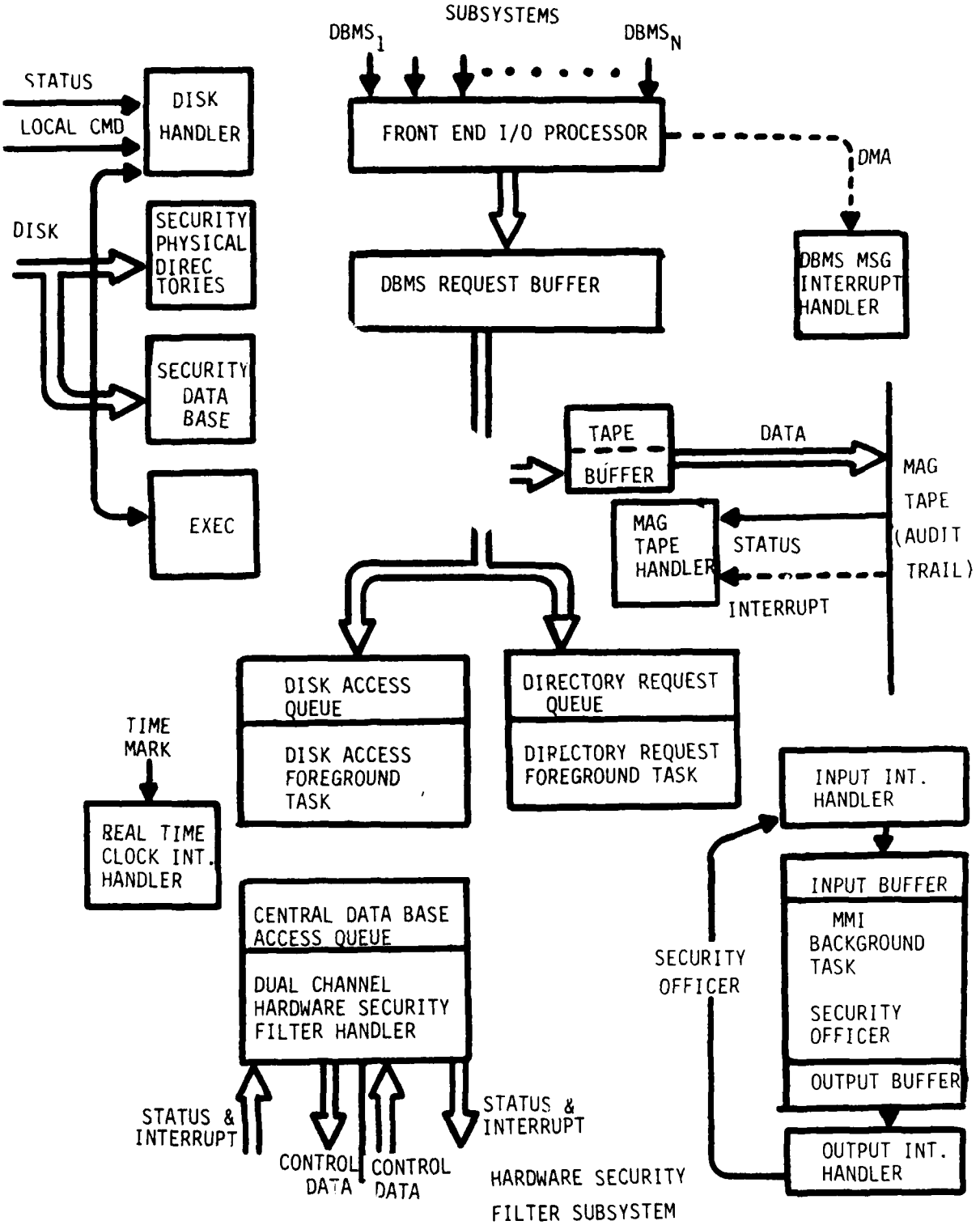


Figure 4.3-2 Conceptual Block Diagram of the Access Control Subsystem

Figure 4.3-3 lists the various buffer and data bases with size estimates. It should be noted that directory data bases will not fit in memory and hence will need to be accessed in parts from disk. This results in more complicated application and system software to perform this task and makes performance very sensitive to its structure, buffer size and accessing strategy. To achieve high performance, a hierarchical structure and/or a least recently used buffer strategy may be required.

It appears that a small minicomputer, approximately 128 Kbyte/250 Kips (thousands of instructions per second) system, is best suited to this application. Based on the 15ms access request latency requirement, about 4,000 machine instruction or 750 HOL instructions are thus available to implement a single request. Allocating 2ms to the message interrupt handler, 2ms to the central data base controller handler, and 1ms to the mag tape handler, a 10ms or 500 executed HOL instruction allocation for the Disk Access Foreground Task results.

In all, it is estimated that this subsystem will contain approximately 10K lines of certified code with a 50% assembly/50% HOL mix. Initial Application Task HOL applications are given in Figure 4.3-4.

4.3.2 Alternate Access Control Subsystem: Distributed Architecture

Validation of the Access Control Subsystem (ACS) as described in the previous section is virtually impossible due to the quantity of code required. One approach for those installations requiring certification is to construct the ACS in a dedicated architecture wherein the modules are restricted to a size which can be validated. Only those portions of the ACS deemed critical (i.e. those related to security) must actually be processed through a validation program.

By using a distributed network of dedicated microprocessors rather than a single concurrent multitasking minicomputer, it is possible to design a system where each processor runs a single task. Figure 4.3-5 shows such a distributed architecture. This particular design uses 6 microprocessors which interconnect through various data and status interfaces, dual ported disks, and shared memory. Despite this heavy hardware coupling, the software/firmware is only loosely coupled. The "real time" portion of this architecture uses only data and status interfaces to pass single messages (DBMS requests) with subsequent messages held externally

Security Data Base

256 users x 16 bytes/user	4K bytes
8 processors x $2^{(1+2+11)}$ bits/access policy map	16K bytes
	20K bytes

Director Data Base

Security (level, compartments, access control list)	
16,384 files x 128 bytes/file	20 Mbytes
Physical (size, disk address, map)	
16,384 files x 256 bytes/file	40 Mbytes
	60 Mbytes

Buffers

Msg buffer - 8 processor x 1msg/processor x 64 bytes/msg	512 bytes
Disk Access Queue - 32 x 32 bytes each	1024 bytes
Directory Request Queue - 32 x 64 bytes each	2048 bytes
Central Data Base Request - 32 x 8 bytes each	256 bytes
MMI buffers - 4 lines x 80 bytes each	320 bytes
	4160 bytes

Software

5K lines of assembly @ 2 bytes each	10K bytes
5K lines of HOL @ 12 bytes each	60K bytes
	70 K bytes

Total Memory Required

20K bytes + 32K bytes + 4K bytes + 70K bytes = 126K bytes

Total Disk Required

60 Mbytes x 1.33 = 80 Mbytes

Figure 4.3-3 Memory Allocation For the ACS

Disk Access Foreground Task	300 lines of HOL
Directory Request Foreground Task	1000 lines of HOL
MMI Background Task	3000 lines of HOL

Figure 4.3-4 Application Task HOL Allocation For the ACS

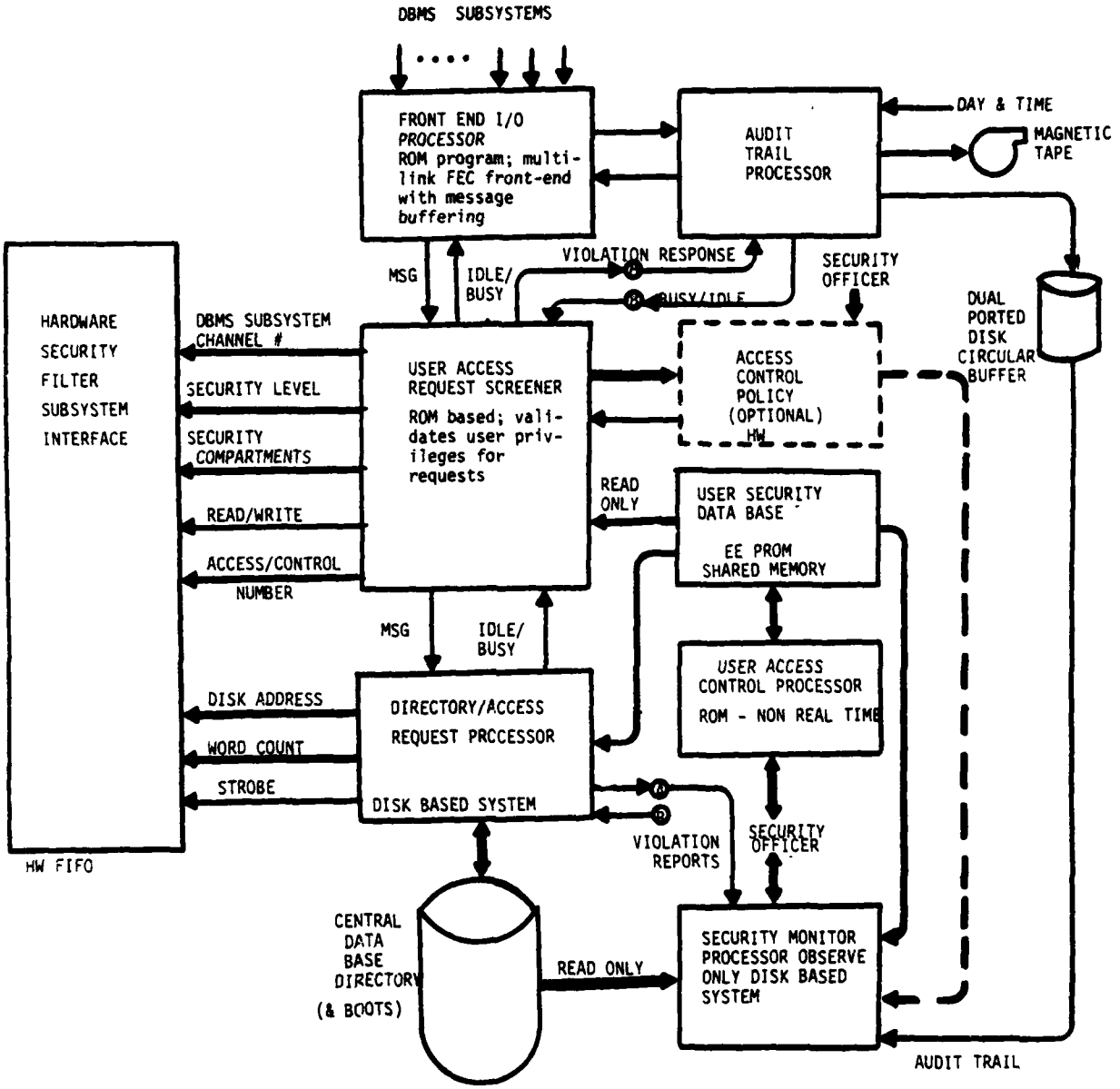


Figure 4.3-5 Distrubuted Access Control Subsystem Architecture

until the processor has finished processing the previous message. Idle/busy processor status is used for flow control, so as to preclude any concurrent processing requirement.

There are four real-time processors and two non-real time processors in this architecture. The Front-End I/O Processor accepts request messages from the DBMS processors on receive only links and queues the requests for the audit trail and User Access Request Screener processors. It sends the same message to both processors. The Audit Trail processor accepts messages from any of the ACS processors and places the messages on the audit trail along with a time tag. The User Access Request Screener processor accepts messages from the Front-End I/O processor and verifies the non-discretionary access requirements for the request against the User Security Data Base. The Directory/Access Processor maintains discretionary and integrity protection and converts valid requests to physical addresses. In addition, it generates the transfer signal enabling the HSF FIFO Buffer to be loaded from the ACS. The User Security Data Base shown is stored in EEROM. This subsystem contains the user access privileges of non-discretionary and integrity access privileges for ACS screenings. The user privileges are entered into this data base through the background User Access Control Processor.

The Front-End I/O Processor, the User Access Request Screener Processor, the User Access Control Processor, and the Audit Trail Processor all require certified code and require relatively small amounts of code which will be placed in ROM, i.e., these tasks will be performed with firmware. In conjunction with the Hardware Security Filter System which will enforce the security access selection, these secure tasks enforce user non-discretionary security access control and provide a secure audit trail.

The Directory/Access Request Processor contains the majority of the code and is only trusted to enforce discretionary security. Any trap door software in this task could only compromise discretionary security or deny system operation. It cannot compromise compartment or level access controls unless other trusted hardware or software is simultaneously compromised. Thus, it is desirable but not necessary to certify this trusted software. It is expected that this software will reside on disk.

Figure 4.3-6 summarizes the tasks of the individual elements of of the distributed ACS architecture.

Front End I/O Processor (real time)

- o Queues all user requests from the DBMS subsystems
- o When Audit Trail Processor is idle, sends message to its Audit Trail Processor
- o When User Access Request Screener is idle, sends message to it for processing
- o Waits for idle signals from both the Audit Processor and User Access Request Screener

Audit Trail Processor (real time)

- o Accepts all messages from other ACS processors and sets busy flag
- o Tags all reports with time
- o When Buffer is full, initiates Writes to Audit Trail
- o Logs message on circular disk buffer
- o Resets idle state upon completion of tasks

User Access Request Screener (real time)

- o Accepts messages from Front-end I/O processor and sets busy flag
- o Identifies requesting user
- o Screens request against the non-discretionary user access privileges obtained from the user security data base
- o Violations are passed to both the security monitor processor and the Audit Trail processor
- o Valid directory requests are sent to the Directory/Access Request processor
- o Valid Data base requests are processed by sending control data to the security filter subsystem FIFO interface
- o Sends message to Director/Access processor to obtain physical addresses
- o System must wait for completion of task from either the Directory/Access Request or Security Filter to complete processing. Upon completion, the User Access Request Screener resets the flag to idle.

Figure 4.3 -6 Distributed Access Control Subsystem Architecture Functions

Directory/Access Processor

- o Accepts messages from User Access Screener and sets flag to busy
- o Validates the request for discretionary security and integrity protection as read from the user security data base
- o Valid directory requests are processed
- o Provides physical data base addresses for Valid Access requests
- o Outputs addresses and word counts to the hardware security filter system and strobes User Access Request Screener data into hardware security filter FIFO interface

Security Monitor Subsystem (MMI) (Background)

- o Permits security officer to make queries
- o Permits security officer to request reports
- o Generates reports of attempted security violations and anomalies in near real-time

User Access Control Processor (Background)

- o Provides the means for the security officer to establish the user privileges for non-discretionary and integrity protection.

Figure 4.3-6(Cont'd) Distributed Access Control Subsystem Architecture Functions

The Security Monitor Processor also involves a great deal of trusted software. Since it can only observe the rest of the system, the largest security threat is altering data presented to the security officer, e.g., ignoring violation reports.

4.4 Encryption

Multi-key encryption is a primary technique for enforcing data separation both within the central data base by security classification and at the terminal level by user. Static encryption is required for crypto separation of data within the central data base due to the random access requests. Dynamic end-to-end (E^3) link encryption can be used to provide data separation by user at the terminal level.

4.4.1 Static Encryption

Static data base encryption is a method of enforcing the data separation by security level, security compartment, and user need-to-know that is required in a secure multi-level DBMS. The feasibility of such a system depends in part on the degree of this data separation which determines the number of encryption keys required. Analysis shows that from 20 to 8000 keys may be required depending upon the degree of user, security level, and compartment separation required. The following sections investigate various means of providing the desired degree of separation and other requirements for static encryption use.

4.4.1.1 Separation Through Encryption Key

This section discusses the key requirements of a statically encrypted data base where the only means of separation is through the selection of the key generator (KG) key. Conceptually, such a system appears in Figure 4.4-1 wherein a separate KG is used for each security level to be isolated. Such a system requires only four keys.

The KG key can be used to separate more than just security level. To enforce DoD non-discretionary security, compartment separation is also required. There are many security compartments, e.g., NATO, Crypto, CNWDI, NFN, various SI compartments, etc., not all of which are mutually exclusive for data base objects.

If there are from 5 to 20 combinations of these compartments, non-discretionary security requires from 20 to 80 KG keys.

In addition to the non-discretionary security requirement, security requires a need-to-know before granting access of objects to subjects. This concept is the basis of discretionary security. For a system supporting a moderate number of subjects, 10 to 100 users for example, the total key requirements now range from 200 to 8000 keys.

An alternate approach is to provide data separation solely by user however, this approach conflicts with the concept of shared data which is the major justification for secure multi-level DBMS.

Figure 4.4-2 shows the number of encryption keys required for various degrees of data separation.

4.4.1.2 Basic Candidate Architectures for Static Encryption

To support random access of encrypted data, some means is required to establish the phase, or sync, of the encryption code sequence to be used. One approach to static encryption is to use a repeatable pseudo noise (PN) source as input to a non-linear network (i.e., National Security Agency (NSA), Key Generator (KG) with the output mixed with data). Figure 4.4-3 shows a block diagram of this approach capable of providing both encryption and decryption. The PN source shown is derived from a linear feedback shift register (LFSR) with the resulting code sequence displaced by a linear programmable adder (LPA). With every random access this linear PN source is loaded with a known repeatable code state. The LPA and non-linear network (NLN) both contain shift registers which must be loaded with the new PN sequence prior to KG output code synchronization. This is done by clocking n + m bits through the system prior to the disk transfer, where n and m are the lengths of the respective shift registers for the LPA and the NLN. This period constitutes a crypto transient during which the KG is not synchronized and hence not available.

This architecture allows direct control of three parameters: initial LFSR code state, LPA code offset, and choice of KG key. These three

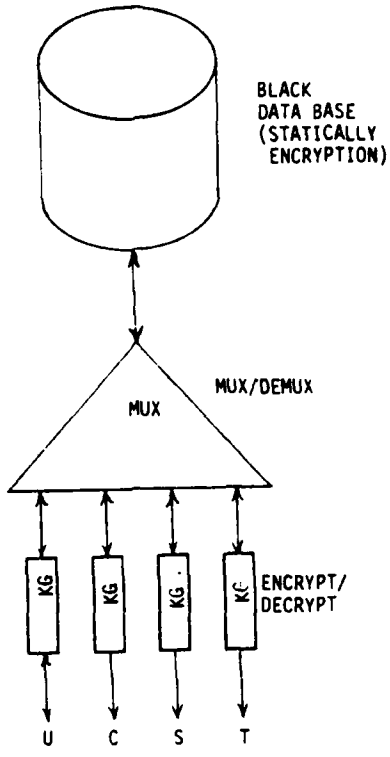


Figure 4.4-1 Static Data Base Encryption To Enforce Data Separation By Security Level

Degree of Data Separation	# of Crypto Keys Required
(U,C,S,T)	4
(U,C,S,T) x (compartments)	20 - 80
(user)	10 - 100
(U,C,S,T,)x(users)	40 - 400
(U,C,S,T) x (compartments) x (users)	200 - 8,000

*These numbers ignore potential independence of some security compartments and the lack of compartment relevance at the Unclassified level.

Figure 4.4-2 Degree of Data Separation vs. Number of Crypto Keys Required

degrees of freedom are exploited in subsequent sections to reduce the KG key requirements, thus increasing the feasibility of using static encryption.

4.4.1.3 Approach to Random Access Using Static Encryption with KG Latency Limited by the KG Transient Period

The previous section described the latency time required to synchronize the KG by initializing the PN code source with a known repeatable initial fill. For random access static encryption, it is possible to limit this latency time by directly mapping the disk address sector (sector or track) into a crypto state by "instantly" loading the LSFR. This latter approach would require a 78 entry table of sector codes for a 300 Mbyte drive and is easily implementable as a ROM translation table. (Figure 4.4-4).

An additional feature of this design is the capability of using the track address to provide a code phase offset for each track via selection of LPA taps. Without this feature the LFSR would provide the same PN sequence for each track with corresponding weakening of crypto protection. In particular, if a hardware fault results in improper track selection, the data would be correctly decrypted without the use of track encoding. Hence, while the LFSR only repeatedly produces a short portion of the PN code sequence, the addition of the LPA induced PN code phase offsets allows the KG input (and hence output) to be unique for each data location. If multiple disk drives were used, disk addressing can be used as an extension of the track addressing to insure unique encryption sequences for each drive.

4.4.1.4 Using Code Sequence Assignments to Achieve User Separation

The previously described approaches to random access static encryption have discussed means of separating users and security levels by crypto key and the methodology of establishing crypto sync. However, all the previous approaches still require an unacceptably high number of keys, from 20 to 8000. One approach is to move the user separation from the key and provide user separation through code phase offsets using the LPA's. This procedure provides very strong KG key separation of data by (U,C,S,T) security levels and various security compartments with weaker separation by users within the same security level and

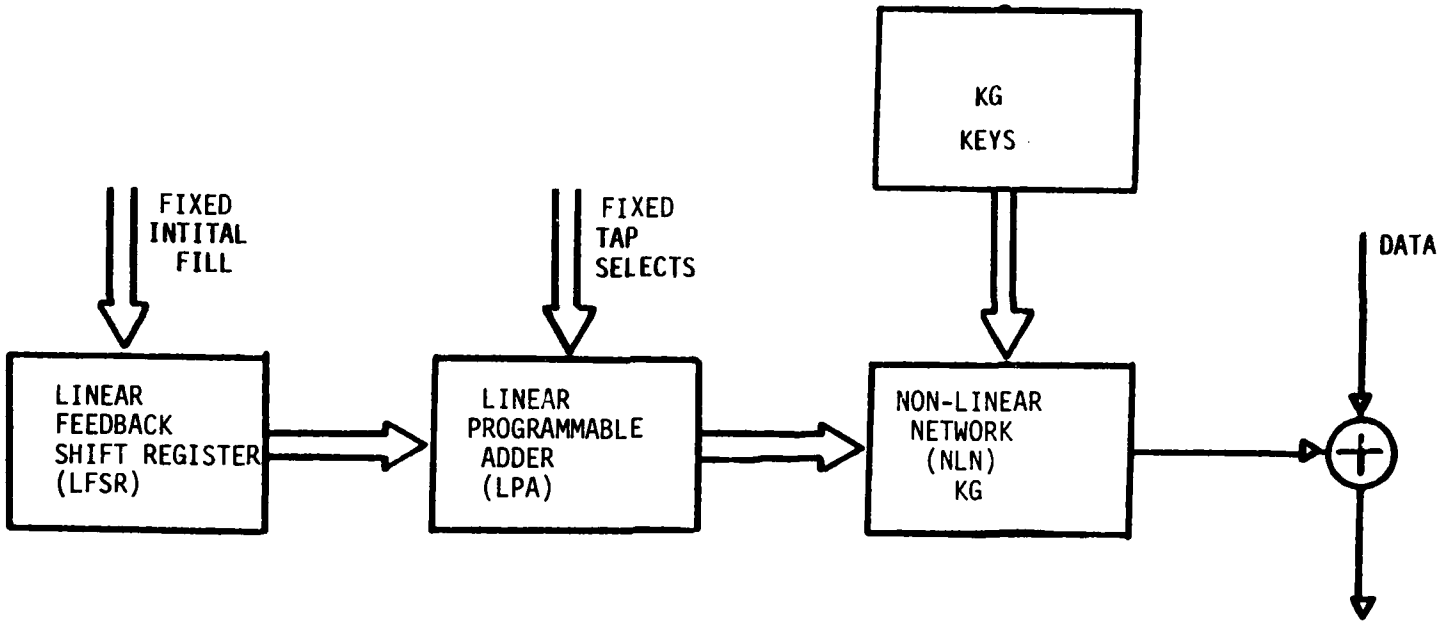


Figure 4.4-3 STATIC ENCRYPTION ARCHITECTURE

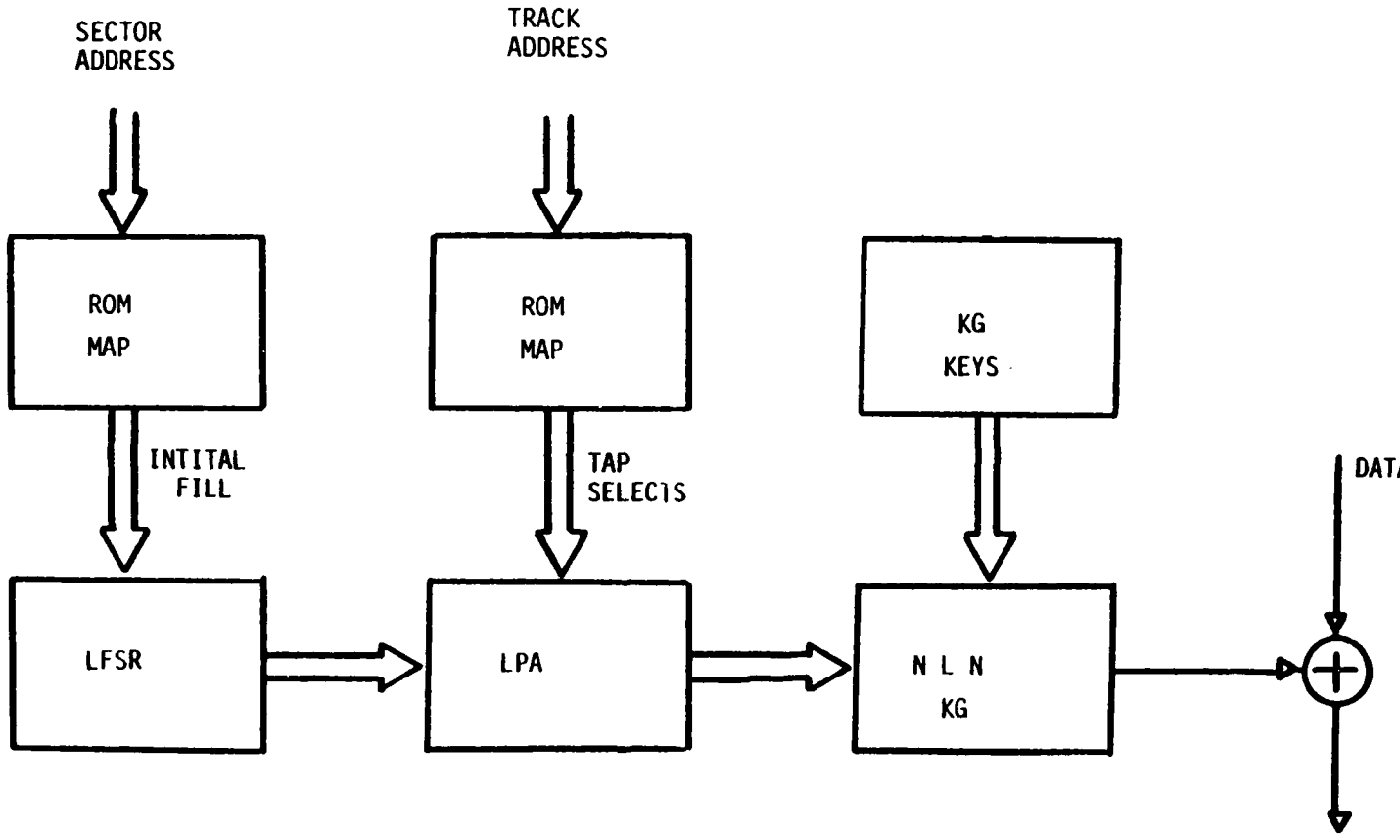


Figure 4.4-4 Random Access Static Encryption Architecture

compartment. The primary advantage gained is that the worst case KG key requirement is reduced from 8,000 keys to 80 keys and hence is more feasible to implement. Furthermore, failure of the weaker user separation within a security level and compartment could only violate the need-to-know principle, i.e., it would not violate clearance access level. As such, the user "number" is a secondary key.

In detail, this concept requires a unique key (and hence unique code sequence) for each security level and compartment. This code sequence is then divided among the users, i.e., each user is restricted to an assigned portion of the total code sequence. If 100 users are separated this way over a statically encrypted 300 Mbyte disk the basic code sequence length required would be greater than 2.4×10^{11} bits:

$$100 \text{ users} \times 300 \text{ Mbytes} \times 8 \text{ bits/byte} = 2.4 \times 10^{11} \text{ bits} \qquad \text{(Eq. 4-1)}$$

If this is implemented by a max length LFSR in the previously proposed architecture it's length would need to be greater than 37 bits ($\log_2 2.4 \times 10^{11}$), a requirement which is easily met.

Figure 4.4-5 shows an implementation of this concept. The length 63 maximum length LFSR would easily support separation of thousands of users over hundreds of 300 Mbyte disks using less than .01% of the code sequence, as

$$\frac{1000 \text{ users} \times 100 \text{ disk drives} \times 300 \text{ Mbytes/drive} \times 8 \text{ bits/byte}}{2^{63} \text{ bits per code sequence}} \qquad \text{(Eq 4-2)}$$

$$= .000026 \text{ users/code sequence}$$

This user separation is achieved by using the secondary key (user number) to load a stored set of LPA taps that provide a user code phase offset. These LPA values would be chosen to insure code subsequence separation of all users, over the range of physical data locations. To provide separation for 2,000 users, eight 16K bit ROM's (2K by 8 bits each) can be used to provide one LPA value for each user. Only some 2K subsets of these LPA values will provide real code separation.

In this approach, the LFSR initial fill is created by the total physical disk address: drive, track and sector. However, a ROM look up table can

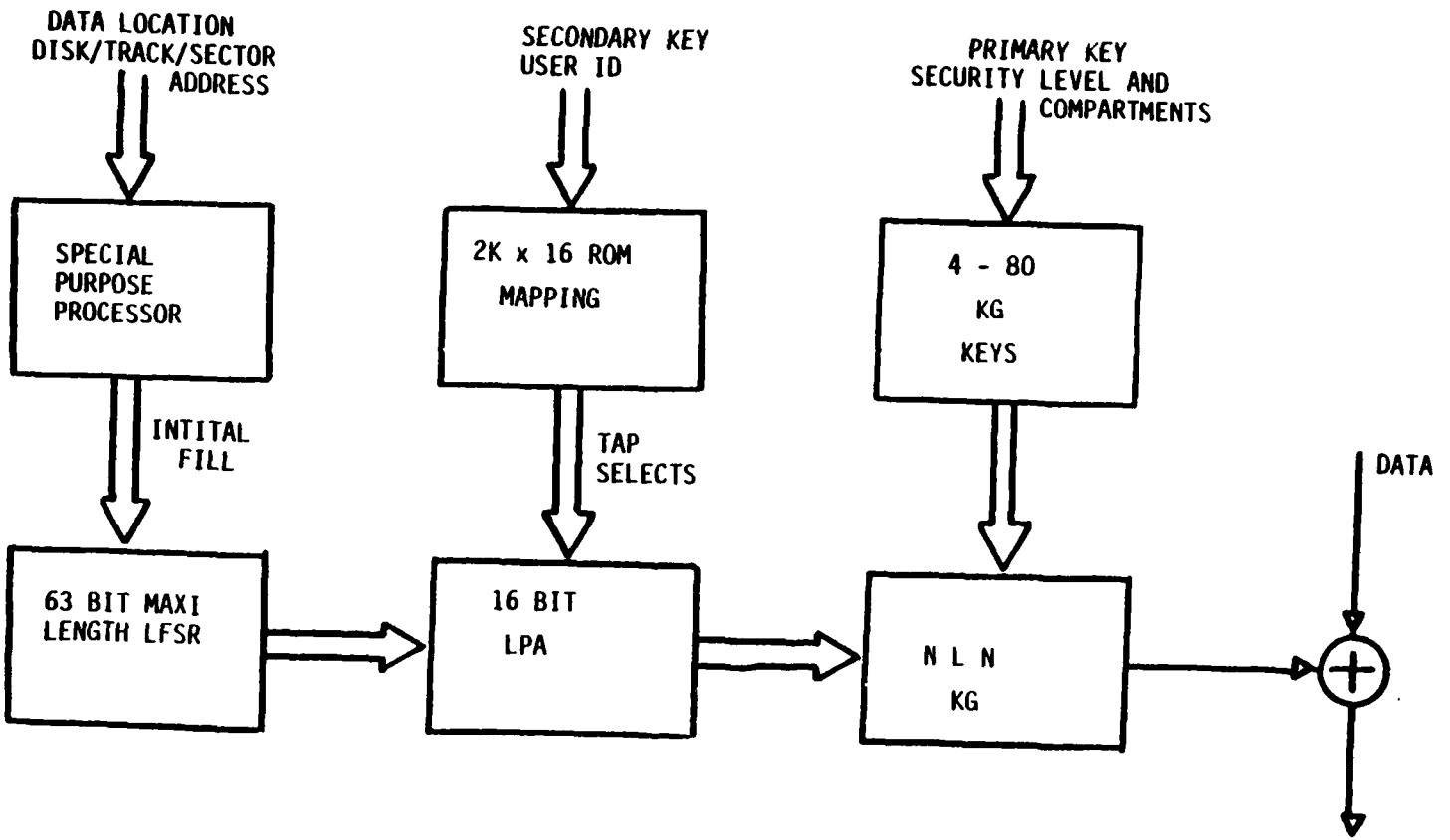


Figure 4.4-5 Static Encryption Architecture With User Separation

no longer be used to map the physical address to a PN state. The following section details the requirements necessary to achieve crypto sync for the above static encryption architecture.

4.4.1.5 Requirements for Conversion from Pseudo Random Sequence Index to PN State

This section describes the requirements for conversion of the physical disk address to a PN code state in the LFSR. The basic procedure is detailed in Appendix C. This section then summarizes the requirements for processing and discusses available technology to support these requirements.

High speed, random access, statically encrypted, disk I/O requires rapid conversion of random access disk location (i.e., disk number, track number, sector number) (the PN sequence index) into the associated LFSR PN state. The procedure for this involves first converting the disk drive number, track number, and sector into a logical data-storage bit address which is used as the PN sequence index. This PN sequence index is subsequently converted to the associated PN state for the LFSR. The first operation is very simple, requiring only three integer multiplies and two integer additions. The second process is quite complicated and is based upon the representation of a length n shift register as a primitive polynominal in a Galois Field. The mathematics are described in Appendix C. Figure 4.4 - 6 summarizes these requirements in terms of the number of bits in the LFSR, n.

While algorithmically simple, the n/2 n by n bit integer multiplies and the $n^2/4$ n-bit exclusive-or's required for this conversion generates a large processing load for even moderate length LFSR's, $37 < n < 63$. The data storage and processing load for keys of length 41 and 63 are shown in Figure 4.4-7. These requirements are just barely in reach of top-of-the-line microcomputers such as the 8086 with a high speed math chip such as the 8087. Because the amount of code required is very small (approximately one hundred lines of assembly language code), the software can be certified if required.

An alternate to the high-end microprocessor approach is building a special processor for this purpose. Current bit-slice technology can easily meet the speed required for this range of key. This approach trades an increase in

cost (hardware and software development), physical size and power requirements for a reduced synchronization latency and longer feasible LFSR code.

It should be noted that the requirements in Figure 4.4-7 reflect a 20% utilization allocation at 20 accesses/second. If the system design supported pre-fetching several future disk accesses, a trivial pipelining (buffering) implementation would cut these requirements by a factor of 5 at 20 accesses/second i.e., 200 to 500 KIPS (thousands of instructions per second). However, this approach would only solve the throughput problem: the additional latency degradation would still remain.

Category	Worst Case	Average
Memory	4n by n bits	4n by n bits
Integer Multiplies	n(n bit by n bit)	n/2 (n bit by n bit)
Integer Divides	1(n bit by n)	1(n bit ny n)
Exclusive Ors	n^2(n bit XORs)	$n^2/4$(n bit XORs)
Table look-ups	n^2 + n	-

Figure 4.4-6
LFSR Index to State Conversion Requirements

Processor Requirements* \ n=	63 bits	41 bits
Bytes of ROM	2,048	1,024
n x n Integer Multiples/sec	3K	2K
n x m Integer Divisions/sec	20	20
n bit Exclusive ORs/sec	100K	42K
Approximate MIPS (16 bit words) (3:1 ratio)	2+	1-

*assume 20% utilization at 20 accesses per second

Figure 4.4 -7
Data Storage and Processing Loads for LFSR Length 63 and 41

4.4.1.6 Current Disk Technology Drives Static Encryption Requirements

High speed random access disk technology will determine the crypto requirements necessary to provide static data base encryption with little DBMS performance degradation. The primary characteristics of interest are transfer

rate and access latency. Figure 4.4 -8 lists characteristics of a common 300 Mbyte CDC moving head disk that is typical of current disk technology. Actually performance is generally well below the maximum performance specified and is highly dependent upon actual data structure and accessing strategy. This is especially true of multi-user DBMSs where performance is typically limited to an average of 20 accesses/second. Some of the newer disk technology supports peak throughput rates as high as 10 Mbytes/sec.

Thus, it is reasonable to require an encryption system to support random access disk I/O at a 1.2 Mbyte/sec transfer rate with 20 accesses/second. This implies that a crypto throughput rate of at least 10 Mbd is required. It must also support 20 crypto syncs per second to accommodate the 20 random accesses per second. If a 20% utilization is allocated to this task, then there is a requirement to meet a time line of 10 ms from track/sector selection to crypto sync. At 10 Mbd this also requires the crypto sync/transient period to be less than 10 Kbits. These crypto requirements are summarized in Figure 4.4-9.

4.4.1.7 Extension of Static Encryption to Multi-Level,Multi-Compartmented Files

This section describes an application of static encryption to the multi-level files, as an additional technique to those described in Section 4.1. At the lowest level of security granularity, each data attribute would have a separate security classification. This requires an expansion of the static encryptor architecture to support demuxing and muxing of data based on a security tag and multiple KG's.

Figure 4.4 -10 is an example of this type of architecture. This architecture is built on the previous static encryption architecture with random access data location handled by the initial load of the LFSR and user separation provided by an LFSR code phase offset implemented via an LPA assignment per user. This linear PN sequence now feeds a number of non-linear networks (KG's) with each KG output sequence gated by the current security tag for the data being encrypted or decrypted. The data flowing through is then encrypted/decrypted with one of four security level KG's and zero to n security compartment KGs, where n is the max number of overlapping security compartment classifications allowed. Demuxing and remuxing of security classification tags and data is also required at the two data interfaces.

Capacity	300 Mbytes (2.4 Gbits)
Peak Xfer Rate	1.2 Mbytes/sec (10 Mbd)
Data Organization	19 cylinders 14,880 tracks 78 sectors/track 256 bytes/sector 1.1M sectors
Access Times	55 ms wc seek 16.7 ms wc latency (3600 RPM Synchronous Motor)
Access Performance	30 access/sec max (track read/random access assumptions)

Figure 4.4-8 Typical Disk Characteristics

Throughput	>10 Mbd
Mode	Asynchronous
Crypto Sync/Transient	<10 ms
Crypto Sync Period	<100 Kbits

Figure 4.4-9 Crypto Requirements for Static Disk Data Base Encryption

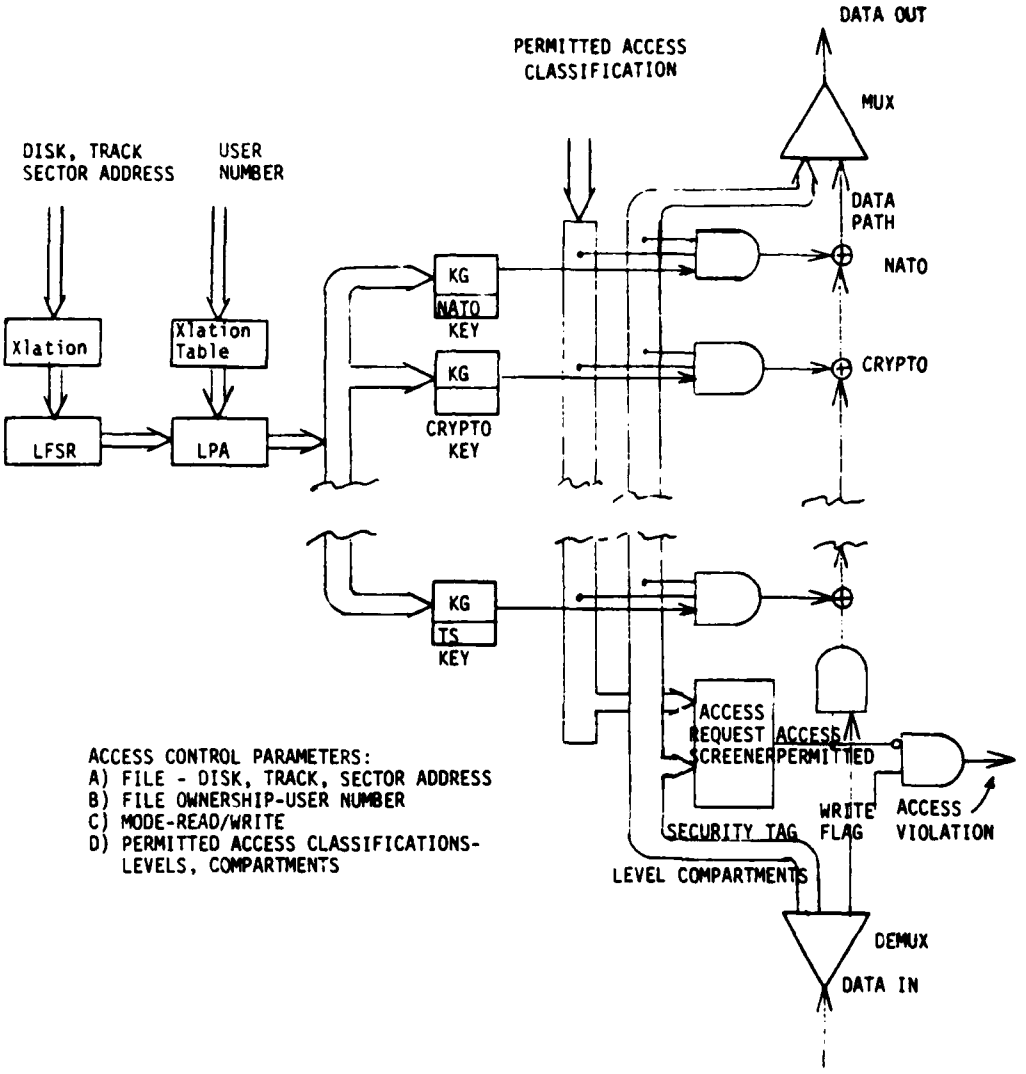


Figure 4.4-10 Static Encryption Architecture for Multi-Level Compartment Files

This change of security granularity from the disk access level to the data item level also requires that all security tags be screened to insure security. That is, an unclassified user cannot be allowed to read the classified data in a file containing unclassified and classified data. Therefore, the KG's output PN sequences require gating with the access classifications permitted for the current access as well as the security tag of the data item. Violation of the current access classifications permitted merely results in blanking of the data for read operations because this is allowable given the concept of multi-level, multi-compartment files. However, during write operations violation of current access classifications permitted should result in termination of the operation and immediate notification of the security monitor because such action constitutes a security compromise (unauthorized declassification) and/or a data integrity (spoofing) threat.

There are two disadvantages with this approach. The primary disadvantage is the high overhead associated with a security tag for each item of information. This is not only true for disk throughput which could be diluted as much as 50% but even more true of the software and special hardware in the DBMS processor which must generate these security tags for write operations. The other disadvantage is the number of KG units required, apparently 9 to 24 for a 4 level 5 to 20 compartment assumption. However, if a user is restricted to only a single compartment at a time, then the number of KG's can be reduced if the KG's are capable of storing a number of compartment keys with only a single key selected for a given access. In addition, because the encryption of unclassified data is optional, the actual number of KG's required should be under eight.

4.4.1.8 Summary/Base Line Design

It is clearly feasible to provide complete data separation for 4 DoD security levels, up to 20 security compartments, and several thousand users over 10 Gbytes of high speed random access disk storage with data item security granularity by using the base line architecture described here to provide static encryption with only minor degradation of disk performance.

Figure 4.4 -5 of section 4.4.1.4 shows the baseline architecture. For each disk access, the disk, track and sector address is converted by a dedicated processor into an initial load of a length, 63 maximum length linear feedback

shift register (sequence length 10^{19} bits). The user identifier of the owner of the file being accessed is used as a secondary key to select a code phase offset via an assigned set of taps for a length 16 linear programmable adder (LPA). These LPA values are empirically mapped to the user identifier to provide user separation of at least 10^{11} bits (≈ 10 Gbytes). If each access is constrained to a single security level and a single optional compartment, then a single KG with storage for 80 keys can be used to provide separation for all combinations of security level and security compartment. The KG key is selected for the security level and compartment of the current access to allow for the KG sync. (This KG is assumed to be a non-linear network with a length m shift register and no internal feedback). At the end of this crypto transient this KG network is now in crypto sync and is ready to either encrypt or decrypt the data access being made. Based on current disk technology, (see Section 4.4.1.6) this KG network must run at at least 10 Mbd and the computation of the initial LFSR load and the crypto transient should average less than 10 ms to keep from appreciably degrading disk I/O performance. If security granularity is provided by security tags at the data item level, then this architecture must be expanded to support mult -level and multi-compartment data items within a single access. This requires n + 3 KG's and an extensive gating network as shown in Figure 4.4-10 where n is the number of disjoint subsets that the set of security compartments can be broken into such that the elements of each subset are mutually exclusive for a single access. It is expected that almost all applications will require no more than 8 KG's.

As data flows through this expanded network, the security tags are extracted. Data items with the security tag gate which set of KG's encrypt or decrypt each data item. Encryption of security tags and unclassified data is optional. Each data item may be encrypted/decrypted by one or more KG's. If the security tag violates either security level and/or compartments authorized for the current access, the data is blanked for a read operation or the access is terminated and the security monitor notified for a write operation.

4.4.2 End to End Encryption (E^3) and Link Encryption Applications

End-to-end link encryption can be used to perform authentication, provide secure communication links, and for this application, potentially provide

user crypto separation at the user terminal level. The basic approach is to encrypt all data transfered between the HSF and a user's terminal in a user-unique key. This requires a single key crypto in each terminal and a multi-key crypto and possibly a key generator at the HSF as shown in Figure 4.4-11.

In practice, such an approach is only compatible with a back-end Internal Bypass Multi-level DBMS approach and requires further customizing of the DBMS to support this feature. The major advantage of this approach is that by having the ACS simultaneous control data base accesses and E^3 key selection it can enforce discretionary security for terminal users.

A more general application of End-to-End encryption is to provide link encryption for secure communications links in a transparent fashion.

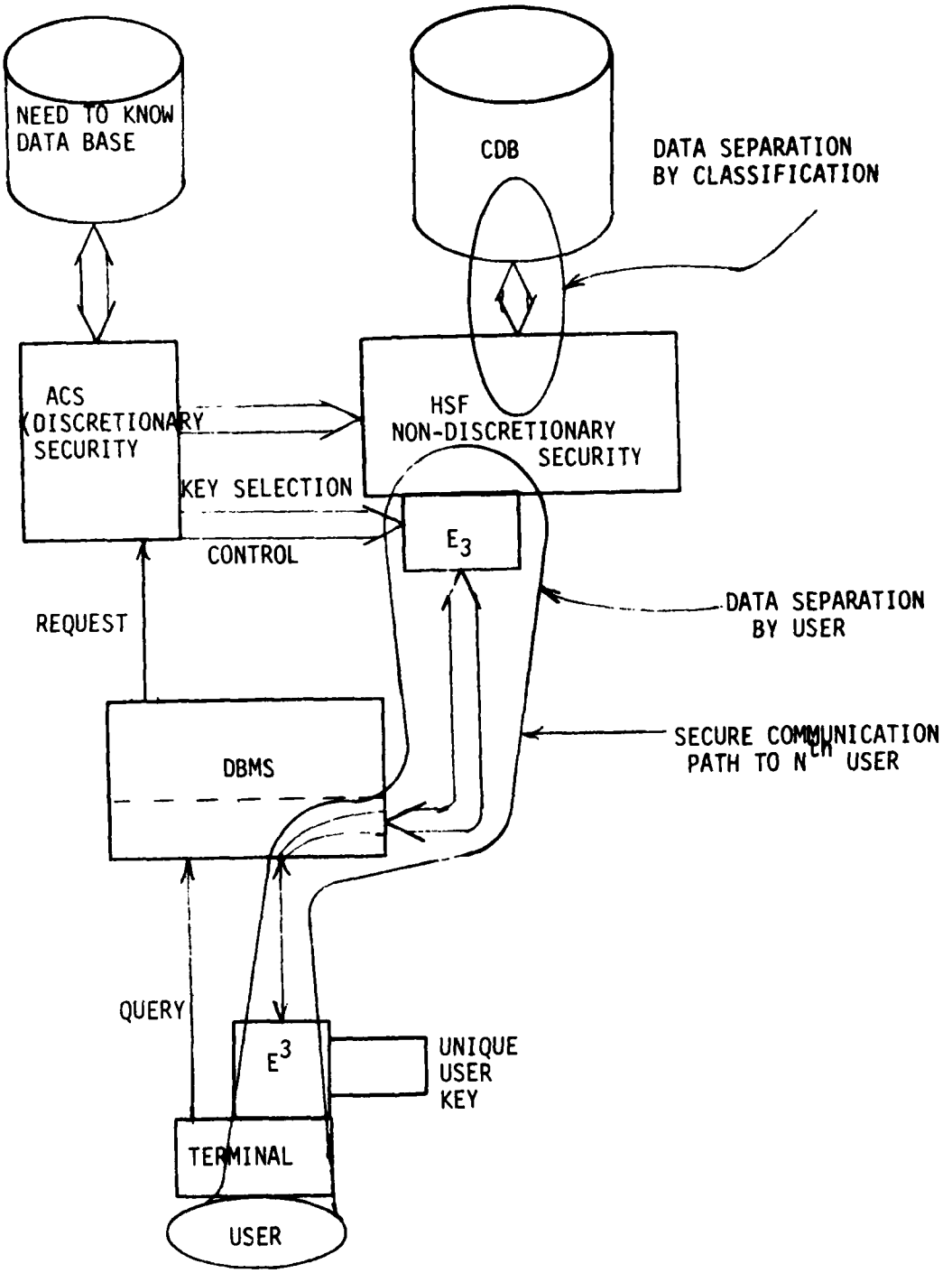


Figure 4.4-11 ACS Translation of Data Separation by Classification And Need-To-Know To Data Separation by User

5.0 FUNCTIONAL DESCRIPTIONS

A complete functional description of a system that meets the requirements of a specific application normally takes the form of functional specifications (e.g. B specs.). These are the baseline for subsequent production specifications (C specs.). Because there are no application baseline requirements, e.g. number of security levels and category combinations, number of concurrent compartments, number of terminals, data base size, number of queries per second to support, reliability requirements, data base structure, etc., this normal top down approach cannot be used. Instead, a bottom up approach has been used in this report.

The information in Section 4 is the basis for the top level of a production specification. Section 5.1 contains some general system functional specifications and design goals and more detailed subsystem functional specifications and design goals. These general system functional specifications and design goals basically partition functional responsibilities among subsystems and call for relatively minimal degradation of central data base throughput and latency. the subsystem specifications and design goals are much more detailed as they are based on current standard disk technology capabilities. For example, the central data base subsystem specifications are standard disk specifications and other ACS and HSF subsystem specifications were chosen to be compatible with these.

Section 5.2 contains four example functional system descriptions with different features. These systems are:

- o Front-End Architecture Without Encryption
- o Front-End Architecture With Static Encryption
- o Back-End Architecture Without Encryption
- o Back-End Architecture With Static Encryption

Finally Section 5.3 contains a functional description of the alternate architecture approach that employs end-to-end encryption, a back-end DBMS, and secure message switching (front-end I/O) processors. This can be considered a major variation on the last system described in Section 5.2.

5.1 Subsystem Specifications

The following five subsections present general system design requirements and goals for the four major Secure DBMS subsystems. These specifications are referred to in the functional descriptions of the four systems presented in Section 5.2. In most cases, the specifications are general and apply to most systems. However, in certain cases, a particular example system will take exception to some specifications and will require that additional specifications be stated. In these cases, any specification modification will be noted in the individual system implementation discussion section.

This section contains the following specifications:

5.1.1 General System Specifications (SYS);

5.1.2 Central Data Base Subsystem Specifications (CDB);

5.1.3 Access Control Subsystem Specifications (ACS);

5.1.4 Hardware Security Filter Subsystem Specifications (HSF);

5.1.5 DBMS Processor Subsystem Specification (DBP).

Section 5.1.5 also describes the requirements for the "secure" operating system (OS) employed by the DBMS processor. The above abbreviations in parentheses are used as prefixes to the applicable specifications. A suffix for each specification is created from the following schema:

Number Range	Type of Specification
11-39	Common Subsystem Specifications
41-49	Additional Specifications for Systems Without Encryption
51-69	Additional Specifications for Systems With Encryption
71-99	Design Goals

In the course of discussing the applicability of these specifications to the system implementations of Section 5.2, references are made to various systems by number, as follows:

System 1: Front-End DBMS Without Encryption

System 2: Front-End DBMS With Encryption

System 3: Back-End DBMS Without Encryption

System 4: Back-End DBMS With Encryption

System 5: Alternate Back-End DBMS Configuration

5.1.1 General System Specification

The general specifications for the implementation of a secure DBMS are presented in this section. These specifications are divided into four categories:

- Common Specifications - applicable to all secure DBMS designs — Figure 5.1.1-1
- Additional Specifications for Secure DBMS Designs Without Encryption — Figure 5.1.1-2
- Additional Specifications for Secure DBMS Design With Encryption — Figure 5.1.1-3
- Additional Secure DBMS Design Goals — Figure 5.1.1-4

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
SYS - 11	The access control subsystem shall provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts.	
SYS - 12	The access control subsystem shall not degrade central data base throughput and shall not increase average access latency more than 100ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic).	200 ms latency permitted for Encryption system 2, 4, & 5
SYS - 13	The hardware security filter subsystem shall enforce the selected access control policy for each DBMS subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their interconnections.	
SYS - 14	The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., additional throughput and access time degradation vs. effective utilization shall be less than 3dB.	
SYS - 15	The DBMS processor subsystems will use a secure OS.	Not required for system #5
SYS - 16	The access control subsystem shall be a system high, collection (receive only) node except for interfacing control lines to the Hardware Security Filter Subsystem. This control data shall be internally audited by the hardware security filter subsystem for compliance with access control policy.	Not applicable for Design # 5
SYS - 17	The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator.	
SYS - 18	DBMS processor subsystems shall be physically independent and isolated from each other except for indirect connections provided by the other subsystems.	Some back-end designs may use a single processor (See Design 5)

Figure 5.1.1-1 Common Secure DBMS System Specifications

APPLICABILITY: SYSTEMS #1 and #3

SPEC ID	SPECIFICATIONS	COMMENTS
SYS - 41	The multi-level data base subsystem shall provide logical data separation through use of a unique security tag for each combination of security level and security compartment.	
SYS - 42	The hardware security filter subsystem shall tag all data written into the central data base and shall screen these security tags for all data read from this data base.	
SYS - 43	The hardware security filter subsystem shall allow the security officer to select which combinations of read/write, security level, and security compartment(s) will be allowed for each DBMS processor subsystem.	

Figure 5.1.1-2 Additional Specifications for Secure DBMS Systems Without Encryption

APPLICABILITY: SYSTEMS #2,4 and 5

SPEC ID	SPECIFICATIONS	COMMENTS
SYS - 51	The multi-level data base subsystem shall be statically encrypted with a unique key for each security level and security compartment.	
SYS - 52	The hardware security filter subsystem shall encrypt all data written into the central data base and shall decrypt all data read from this data base.	
SYS - 53	As a design goal the system shall be capable of transparent on-line re-keying of statically encrypted central data base.	

Figure 5.1.1-3 Additional Specifications for Secure DBMS Systems with Encryption

APPLICABILITY: ALL SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
SYS - 71	The tagging of data shall not degrade effective disk packing density more than 3dB.	Non-encryption systems only (#1 and 3)
SYS - 72	The DBMS subsystems shall minimize central data base utilization	Except System #5
SYS - 73	The system shall be fault tolerant with graceful degradation	Except System #5
SYS - 74	Standard off-the-shelf hardware and software shall be used or modified where practical	Except System #5
SYS - 75	The system shall minimize the amount of certified and/or trusted software.	Except System #5
SYS - 76	The central data base will have a 50% spare disk capacity.	
SYS - 77	The system shall be capable of supporting 20 central data base accesses per second with a total throughput of 250 Kbytes/sec.	Except encryption System 2,3 and 5: requires 40 access/sec; and 500 Kbytes/sec (System #5)

Figure 5.1.1-4 Design Goals for the Secure DBMS System

5.1.2 Central Data Base Subsystem Specifications

This section describes the specifications for the Central Data Base Subsystem as described functionally in Section 3.3. These specifications are presented in three parts:

- Common Central Data Base Subsystem Specifications — Figure 5.1.2-1
- Additional Specifications for the Central Data Base Subsystem For Systems Employing Encryption — Figure 5.1.2-2
- Design Goals for the Central Data Base Subsystem — Figure 5.1.2-3

Actual implementations for these specifications are detailed in Section 5.2.

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
CDB - 11	Multiple drives shall be used.	
CDB - 12	50% spare storage capacity for the data base shall exist.	
CDB - 13	Peak transfer rate shall be at least 1.2 Mbd	
DCB - 14	Worst case access time shall be less than 55ms	
CDB - 15	Worst case latency shall be less than 17ms	
CDB - 16	Each disk controller shall provide a track buffer and flow control with control inputs from HW security filter subsystem.	
CDB - 17	The central data base subsystem shall provide high subsystem availability.	

Figure 5.1.2-1 Common Central Data Base Subsystem Specifications

APPLICABILITY: SYSTEMS 2, 4, and 5

SPEC ID	SPECIFICATIONS	COMMENTS
CDB - 51	There shall be two physically independent functionally redundant interfaces to the hardware security filter subsystem.	
CDB - 52	Redundant disk controllers shall be used. Each controller shall have access to all disk drives.	
CDB - 53	The disk system shall allow simultaneous dual port reads of any single disk drive and simultaneous accesses of any type to separate drives.	
CDB - 54	High subsystem availability will be achieved through a fault tolerant design employing some redundancy. Single point failures may degrade subsystem and system performance and/or result in temporary loss of portions of the data base i.e., only graceful degradation required vs. a hot standby. As a design goal, any single point failure will not degrade subsystem performance more than 3dB.	

Figure 5.1.2-2 Additional Specifications for the Central Data Base Subsystems Employing Encryption

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
CDB - 71	This subsystem shall consist of standard disk drives and modified or custom disk controllers configured so as to provide very high subsystem availability.	
CDB - 72	The disk controllers shall provide separate control and data interfaces.	
CDB - 73	The disk controller will have a 0.9998 availability	Except by encryption system (systems 2,4,& 5)
CDB - 74	Mean Time to Restore (MTTR) shall be an hour to reload the last check pointed version of each "lost" file	
CDB - 75	The disks will have an 8000 MTBF.	
CDB - 76	Winchester disk technology will be used to achieve the strict reliability and maintainability requirements.	

Figure 5.1.2-3 Design Goals for the Control Data Base Subsystem

5.1.3 Access Control Subsystem Specifications

This section presents the design requirements for the access control subsystem (ACS) previously described functionally in Section 3.3. These specifications are presented in three parts:

- Common Access Control Subsystem Specifications — Figure 5.1.3-1
- Additional specifications for Access Control Subsystems employing encryption — Figure 5.1.3-2
- Design goals for the Access Control Subsystem — Figure 5.1.3-3

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 11	This subsystem shall consist of an Access Control Subsystem (ACS), custom hardware interfaces to the DBMS subsystems and Hardware Security Filter Subsystem, and trusted,certified software.	
ACS - 12	This subsystem shall provide arbitration of central data base access requests, associated directory operations, central security monitoring, and support maintenance activities.	
ACS - 13	This subsystem shall screen all requests for compliance with security level and compartment access privileges.	
ACS - 14	In conjunction with the secure OS in the DBMS processor subsystems enforce discretionary security.	System #5 enforces discretionary security through an E^3 technique.
ACS - 15	The ACS process functions will be implemented by trusted/certified software.	
ACS - 16	The ACS subsystem shall reside in a dedicated processor(s) at a system high node and specialized hardware interfaces to the other subsystems.	
ACS - 17	This subsystem shall interface with the numerous DBMS processor subsystems by low speed message links (2.4Kbd) capable of supporting 10 messages per second each.	System #5 specs 19.2Kbd and 40msg/sec
ACS - 18	This subsystem shall be capable of receiving messages at the rate of 10 messages per second per DBMS processor subsystem.	System #5 requires 40 msg/sec

Figure 5.1.3-1 Common Access Control Subsystem Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 19	As necessary, these links shall use a front end I/O processor(s) to implement a forward error correction and permitting the DBMS system interface to be unidirectional, permitting the ACS to be operated as a system high collection node.	
ACS - 20	The front-end I/O processor(s) shall be modified so that all data and control communication flows from it into the dedicated processor(s) i.e., it may only write shared memory and its operation is independent of the dedicated processor(s).	
ACS - 21	This subsystem shall interface through local peripherals with the security officer and possibly the data base administrator.	
ACS - 20	This subsystem shall interface with the Hardware Security Filter Subsystem receiving status inputs and generating control outputs.	
ACS - 23	These control outputs to the HSF shall be audited by the Hardware Security Filter Subsystem for compliance with access control policy. Logically this system receives central data base access requests, directory requests and security monitor data from the DBMS processor subsystems, sends central data access commands to the Hardware Security Filter Subsystem, and exchanges security data with the security officer.	
ACS - 24	Each DBMS subsystem shall have no more than four central data base access requests pending.	
ACS - 25	This subsystem shall be capable of servicing a valid access request in 15ms.	
ACS - 26	This system shall be capable of 20 central data base accesses per second throughput.	

Figure 5.1.3-1(Cont'd) Common Access Control Subsystem Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 27	All messages received shall be logged on mag tape to form an extensive audit trail.	
ACS - 28	This subsystem shall maintain the central security data base. This data base shall contain the access control policy of each DBMS processor subsystem, a list of all users and their access privileges (e.g. maximum security level security compartment(s) permitted, create/delete/read/write/permit/rescind capability), and a list of users currently logged on each DBMS processor.	
ACS - 29	This subsystem shall maintain security and physical directories. The security directory information shall include the owner, security level and compartment(s), time of last read and write access, and an access control list. The physical directory information shall permit a logical file name and logical sector address to be translated into a physical disk address.	
ACS - 30	The security officer shall have the capability to modify the central security data base and dump selected portions of the central security data base and/or directories.	
ACS - 31	This subsystem shall also implement security checking and implementation of valid directory commands. These commands shall include create, delete, permit and rescind.	
ACS - 32	This subsystem shall be capable of handling 256 users and 32,768 files.	
ACS - 33	This system shall handle DBMS processor subsystem requests on a real time basis.	
ACS - 34	This system may handle MMI functions with the security officer in a background mode.	

Figure 5.1.3-1(Cont'd) Common Access Control Subsystem Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 35	This subsyster shall have no programming capability and security officer control shall be limited by the interfacing application program to report and/or query requests, and data base changes which are logged.	
ACS - 36	Diagnostics shall be run in an off-line mode.	
ACS - 37	All software shall be certified and/or trusted.	

Figure 5.1.3-1(Cont'd) Common Access Control Subsystem Specifications

APPLICABILITY: ENCRYPTION ACS SYSTEMS 2, 4 and 5

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 38	The use of a disk optimizer to handle dual ported disk accesses shall be considered with certification cost weighed against system performance.	
ACS - 39	This dual ported disk controller with or without an optimizer shall be capable of being run as two independent disk handlers with an old key-new key distinction during the rekeying interval.	

Figure 5.1.3-2 Additional Specifications for Access Control Subsystems Employing Encryption

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
ACS - 71	As a design goal the software shall minimize certification cost, e.g. it will have such features as structured design, structured programming, modularity, minimum size, simplicity, etc.	
ACS - 72	As a secondary design goal this subsystem will maximize performance and functionality.	
ACS - 73	As a design goal this subsystem will operate fail safe from a security viewpoint.	
ACS - 74	As a design goal this subsystem will provide high availability.	

Figure 5.1.3-3 Design Goals for the Access Control Subsystem

5.1.4 Hardware Security Filter Subsystem Specification

This section describes the requirements for the design of the Hardware Security Filter Subsystem (HSF), as described functionally in Section 3.3. This specification is presented in three separate segments due to the differing system requirements for the various system implementations. These implementations are:

- o Common Hardware Security Filter Subsystem Specifications — Figure 5.1.4-1
- o Additional Specifications for HSF Subsystems Not Employing Encryption — Figure 5.1.4-2
- o Additional Specifications for HSF Subsystems Employing Encryption — Figure 5.1.4-3

The specifications for the encryption systems of Figure 5.1.4-3 are presented in two sections. The first half apply to all systems employing encryption data separation techniques. However, because of the uniqueness of the implementation of System #5 (Alternate Back end DBMS Configuration) some additional specifications are presented (HSF-68 through HSF-71) which supercede several previously presented specifications.

APPLICABILITY: ALL SECURE DBMS SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
HSF - 11	This subsystem shall enforce access control policy for each DBMS subsystem in a provable manner.	
HSF - 12	This subsystem shall allow the security officer to select and validate a separate access control policy for each DBMS subsystem.	
HSF - 13	This subsystem shall allow separate access control of each combination of read/write, security level, and security compartment(s). This capability will allow the security officer to configure a secure multi-level DBMS system.	Excepted by System #5
HSF - 14	Proof of such secure multi-level operation and access control policy enforcement shall be documented by a system security analysis (SSA).	
HSF - 15	This subsystem design shall not violate access control policy in the event of a single point failure. This requirement shall be proven by a security fault analysis (SFA).	
HSF - 16	This subsystem shall have a control interface to the Access Control Subsystem, control and data interfaces to the Central Data Base Subsystem, and a data interface to each DBMS Subsystem.	Encryption systems shall employ a redundant dual-channel design.
HSF - 17	This subsystem will receive control inputs from the Access Control Subsystem	See HSF - 56.
HSF - 18	These ACS control inputs shall be screened by this subsystem to validate compliance with the selected access, word count, data security level, data security compartment(s) if any, and DBMS processor (channel number).	System #5 extends this. See Spec. HSF - 61.
HSF - 19	A busy/idle status output shall be supplied to the Access Control Subsystem.	

Figure 5.1.4-1 Common Hardware Security Filter Subsystem Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
HSF - 20	This subsystem will provide control inputs to the Central Data Base Subsystem and receive status outputs from that subsystem. A flow control mechanism shall be provided for the data interfaces.	
HSF - 21	The worst case subsystem latency shall be less than 15ms where latency is defined as the interval from control data being supplied by the access control subsystem to an idle channel to the start of data transfer with the central data base subsystem excluding any latency in that subsystem. This latency period shall include validation of compliance with access control policy, and setup of the associated data paths.	Latency must also include KG key selection and crypto sync for systems with encyrption. See Spec. HSF - 57.
HSF - 22	Each channel shall be capable of supporting a 1.2 Mbyte throughput rate. The actual transfer rate shall be the clock rate of the DBMS subsystem interface with data over those links transmitted in packets.	
HSF - 23	Control of this data traffic flow shall be through ready/busy status semaphores.	
HSF - 24	Track buffers in the central data base subsystem will be used to match actual disk transfer rates to this subsystem's transfer rate.	
HSF - 25	This subsystem will reside in a separate system high compartment with access limited to the security officer and maintenance personnel.	
HSG - 26	This subsystem shall meet Tempest requirements.	
HSF - 27	This subsystem shall provide high subsystem availability.	

Figure 5.1.4-1 Common Hardware Security Filter Subsystem Specifications (Cont'd)

SPEC ID	SPECIFICATIONS	COMMENTS
HSG - 28	In the event of an access control violation, this subsystem shall sound a violation alarm and shut down the operation of the entire Secure DBMS.	Note that the access control subsystem prescreen requests for violations and hence any access control policy violation in this subsystem represents a compromise of the trusted security monitor software in the access control subsystem.

Figure 5.1.4-1 Common Hardware Security Filter Subsystem Specifications (Cont'd)

APPLICABILITY: NON-ENCRYPTION HSF SYSTEMS

SPEC ID	SPECIFICATIONS	COMMENTS
HSF - 41	This subsystem shall consist of a switching network and control and data logic that shall implement a logical exit guard for each data channel.	
HSF - 42	These exit guards shall enforce an access control policy for each DBMS subsystem.	
HSF - 43	This subsystem shall insure all data written to the central data base is logically tagged with a security level and compartment(s).	
HSF - 44	This subsystem shall screen all data tags to validate compliance with the access control policy	
HSF - 45	This subsystem shall either block or blank any data item whose security tag violates security policy.	
HSF - 46	Each combination of security level and security compartment(s) shall effectively be uniquely tagged.	

Figure 5.1.4-2 Additional Specifications for Hardware Security Filter Subsystems not Employing Encryption

APPLICABILITY: ALL ENCRYPTION HSF SYSTEMS (Unless Otherwise Noted)

SPEC ID	SPECIFICATIONS	COMMENTS
HSF - 51	This subsystem shall consist of redundant multi-key static encryptor/decryptor hardware, a functionally redundant switching network, and control logic that shall enforce an access control policy for each DBMS subsystem.	System #5 extends this requirement (HSF - 58)
HSF - 52	This subsystem shall statically encrypt all data written to the central data base and decrypt all data read from the central data base that complies with access control policy.	
HSF - 53	Each security level will be encrypted under one or more unique keys.	
HSF - 54	Each combination of security level and security compartment(s) shall effectively be encrypted under a unique key. This effective key may consist of multiple keys with a primary key for each security level and security compartment.	
HSF - 55	This subsystem shall allow on-line rekeying. This on-line rekeying shall result in old files read with the old key and rewritten with the new keys in a background mode while users have normal access capabilities, i.e., old and/or new key data shall be available to the user during this rekey interval. This will be accomplished by a dual channel design with independent KGs. This will allow one channel to operate with the old keys and the other with the new keys during this rekeying interval.	
HSF - 56	To support on-line rekeying of the data base and to enhance system reliability, this system will employ a dual channel design with independent KGs.	Supercedes HSF-16
HSF - 57	Latency shall be less than 15ms as prescribed in HSF - 21 and shall include the additional times associated with selection of KG keys and crypto synchronization with the disk address.	Extends spec HSF - 21

Figure 5.1.4-3 Additional Specifications for Hardware Security Filter Subsystems Not Employing Encryption

SPEC ID	SPECIFICATIONS	COMMENTS
HSF - 58	This subsystem shall consist of redundant multi-key static encryptor/decryptor hardware, a functionally redundant switching network, buffers, end-to-end E^3 dynamic encryption hardware, and control logic that shall enforce an access control policy for each processor connected, provide the capability of byte granularity in central data base updates, and allow an optional E^3 processor interface.	
HSF - 59	The interfaces to DBMS processors shall be unencrypted while data interfaces to the front end I/O processor shall be through an E^3 system with key selection made by the access control subsystem.	
HSF - 60	This subsystem shall provide internal buffering so as to provide a read (sector), modify (subsector), write (sector) capability.	
HSF - 61	Control data from the ACS shall include disk address, read/write access, word count, data security level/compartment(s) format, DBMS processor (channel number), E^3 key selection, and merging format or write format. This subsystem shall be capable of merging old data from an internal buffer, user data from a front-end I/O processor, and DBMS generated data from the back-end DBMS during write operations.	Supercedes HSF-18
HSF - 71	As a goal, single point failures shall degrade subsystem throughput no more than 50%.	

Figure 5.1.4-3(Cont'd) Specifications for Hardware Security Filter Subsystems Not Employing Encryption

5.1.5 Data Base Processor Subsystem Specifications

This section presents the requirements of the Data Base Processor Subsystem, as functionally described in Section 3.3. The specifications are presented in sections as follows:

- Common Data Base Processor Specifications — Figure 5.1.5-1
- Additional Specifications for the Back-End DBMS Process Subsystem Employing Encryption — Figure 5.1.5-2
- Design Goals for the Data Base Processor Subsystem — Figure 5.1.5-3
- Operating System Requirements for the Data Base Processor Subsystem — Figure 5.1.5-4

The separate specification for the back-end with encryption implementation is necessary due to the major difference in approach that this design represents compared to the other systems. Although not directly a DBMS Processor specification, the requirements for the operating system (OS) used by the DBMS processor are also presented in this section.

APPLICABILITY: ALL SECURE DBMS SYSTEMS (except as noted) EXCEPT SYSTEM #5

SPEC ID	SPECIFICATIONS	COMMENTS
DBP - 11	These subsystems shall consist of a standard computer system and peripherals, a secure OS, a multiprocessor relational DBMS, and custom and/or modified hardware/software interfaces to the Access Control Subsystem, and Hardware Security Filter Subsystem.	
DBP - 12	Each DBMS processor subsystem will reside in a separate physical compartment.	Also applies to System #5
DBP - 13	The only physical interconnections among these subsystems shall be indirect through secure links to/from the access control subsystem and hardware security filter subsystem.	
DBP - 14	The only logical interconnections among these subsystems shall be through the multi-level central data base.	
DBP - 15	These logical interconnections shall be restricted by the hardware security filter subsystem as to which combinations of read/write, security level, and security compartment(s) accesses are permitted for each DBMS subsystem.	
DBP - 16	From a user perspective these logical interconnections shall be further restricted by the secure OS and access control subsystem so as to enforce user discretionary security controls. This will be performed by funneling all user central data base access requests through the secure OS which will tag the request with the user's identity.	
DBP - 17	The access control subsystem shall then screen each request against the user's privileges as defined by the security officer.	Also applies to System #5.
DBP - 18	Each DBMS processor subsystem shall be further sub-compartmented so as to separate users and their peripherals from system personnel, e.g., operator and the main processor subsystems, e.g., processor, memory, disk, op console, etc.	

Figure 5.1.5-1 Common Data Base Processor Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
DBP - 19	Personnel access to these various sub-compartments will be supervised/restricted by the security officer.	
DBP - 20	Any remote users will be linked to these subsystems by secure links.	
DBP - 21	Use of user authentication techniques shall be considered.	
DBP - 22	Secure operation of these subsystems shall be detailed in an Automatic Data Processing/Standard Practice Procedure document.	
DBP - 23	This subsystem shall use standard off-the-shelf hardware and software to the extent feasible.	
DBP - 24	The use of smart terminals tailored to support and/or enhance secure operation and/or reduce the minimum skill level required to utilize this DBMS system will be considered.	
DBP - 25	The DBMS processor subsystem/access control subsystem interface shall support a 10 msg/second simplex message channel.	
DBP - 26	DBMS processor subsystem/access control subsystem interface shall employ an unidirectional query channel employing Forward Error Correction Coding.	Also applies to System #5
DBP - 27	The DBMS processor subsystem/hardware security filter subsystem interface shall support a 10Mbd full duplex channel.	Also applies to System #5
DBP - 28	The DBMS processor/HSF interface shall allow implementation of a flow control mechanism and shall support DMA operation	Also applies to System #5
DBP - 29	The DBMS processor/HSF interface will use two simplex serial fiber optic links with Manchester encoding (i.e., 10Mbd half rate encoded producing 20Mbd channel rate).	Also applies to System #5

Figure 5.1.5-1(Cont'd) Common Data Base Processor Specifications

SPEC ID	SPECIFICATIONS	COMMENTS
DBP - 30	The DBMS processor subsystem shall be capable of operation up to 100 meters from the access control subsystem and hardware security filter subsystem.	Also applies to System #5.

Figure 5.1.5-1(Cont'd) Common Data Base Processor Specifications

APPLICABILITY: SYSTEM #5 ONLY EMPLOYING ENCRYPTION

SPEC ID	SPECIFICATIONS	COMMENTS
DPS - 61	This subsystem shall consist of a standard computer, a custom multi-processor relational DBMS, and custom hardware/software interfaces to the Access Control Subsystem, the Hardware Security Filter Subsystem, and the Secure Message Switch (Front End I/O Processors Subsystems).	
DPS - 62	The DBMS shall support indirect secure multi-level data base queries.	
DPS - 63	The DBMS shall be capable of generating security level/compartment formats for multi-level files and merge formats for updates.	
DPS - 64	DBMS shall support integrity level controls for updates as well.	
DPS-65	This DBMS shall support smart multi-level terminals.	

Figure 5.1.5-2 Additional Specifications for the Back End DBMS Processor Subsystem

APPLICABILITY: ALL DBMS SYSTEMS EXCEPT AS NOTED

SPEC ID	SPECIFICATIONS	COMMENTS
DPS - 71	As a design goal each DBMS processor subsystem will support 20 on-line users with less than 60% processor utilization and less than 50% throughput and 70% latency degradation relative to single user operation.	
DPS - 72	As a design goal, the standard equipment selected shall support a secure OS.	Excepted by System #5
DPS - 73	As a design goal, the system shall use an efficient multi-processor relational data base.	Excepted by System #5
DPS - 74	As a design goal, the system shall support VM operation	
DPS - 75	As a design goal, at least 1 Mbyte of memory shall be available	
DPS - 76	As a goal, the DBMS subsystem shall exhibit good reliability	
DPS - 77	As a goal, the DBMS subsystem shall provide large program and data spaces	
DPS - 78	As a goal, the DBMS subsystem shall permit the processor and/or memory and/or I/O and/or disk to be upgraded without extensive modifications, i.e. good expandability with software impact limited to sysgen, and front-end I/O processor(s).	

Figure 5.1.5-3 Design Goals for the Data Base Processor Subsystem

APPLICABILITY: ALL SECURE DBMS SYSTEMS EXCEPT #5 WHICH DOES NOT REQUIRE A SECURE OS

SPEC ID	SPECIFICATIONS	COMMENTS
OS - 11	The multiprocessor relational data base used with the DBMS subsystem shall by modified, if required, to run under the secure OS.	
OS - 12	The secure OS shall be modified to include a smart terminal handler, if used	
OS - 13	The sysgen for the secure OS shall have the capability of restricting selected terminals to interactive query operations, i.e., no programming capability.	
OS - 14	The secure OS shall be modified to support VM operation	
OS - 15	The secure OS shall be modified to support central data base access requests. This will be done by replacing the standard disk handler with a new handler while maintaining the same logical disk interface. This modification shall still allow use of local disks for VM operation.	
OS - 16	The secure OS shall be recertified after these modifications.	

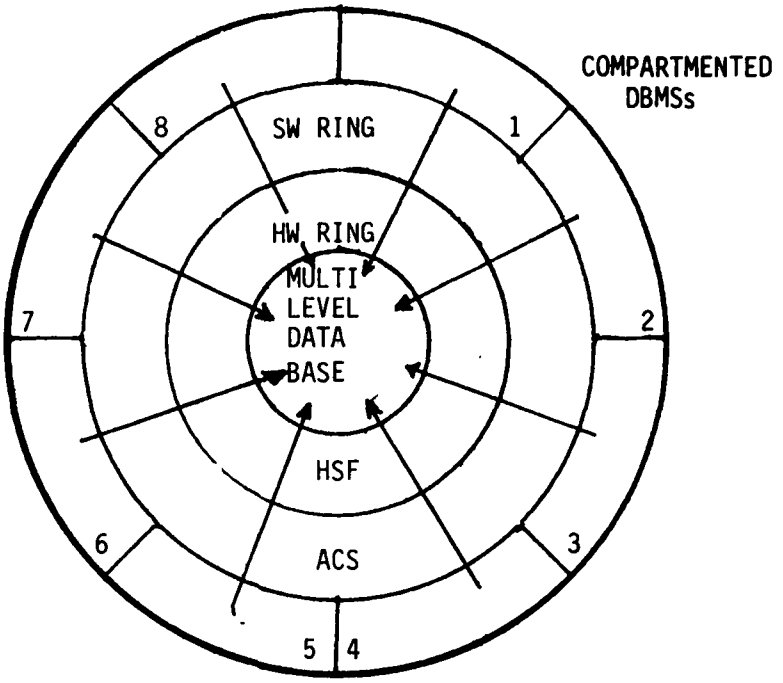
Figure 5.1.5-4 Operating System Requirements for the Data Base Processor Subsystem

5.2 Functional System Descriptions

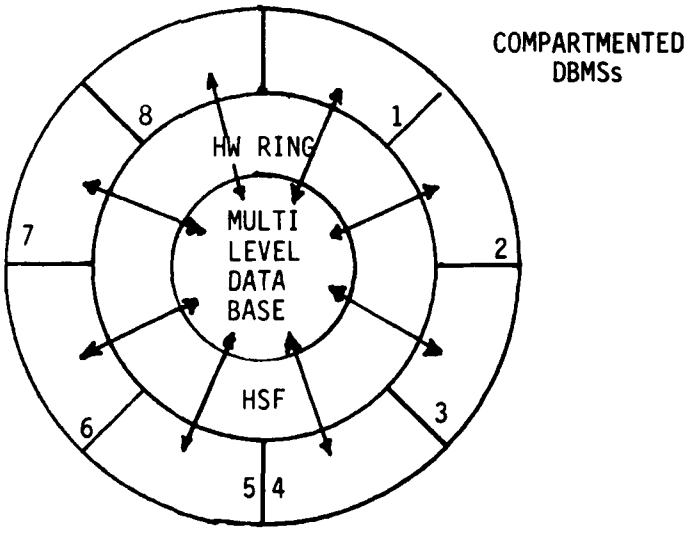
The primary differences between the four systems subsequently described in this section are the use or non-use of static encryption in the HSF and the use of front-end or back-end DBMS processors. In terms of the distributed architecture, the use of static encryption may result in a separate KG compartment (subsystem) with access limited to NSA cleared personnel. The back-end DBMS processors are likely to have one or more host processors supporting some of their users. While functionally different in terms of the basic distributed architecture, these systems are only minor variations on a theme.

These systems can conceptually be described in terms of protection ring structures consisting of a multi-level, multi-compartment central data base, a hardware security filter ring including a switching network and entrance/exit guards, a software security monitoring ring in the form of a central access control subsystem, and multiple compartmented DBMS processors, as shown in Figure 5.2-1. The security ring structures shown in this figure are rather unique in that each ring and each of the outer compartments consist of separate hardware elements in a distributed architecture. Because of this physical separation, it is relatively easy to prove, by control and data flow analysis, that hard access control of each combination of read/write, security level, and security compartment(s) can be enforced in hardware for each DBMS processor. Note that the control flow goes from the separate DBMS processors through the software (SW) monitoring and Hardware (HW) access control rings into the central data base, while the data flow is bi-directional through a single hardware security filter ring.

The remainder of this section is organized as follows:



CONTROL RING STRUCTURE



DATA TRANSFER RING STRUCTURE

Figure 5.2-1 Security Ring Structures of Distributed Multi-Level DBMS Architectures

5.2.1 Front-End DBMS Architecture Without Encryption

Functional Description

The distributed architecture that implements this access control structure is shown in Figure 5.2-2. The key element that makes this network a secure multi-level DBMS system is the hardware ring which includes a switching network and exit guards. These exit guards control the security level and compartments, read/write and channel (DBMS processor) combinations allowed, thus implementing a trusted security filter. The ACS monitors and arbitrates central data base access requests from the numerous compartmented front-end DBMS processors. Thus, this system consists of a central multi-level data base subsystem, a hardware security filter subsystem that provides an exit guard for each DBMS processor subsystem, a software security monitor subsystem (ACS) that also arbitrates central data base access requests, and multiple front-end DBMS processor subsystems.

Functional System Specification

This system shall consist of a central multi-level data base subsystem, a hardware security filter subsystem, an access control subsystem, and multiple front-end DBMS processor subsystems. The multi-level data base subsystem shall provide logical data separation through use of a unique security tag for each combination of security level and security compartment. The hardware security filter subsystem shall tag all data written into the central data base and shall screen these security tags for all data read from this data base. The hardware security filter subsystem shall allow the security officer to select which combinations of read/write, security level, and security compartment(s) will be allowed for each front end DBMS processor subsystem. The hardware security filter subsystem shall enforce the selected access control policy for each DBMS subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their interconnections. The access control subsystem shall provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts. The access control subsystem shall be a system high, collection (receive only) node except for control lines to the hardware security filter subsystems. This control data shall be internally audited by the hardware security filter subsystem for compliance with access control

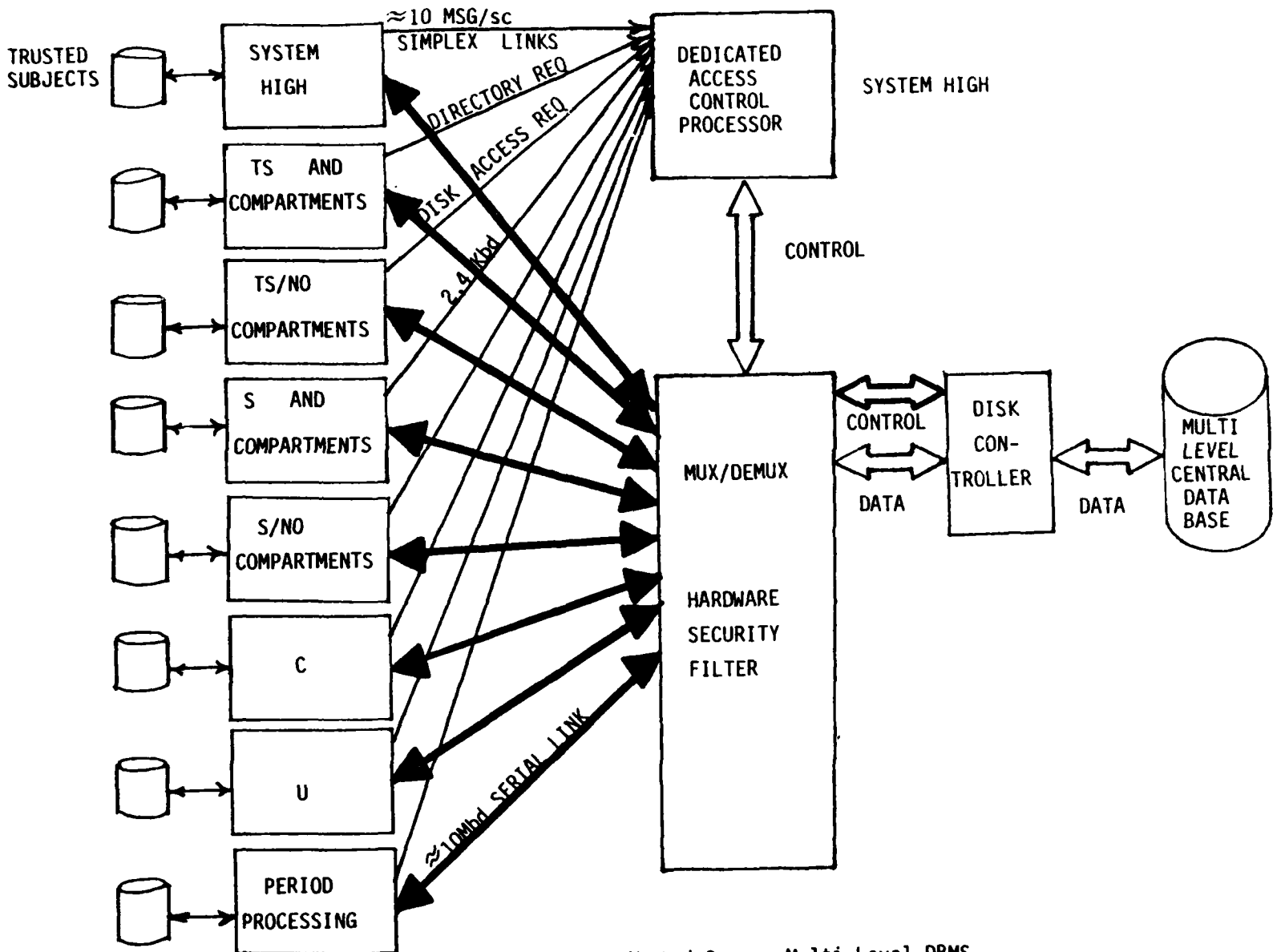


Figure 5.2-2 Distributed Secure Multi-Level DBMS Architecture

policy. The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator. The multiple front-end DBMS processor subsystems shall be physically independent and isolated from each other except for indirect connections provided by the other subsystems. The DBMS processor subsystems will use a secure OS.

The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., additional throughput and shall not increase average access latency more than 100ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic). As a design goal the tagging of data shall not degrade effective disk packing density more than 3dB.

As a design goal the DBMS subsystems shall minimize central data base utilization. As a design goal the system shall be fault tolerant with graceful degradation. As a design goal standard off-the-shelf hardware and software shall be used or modified where practical. As a design goal the system shall minimize the amount of certified and/or trusted software. As a design goal the central data base will have a 50% spare disk capacity. As a design goal the system shall be capable of supporting 20 central data base accesses per second with a total throughput of 250 Kbytes/sec.

Functional Subsystem Specifications

The following subsystem specifications and design goals from Section 5.1 are applicable:

Subsystem Specifications	11 - 49
Design Goal Specifications	71 - 99

5.2.2 Front-End DBMS Architecture With Static Encryption

Functional Description

The distributed architecture that implements this access control structure is shown in Figure 5.2-3. The key element that makes this network a

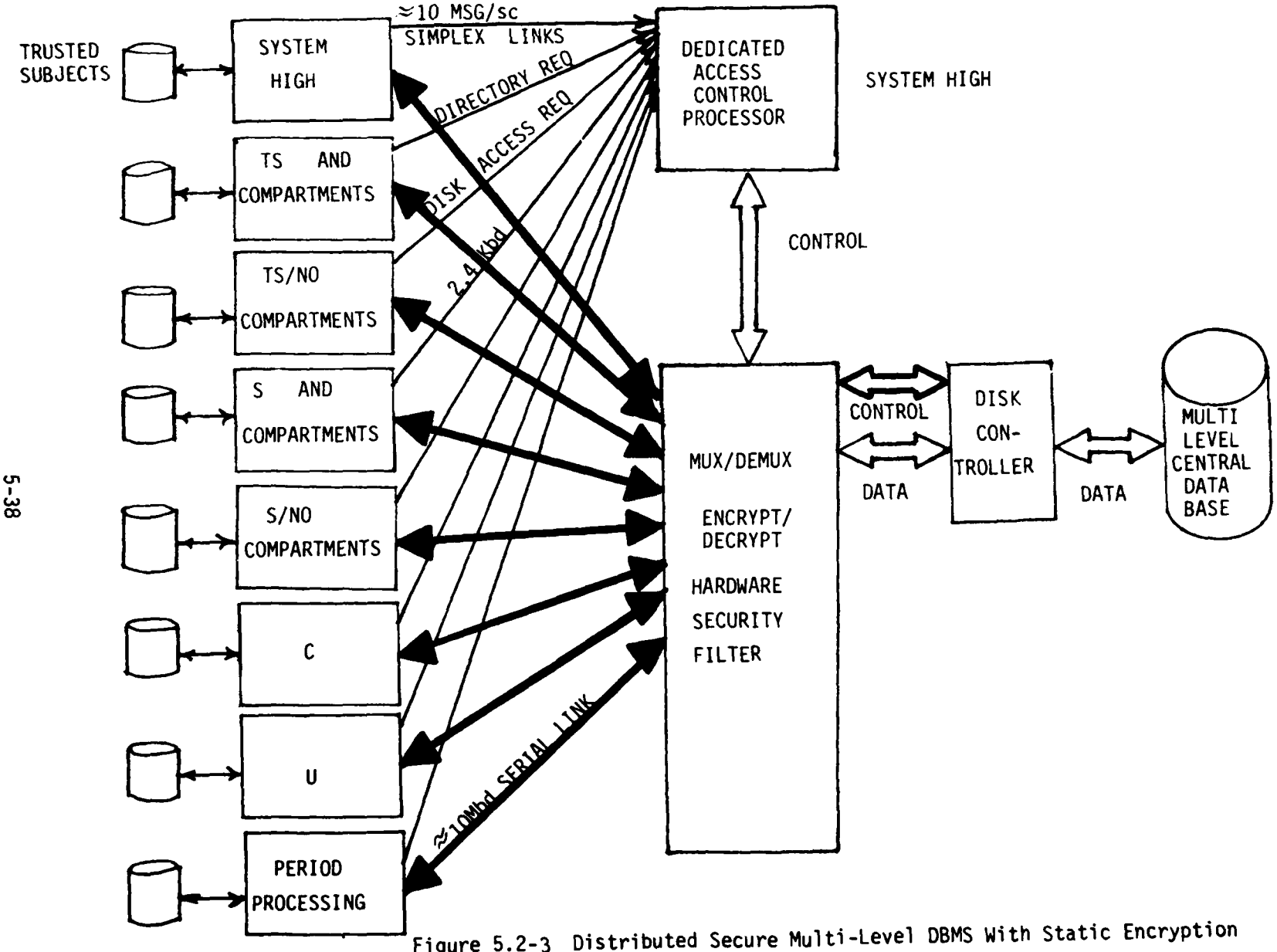


Figure 5.2-3 Distributed Secure Multi-Level DBMS With Static Encryption Architecture

secure multi-level DBMS system is the hardware ring which includes a switching network and multi-key encryptor/decryptor. By having each security level and set of compartments statically encrypted by a unique key and limiting the key (security level and compartment), mux/demux, (read/write) and channel (DBMS processor) combinations allowed, this hardware implements a trusted security filter. The ACS monitors arbitrates central data base access requests from the numerous compartmented front-end DBMS processors. Thus, this system consists of a central statically encrypted multi-level data base subsystem, a hardware security filter subsystem that provides the encryption and decryption, a software security monitor subsystem (ACS) that also arbitrates central data base access requests, and multiple front-end DBMS processor subsystems.

Functional System Specification

This system shall consist of a central multi-level data base subsystem, a hardware security filter subsystem, an access control subsystem and multiple front-end DBMS processor subsystems. The multi-level data base subsystem shall be statically encrypted with a unique key for each security level and security compartment. The hardware security filter subsystem shall encrypt all data written into the central data base and shall decrypt all data read from this data base. The hardware security filter subsystem shall allow the security officer to select which combinations of read/write, security level, and security compartment(s) will be allowed for each front-end DBMS processor subsystem. The hardware security filter subsystem shall enforce the selected access control policy for each DBMS subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their interconnections. The access control subsystem shall provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts. The access control subsystem shall be a system high, collection (receive only) node. This control data shall be internally audited by the hardware security filter subsystem for compliance with access control policy. The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator. The multiple front-end DBMS processor subsystems shall be physically independent and isolated from each other except for indirect connections provided

by the other subsystems. The DBMS processor subsystems will use a secure OS.

The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., throughput and access time degradation vs effective utilization shall be less than 3dB. The access control subsystem shall not degrade central data base throughput and shall not increase average access latency more than 100ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic).

As a design goal the DBMS subsystems shall minimize central data base utilization. As a design goal the system shall be fault tolerant with graceful degradation. As a design goal standard off-the-shelf hardware and software shall be used or modified where practical. As a design goal the system shall be capable of transparent on-line re-keying. As a design goal the system shall minimize the amount of certified and/or trusted software. As a design goal the central data base will have a 50% spare memory capacity. As a design goal the system shall be capable of supporting 40 central data bas accesses per second with a total throughput of 500 Kbytes/sec.

Functional Subsystem Specifications

The following subsystem specifications and design goals from Section 5.1 are applicable:

Subsystem Specifications	11 - 39 51 - 69
Design Goal Specifications	71 - 99

5.2.3 Back-End DBMS Architecture Without Encryption

Functional Description

The distributed architecture that implements this access control structure is shown in Figure 5.2-4. The key element that makes this network a secure multi-level DBMS system is the hardware ring which includes a switching network and exit guards. These exit guards control the security level and compartments, read/write and channel (DBMS processor) combinations allowed, thus implementing a trusted security filter. The ACS monitors and arbitrates central data base

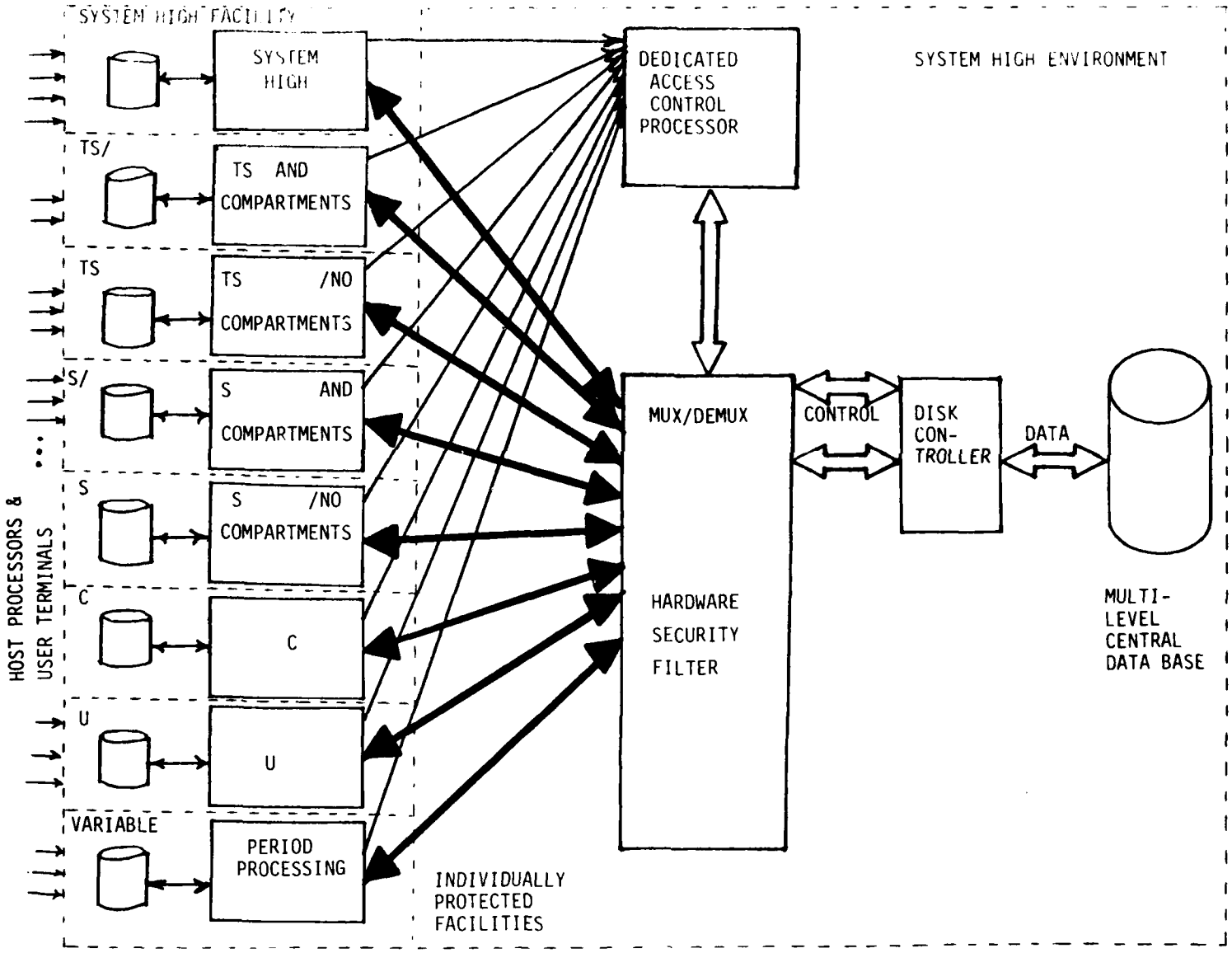


Figure 5.2-4 Distributed Secure Multi-Level Back-End DBMS Architecture

access requests from the numerous compartmented back-end DBMS processors. Each back-end DBMS processor may support one or more host processors as well as on-line interactive query user terminals. Thus, this system consists of a central multi-level data base subsystem, a hardware security filter subsystem that provides an exit guard for each DBMS processor subsystem, a software security monitor subsystem (ACS) that also arbitrates central data base access requests, and multiple front-end DBMS processor subsystems.

Functional System Specifications

This system shall consist of a central multi-level data base subsystem, a hardware security filter subsystem, an access control subsystem, multiple back-end DBMS processor subsystems, and numerous users. The multi-level data base subsystem shall provide logical data separation through use of a unique security tag for each combination of security level and security compartment. The hardware security filter subsystem shall tag all data written into the central data base and shall screen these security tags for all data read from this data base. The hardware security filter subsystem shall enforce the simple security rule and the *-security rule for each single level back-end DBMS processor subsystem. The hardware security filter subsystem shall enforce these security rules for each DBMS subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their interconnections. The access control subsystem shall provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts. The access control subsystem shall be a system high, collection (receive only) node except for control lines to the hardware security filter subsystems. This control data shall be internally audited by the hardware security filter subsystem for compliance with access control policy. The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator. The back-end DBMS processor subsystems shall be physically independent and almost isolated from each other except for indirect connections provided by the other subsystems. The DBMS processor subsystems will use a secure OS.

The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., additional throughput and access time

degradation vs. effective utilization shall be less than 3dB. The access control subsystem shall not degrade central data base throughput and shall not increase average access latency more than 100ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic). As a design goal, the tagging of data shall not degrade effective disk packing density more than 3dB.
As a design goal the DBMS subsystems shall minimize central data base utilization. As a design goal the system shall be fault tolerant with graceful degradation. As a design goal standard off-the-shelf hardware and software shall be used or modified where practical. As a design goal the system shall minimize the amount of certified and/or trusted software. As a design goal the central data base will have a 50% spare disk capacity. As a design goal the system shall be capable of supporting 20 central data base accesses per second with a total throughput of 250 Kbytes/sec.

Functional Subsystem Sepcifications

The following subsystem specifications and design goals from Section 5.1 are applicable:

Subsystem Sepcifications	11 - 49
Design Goal Specifications	71 - 99

5.2.4 Back-End DBMS Architecture With Static Encryption

The distributed architecture that implements this access control structure is shown in Figure 5.2-5. The key element that makes this network a secure multi-level DBMS system is the hardware ring which includes a switching network and multi-key encryptor/decryptor. By having each security level and set of compartments statically encrypted by a unique key and limiting the key (security level and compartment), mux/demux, (read/write) and channel (DBMS processor) combinations allowed, this hardware implements a trusted security filter. The ACS monitors and arbitrates central data base access requests from the numerous compartmented back-end processors. Each back-end DBMS processor may support one or more host processors as well as on-line interactive query user terminals. Thus,

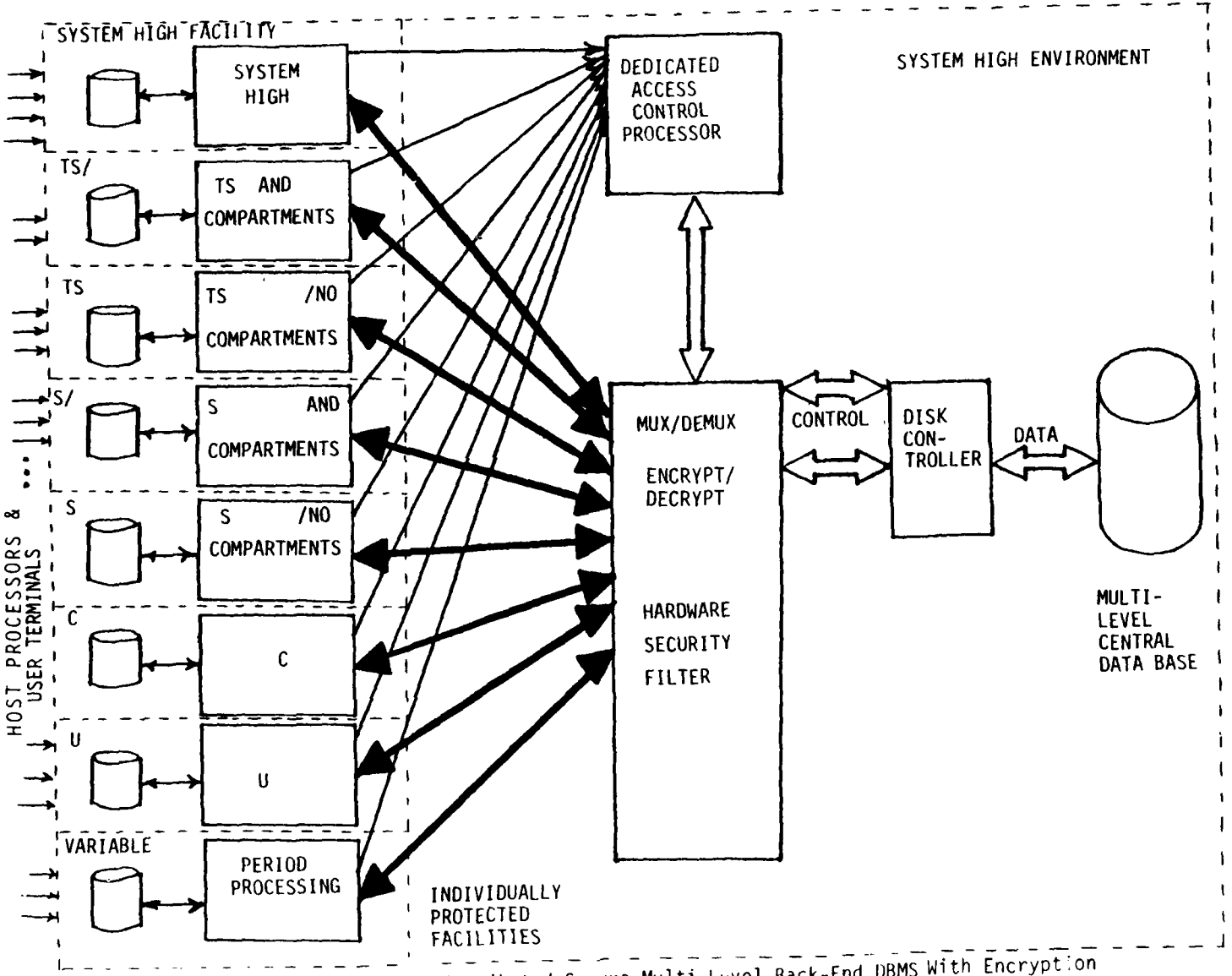


Figure 5.2- 5 Distributed Secure Multi-Level Back-End DBMS With Encryption Architecture

this system consists of a central statically encrypted multi-level data base subsystem, a hardware security filter subsystem that provides the encryption and decryption, a software security monitor subsystem (ACS) that also arbitrates central data base access requests, and multiple back-end DBMS processor subsystems.

Functional Specifications

This system shall consist of a central multi-level data base subsystem, a hardware security filter subsystem, an access control subsystem and multiple back-end DBMS processor subsystems. The multi-level data base subsystem shall be statically encrypted with a unique key for each security level and security compartment. The hardware security filter subsystem shall statically encrypt all data written into the central data base and shall statically decrypt all data read from this data base. The hardware security filter subsystem shall enforce the simple security rule and the *-security rule for each single level processor subsystem. The hardware security filter subsystem shall allow the security officer to select which combinations of read/write, security level, and security compartment(s) will be allowed for each back-end DBMS processor subsystem. The hardware security filter subsystem shall enforce the selected access control policy for each DBMS subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their inter-connections. The access control subsystem shall provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts. The access control subsystem shall be a system high, collection (receive only) node. This control data shall be internally audited by the hardware security filter subsystem for compliance with access control policy. The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator. The multiple back-end DBMS processor subsystems shall be physically independent and isolated from each other except for indirect connections provided by the other subsystems. The DBMS processor subsystems will use a secure OS.

The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., throughput and access time vs. effective

utilization shall be less than 3dB. The access control subsystem shall not degrade central data base throughput and shall not increase average access latency more than 100 ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic).

As a design goal the DBMS subsystems shall minimize central data base utilization. As a design goal the system shall be fault tolerant with graceful degradation. As a design goal standard off-the-shelf hardware and software shall be used or modified where practical. As a design goal the sytem shall be capable of transparent on-line re-keying. As a design goal the system shall minimize the amount of certified and/or trusted software. As a design goal the central data base will have a 50% spare memory capacity. As a design goal the system shall be capable of supporting 40 central data base accesses per second with a total throughput of 500 Kbytes/sec.

Functional Subsystem Specifications

The following subsystem specifications and design goals from Section 5.1 are applicable.

Subsystem Specifications	11 - 39 51 - 69
Design Goal Specifications	71 - 99

5.3 Alternate Approach Functional Description

This system can be described in terms of a protection ring structure consisting of a multi-level, multi-compartment statically encrypted central data base, a hardware security filter ring including a switching network, static encryptor/decryptor with multiple keys, and an end-to-end encryptor/decryptor with multiple keys, a software security monitoring and discretionary access control ring in the form of an access control subsystem (ACS) and a system high back-end DBMS processor with multiple single level front-end I/O processors, and numerous smart terminals and/or host computers with E^3 encryption as shown in Figure 5.3-1.

The security ring structure shown in Figure 5.3-1 is rather unique in that each ring and each of the outer compartments consist of separate hardware

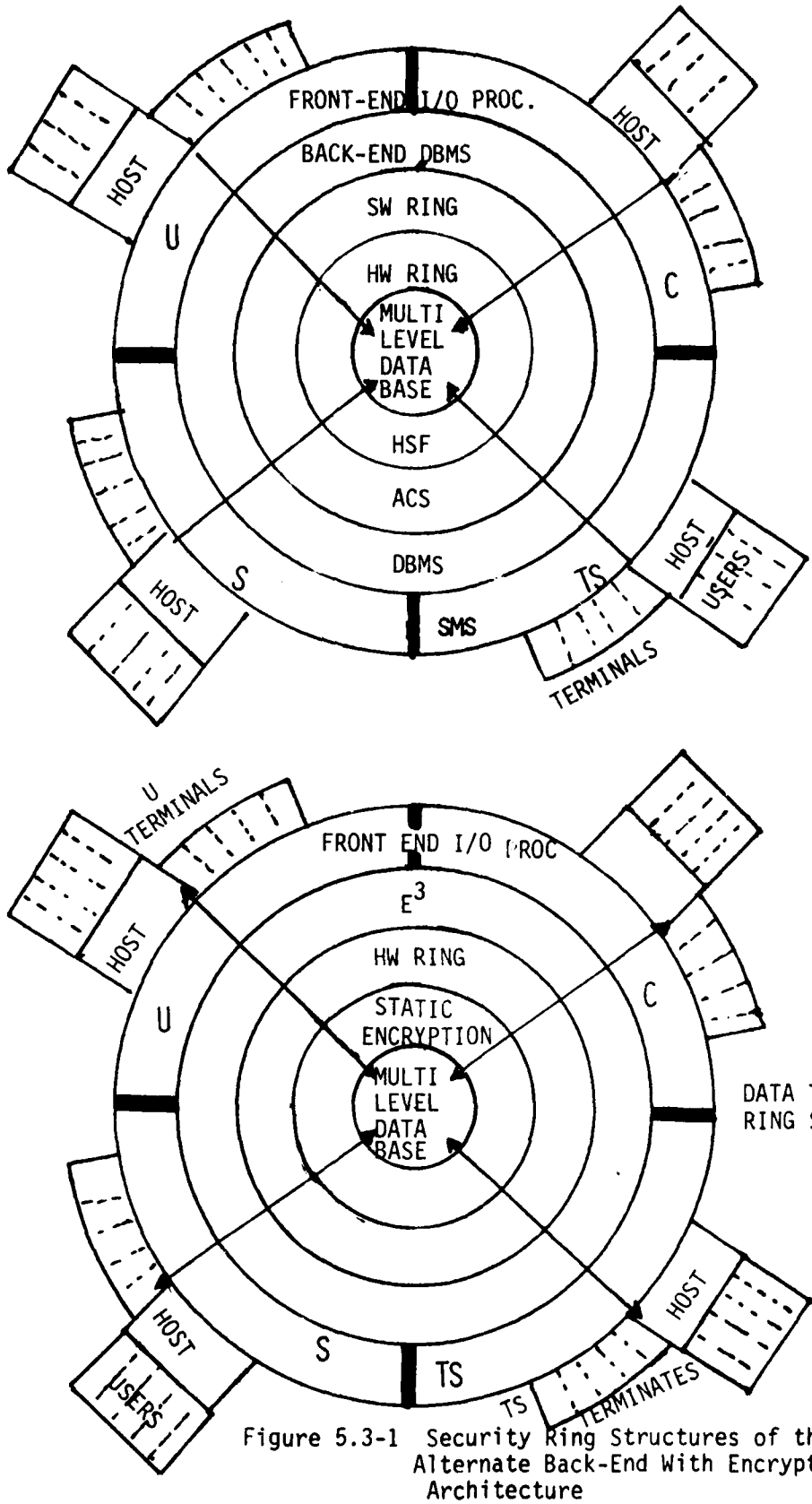


Figure 5.3-1 Security Ring Structures of the Alternate Back-End With Encryption Architecture

elements in a distributed architecture. Because of this physical separation, it is relatively easy to prove, by control and data flow analysis, that hard access control of each combination of read/write, security level and security compartment(s) can be enforced in hardware for each front end I/O processor. In addition, the use of an access control subsystem and end-to-end encryption will enforce compartment and discretionary security for users through crypto separation. Note that the control flow goes from the users through a system high back-end DBMS processor, the SW monitoring and the HW access control rings into the central multi-level data base, while the data flow is bi-directional through a signal hardware security filter ring and multiple single level front-end I/O processors with intermediate encryption interfaces.

Functional Description

The distributed architecture that implements this access control structure is shown in Figure 5.3-2. The key elements that make this network a secure multi-level DBMS system is the hardware ring which includes a switching network, static encryption, and dynamic end-to-end encryption (E^3). These elements are combined to act as exit guards controlling the security level and compartments, read/write, channel (DBMS processor) and E^3 key combinations allowed, thus implementing a trusted security filter. The ACS enforces user compartment and discretionary security controls through choice of E^3 key and monitors and arbitrates central data base access requests from the other processors. Thus, the system consists of a central multi-level data base subsystem, a hardware security filter subsystem that provides an exit guard for each processor and end user, a software access control subsystem that also monitors and arbitrates central data base access requests, a system high back-end DBMS processor, and possible host computers, and numerous smart terminals for each back-end DBMS processor.

In essence, this architecture places a security filter subsystem between numerous terminals and a central data base as shown in Figure 5.3-3 that is capable of enforcing access controls for each user with a common system high back-end DBMS processor generating data storage and retrieval locations. This approach only supports interactive data base queries, i.e., there are no user programs or code. Further, all DBMS generated data is placed in a user inaccessable

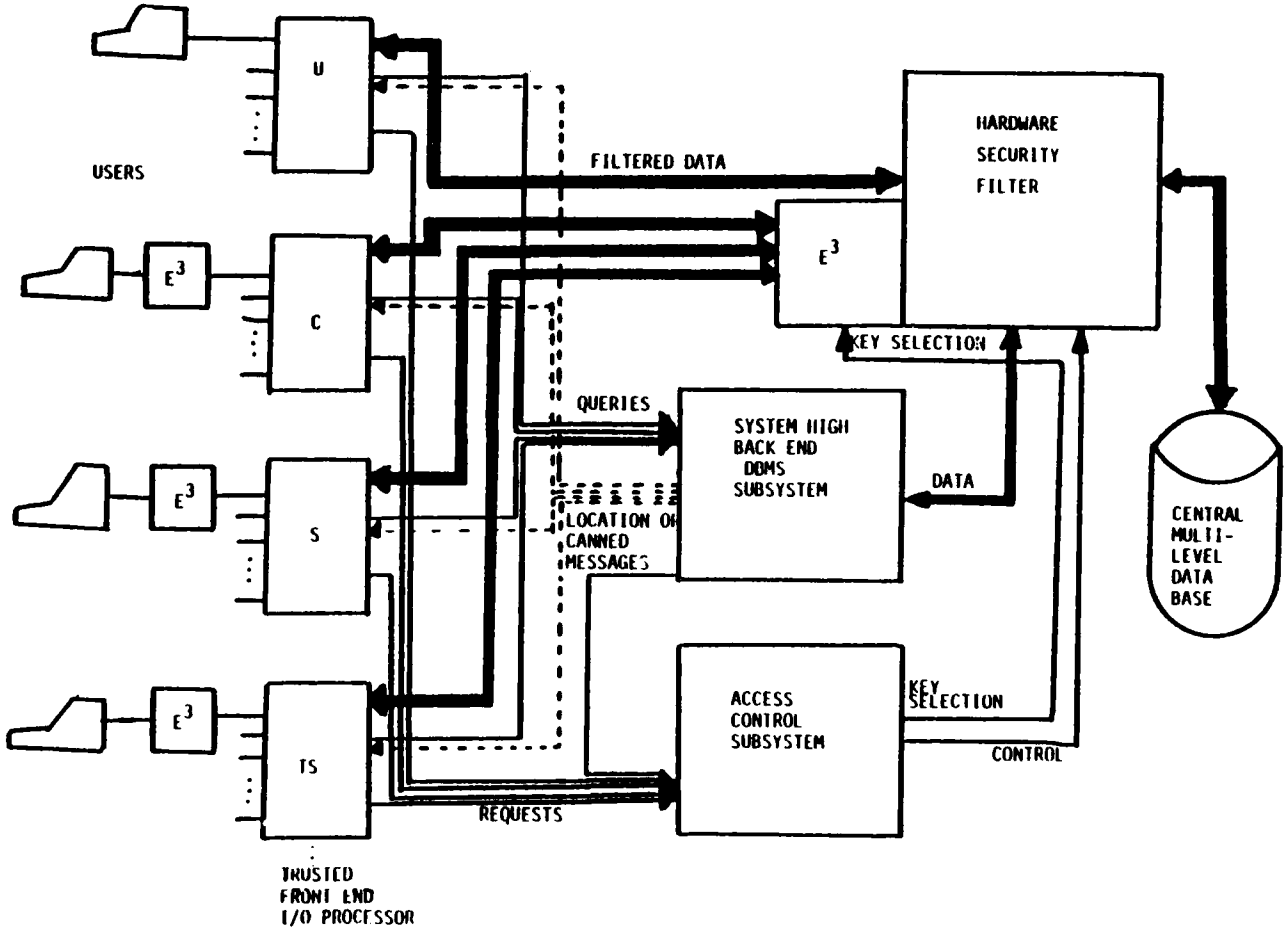


Figure 5.3-2 Alternate Secure DBMS Architecture Block Diagram

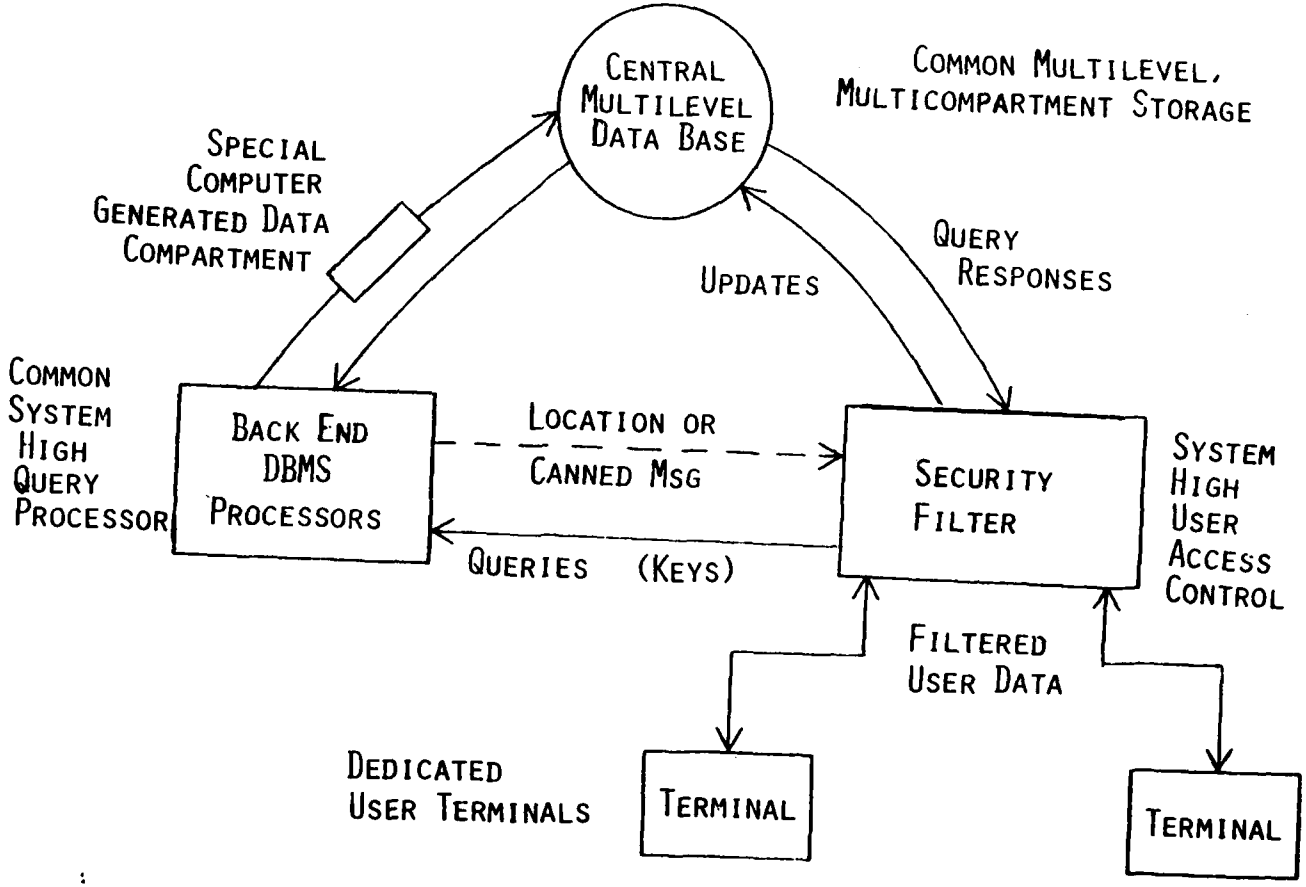


Figure 5.3-3 Alternate Architecture Concept

compartment and no direct I/O is allowed from the DBMS to the users. Instead, the DBMS supplies the address to where the user data or a canned message is to be retrieved from or stored to the access control subsystem. The security filter subsystems are responsible for determining if the user is allowed the associated read or write access. The net result is that all hardware, software, or firmware responsible for non-discretionary and discretionary security resides in the hardware security filter and access control subsystems. Hence, the back-end DBMS processor does not require a secure OS or any trusted software.

Functional Specifications

This system shall consist of a central multi-level data base subsystem, a hardware security filter subsystem, an access control subsystem, secure message switching subsystems, a system high DBMS processor subsystem, and numerous users. The multi-level data base subsystem shall be statically encrypted with a unique key for each security level and security compartment. The hardware security filter subsystem shall statically encrypt all data written into the central data base and shall statically decrypt all data read from this data base. The hardware security filter subsystem shall enforce the simple security rule and the *-security rule for each single level processor subsystem. The hardware security filter subsystem shall place all DBMS computer generated data in a special security compartment that shall be inaccessable to users. The hardware security filter subsystem shall enforce these security rules for each subsystem in a provable manner. Such a proof shall not require any restrictions on any other subsystem except their interconnections. The hardware security filter subsystem shall send and receive user data in an end-to-end encryption (E^3) form, with the user key selected by the access control subsystem so as to enforce compartment and discretionary security controls. The access control subsystem shall also provide arbitration of central data base access requests from the DBMS processor subsystems, central data directory operations required to support these requests, and security monitoring of these requests for violation attempts. The access control subsystem shall be a system high, collection (receive only) node except for control lines to the hardware security

filter subsystem, the end-to-end encryption system attached to the hardware security filter subsystem and the secure message switching processors. This control data shall be internally audited by the hardware security filter subsystem for compliance with subsystem access control policy. The access control subsystem node may also generate local reports for use by the security officer and/or data base administrator. The secure message switching subsystem shall consist of an independent single level message concentrator acting as a custom front-end I/O processor for the back-end DBMS processor. This subsystem shall handle routing of E^3 between users and the central data base and/or messages from the users to the DBMS processor and the access control subsystem. This subsystem shall ensure that there is no direct I/O from the DBMS processor to the user. Indirect I/O in the form of canned messages shall be supported. All data exchanged with the hardware security filter subsystem shall be encrypted user data. Smart terminals with E^3 capability shall support a DBMS query language with data base data sent and received in an E^3 form.

The hardware security filter subsystem shall not appreciably degrade central data base performance, i.e., additional throughput and access time degradation vs. effective utilization shall be less than 3dB. The access control and secure message switching subsystems shall not degrade central data base throughput and shall not increase average access latency more than 200ms over a 0% to 60% central data base utilization range (M/M/1 or M/D/1 traffic). As a design goal the DBMS subsystems shall minimize central data base utilization. As a design goal the system shall be fault tolerant with graceful degradation. As a design goal standard off-the-shelf hardware and software shall be used or modified where practical. As a design goal the system shall be capable of transparent on-line re-keying of the statically encrypted central data base. As a design goal the system shall minimize the amount of certified and/or trusted software. As a design goal certified and/or trusted software shall be confined to the access control subsystem. As a design goal the central data base will have a 50% spare disk capacity. As a design goal the system shall be capable of supporting 40 central data base accesses per second with a total throughput of 500 Kbytes/sec.

5.3.1 Secure Message Switch (Front End I/O Processor Subsystem Specification

These subsystems shall consist of multiple front-end I/O processors, certified firmware, and numerous serial communication links. Each processor will be run at a single level and will interface terminals to the other subsystems.

These processor systems shall provide a secure message switching function. These processors shall relay clear text messages from serial user links (terminals or hosts) to the access control subsystem or the DBMS subsystem as appropriate, and relay end-to-end encrypted data base entries between the users and hardware security system. Limited control inputs shall be accepted from the DBMS and Access Control Subsystems including sending canned messages to the terminal.

Log-on and directory messages shall be routed to the Access Control Subsystem. Query language command messages shall be routed to the DBMS and Access Control Subsystem Audit Trail. Canned messages from the DBMS subsystem shall be routed to the appropriate terminal and the Access Control Subsystem Audit Trail. Canned messages from the DBMS subsystem shall be routed to the appropriate terminal as directed by the Access Control Subsystem. Encrypted data from users will be stored until requested by the Access Control Subsystem.

This subsystem's code shall be certified that all messages sent to a terminal are either canned messages or formatted data received from the Hardware Security Filter Subsystem. All code shall be placed in ROM. Each processor will run at a single access level.

6.0 BENEFIT AND TRADE-OFF ANALYSIS

This section analyzes the benefits and tradeoffs in using a distributed architecture to implement a secure multilevel DBMS. Comparisons to standard approaches such as secure OS, system high, and period processing are made where appropriate. The section is organized in subsections as follows:

6.1 Security Analysis

6.2 Performance Analysis

6.3 Reliability and Maintenance Impact

6.4 Cost Analysis

6.5 Flexibility and Operational Impact

6.1 Security Analysis

This section presents the security analyses of the DBMS architecture proposed in this study. Three major areas are discussed:

Non-discretionary Security	Section 6.1.2
Discretionary Security	Section 6.1.3
Integrity Protection	Section 6.1.4

In addition, a summary of the protection processes is presented in Section 6.1.1

6.1.1 Overview

This architectural approach can easily solve the multi-level security problem without any trusted software or a secure OS if users can be restricted to single level DBMS processors. It also offers at least a partial hardware solution to compartment controls. Discretionary and integrity security controls require certified and/or trusted software. The use of static encryption restricts SFA hardware concerns to the Hardware Security Filter Subsystem (trusted hardware) and reduces the sensitivity of the Central Data Base Subsystem.

The primary advantage of this architecture is that users can be separated on independent DBMS processors with each processor having different limited access rights to a common data base predicated on security levels, security compartment(s) and read/write access mode. Because an access control policy (ACP) can be enforced

for each processor solely in hardware, certified/trusted software and/or a secure OS are not required to enforce non-discretionary security controls. Further, enforcement of these controls is easily provable by flow control analysis and is thus certifiable.

The drawbacks to using this architecture to provide hardware enforcement of non-discretionary security controls are the lack of downgrade capability and the potentially very large number of DBMS processors required to enforce compartment controls. Use of an additional DBMS processor with a system high ACP will add downgrade capability but only with a loss of hardware enforcement of all non-discretionary security for that processor.

The potentially large number of DBMS processors can be solved through a combination of approaches. Four single level DBMS processors with no compartment access are required to handle the non-trusted non-compartment users (levels U, C, S, TS). Batch production jobs requiring special compartmented access can be handled on a period processing basis with the ACP tailored for each period. The most difficult problem is servicing on-line compartment users. A partial solution is to provide additional processors at the S and TS levels with compartment access, thereby isolating the non-compartment users from the compartment users. This approach provides provable enforcement of security level controls (simple security rule and *-security rule) in hardware, allows non-trusted, non-compartment, on-line users to be allowed access to the data base when desirable (versus when required), and moves current multi-level operation problems down to the multi-compartment level.

The solution to this multi-compartment problem is solved through combinations of one or more of the following: 1) clear compartment users for all compartments; 2) period process compartment users, and 3) add additional DBMS processors for each compartment requiring concurrent access to the data base. From a security and functionality perspective, the best solution is the third approach. While the cost of such an approach is prohibitive today, it can be expected that the cost of an appropriate DBMS processor will drop below $40K within 5 years* In the interim, period processing of those DBMS processors requiring compartment accesses is the best approach to minimization of the number of processors required.

*32 bit microprocessor, 1Mbyte of memory, a 50 Mbyte Winchester disk, a 10 Mbd fiber optic link, standard I/O, and a relational DBMS.

Discretionary security will require a secure OS or equivalent technique, a user authentication methodology, and trusted software. These will be supported by the access control subsystem (ACS). This architecture requires that all user requests be "funnelled" through the ACS. This architecture thus has the advantage that the trusted discretionary access controls and audit trail software can be removed from the user accessable DBMS processors and placed in the user inaccessable ACS. If certification of this trusted software is desired, the access control subsystem can be implemented in a distributed architecture that eliminates the need for a concurrent multi-processing environment (operating system) by placing each task in a separate microprocessor. By restricting the software complexity per microprocessor task, the certification task is facilitated.

The methodology for enforcement of integrity protection depends upon the granularity of that protection. If integrity is handled at the file level, it can be handled as a part of the discretionary protection mechanism. If integrity is handled at the data item level, it must be enforced by trusted DBMS software.

Static encryption can be used by this architecture to provide data separation by clearance (level and compartment) and possibly by owner. The primary advantage of using static encryption is that faults in the central data base subsystem will not compromise security; that is, reading from and/or writing to the wrong disk location cannot compromise data. A secondary advantage is that the central data base stores black (encrypted) rather than red (clear) data and is thus less sensitive.

6.1.2 Non-Discretionary Security

Appropriate choice of processor access control policy (ACP) by the security officer will allow the hardware security filter (HSF) to provide certified enforcement of non-discretionary security controls for single level processors. By providing appropriate physical access controls, the HSF will enforce non-discretionary security controls for non-trusted users. The primary non-discretionary security threat comes from any multi-level processor, such a system high processor used for downgrading. The trusted users and software used on such downgrade processors should be limited to the degree feasible. Limited trust can be enforced for application programs which produce lower level summaries (products) from higher level data by period processing such runs on a batch machine with the ACP tailored to the application program.

The primary non-discretionary security check for multi-level machines is the centralization of all data base access requests through the ACS which provides an audit trail and security monitor function. Because the audit trail and security monitor reside in a dedicated processor isolated from potentially hostile user code, it is possible to certify or at least trust their operation. The security monitor can notify the security officer of all downgrading (write access) requests and even delay execution of the downgrade process until explicitly approved by the security officer.

The HSF can be designed to enforce ACPs selected by the security officer independent of ACS access requests. That is, the ACS does not require any certified or trusted code in order for the HSF to enforce non-discretionary security for single level processors. The HSF can also be designed to be fail safe (for single point failure) from a non-discretionary security perspective. Such a design requires static encryption of the central data base so that any data incorrectly acquired by the central data base subsystem is subsequently incorrectly decrypted.

6.1.3 Discretionary Security

Discretionary security controls require the identification and logical separation of users through certified or trusted software/firmware enforcement as opposed to the physical access controls and certified hardware employed to enforce non-discretionary security. Logical separation of users can be: entrusted to a standard commercial OS; certified with a secure OS; enforced with end-to-end encryption; or achieved with individual (personal) processors.

The logical place for the actual discretionary access control software and the associated data base containing access permissions is in the ACS. This centralization of discretionary access control isolates it from potentially hostile user code, and provides good physical protection through its location in a system high compartment with access primarily limited to the security officer and potentially the data base administrator.

The largest discretionary security problem is user authentication. The clear text passwords traditionally used for user identification is the weak link in most computer security systems. Smart terminals with encryption have been proposed to strengthen this link. In one approach users carry individual keys on their person that are used for identification. Physical protection of these keys

is the major disadvantage of this approach. A more sophisticated approach prevents impersonation (via playback techniques) by using an encrypted link. In this approach, a secure dedicated key distribution center (KDC) generates session keys and stores protected clear text passwords. When the user desires to initiate a session he signals the authentication processor to get a new session key. He then enters his password. The password is encrypted in the session key at his terminal and is sent to the KDC. The KDC decrypts the password and compares it to the stored password for validation. Because the password is encrypted between the terminal and the KDC in a session key, it is secure. Without such a hard user authentication technique, the use of more than a commercial OS for user separation is unwarranted.

If certified discretionary security is required then commercial OSs cannot be trusted. The traditional approach to DoD computer security is development of a secure OS. While technically possible, past attempts at developing a secure OS have not been certifiable in practice and have proven to be very expensive.

A technically more feasible approach to a secure DBMS is to use end-to-end encryption to enforce user separation at the terminal level, use a system high back-end DBMS to create non-user accessable access paths and locate central data base locations, use certified single level front-end I/O (secure message switch) processors to route data directly between the HSF and terminals, and use certified firmware in the ACS to enforce discretionary securing through the choice of E^3 keys at the HSF interface. (This is the architecture described in Section 5.5. While technically feasible, such an approach is likely to be as expensive as a secure OS and even more limited. Further this approach limits users to interactive DBMS queries with no application processing capability. To add such application processing capability requires the addition of a host processor between the user's terminal and the E^3 subsystem. If a host processor is shared, the need for user separation immediately reappears. As a result, this solution is limited to interactive terminal queries with no processing support.

The most secure approach is to isolate users on individual processor systems. Discretionary access control can then be enforced by a relatively small amount of certified software in the ACS.

6.1.4 Integrity Security

Integrity protection is a dual of security protection. Integrity attempts to protect high quality data from being corrupted by being directly mixed or replaced with low quality data. The placement of integrity security is sensitive to the granularity required. If file level granularity is chosen it can be implemented in the ACS in a manner similar to that used for discretionary security. If lower level granularity is chosen, it is recommended that it be incorporated into a custom DBMS. However, integrity protection can possibly be implemented by upgrading the HSF and modifying the ACS.

6.2 Performance Analysis

The following sections discuss the expected performance for a secure DBMS in terms of system throughput, data base efficiency trade-offs, and processor capabilities. The following subsections are presented:

6.2.1 System Model

6.2.2 Detailed Traffic Model

6.2.3 M/M/m Queue Performance

6.2.4 System Performance Analysis Conclusions

6.2.5 Relational Data Base Performance Characteristics

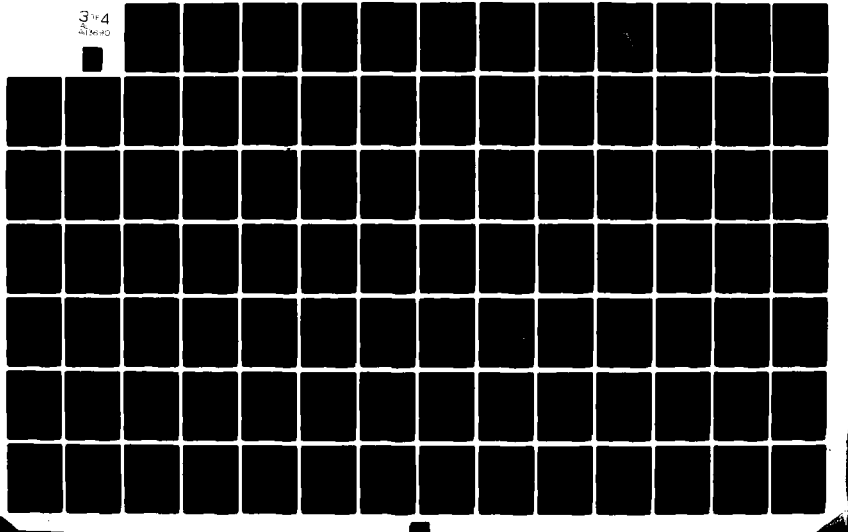
6.2.6 Processor/Microprocessor Performance Characteristics

The system performance analysis is based upon a system traffic model. The analysis is performed by applying M/M/m queueing theory to the model. The discussions presented here in Sections 6.2.1 through 6.2.3 are a summary of the analysis. More details of the model and queueing theory used is provided in Appendix D for reference purposes.

6.2.1 System Model

For the purposes of performance analyses, a general M/M/m queueing network can be used for a system model, as shown in Figure 6.2-1. (For details see Appendix D.) This approximation reduces the complexity of the analysis and does not need additional unknown parameters that would be required for a more detailed model.

AD-A113 690 HARRIS CORP MELBOURNE FL GOVERNMENT ELECTRONIC SYSTE--ETC F/G 9/2
SECURE DBMS.(U)
FEB 82 T D WORMINGTON, C E GIESLER F30602-80-C-0235
UNCLASSIFIED RADC-TR-81-394 NL



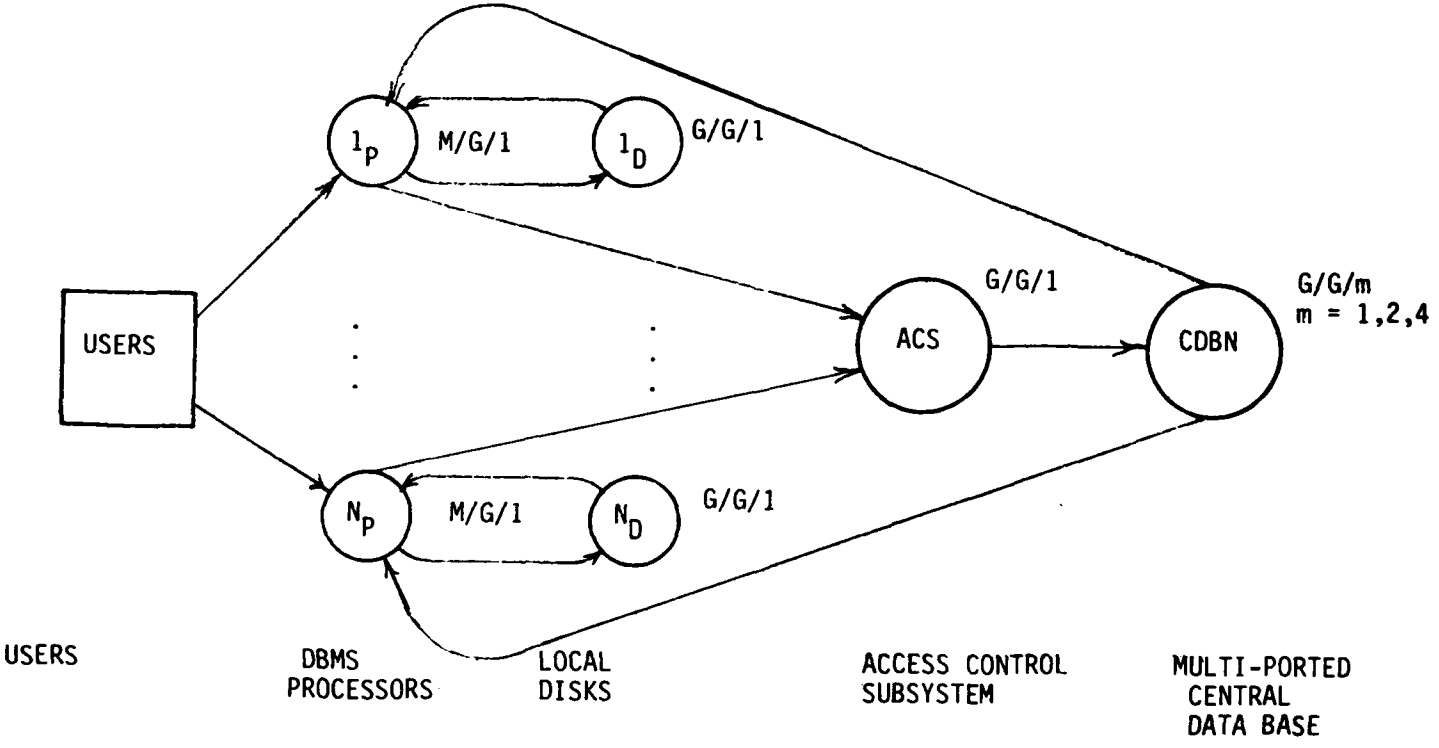


Figure 6.2-1 System Queueing Model

The primary metric for the secure DBMS is the average system response time, τ_s. τ_s is a function of the subsystem loads, subsystem capacities, and average subsystem service times. For M/M/m queues these parameters can be reduced to two types: average subsystem service times and subsystem utilization. This metric is used here to compare use of virtual memory and non-virtual memory DBMS techniques with the performance of a single DBMS processor with a single disk channel.

6.2.2 Detailed Traffic Model

This section summarizes the traffic model used in the secure DBMS performance analysis. Details and further discussion of this model are presented in Appendix D.

The benchmark traffic model was chosen to be an on-line user making a query of a two relation view with the query containing the key to the first relation. The key for the second relation is provided as a foreign key by the record requested in the first relation. This benchmark query is converted to a physical traffic model by specifying the sequence of processor tasks and disk accesses, as shown in Figure 6.2-2. This model ignores I/O and OS overhead and assumes multiprocessing as opposed to true timesharing. Note that disk access 6 is not a part of the system response timeline but does increase the disk subsystem loading, and consequently indirectly increases system response time.

Figure 6.2-3 summarizes the traffic model parameters chosen for evaluation in this analysis. The sequence of disk accesses corresponds to that of a VM system. For a non-VM system, disk accesses 2 and 4 of Figure 6.2-2 are assumed, optimistically, not to be required. These pointer tables are assumed to be resident when the user's program is rolled in. However, although fewer disk accesses are assumed to be required, the model assumes that both the VM and non-VM systems require the same amount of DBMS processing. Basically, this model assumes that the users generate a query every 20 seconds on the average and that a new relation is accessed once every 30 queries. Disk response is assumed to require an average of 2.5 revolutions at 16.6 ms/rev.

Two major assumptions are made for this model:

1. The DBMS systems are I/O bound; and
2. The access control node is very fast relative to the disk performance.

Event	Task #	Disk #	Description
1		Disk Access 1	Upon arrival of a user query, the user is rolled into the DBMS processor.
2	Processor Task 1		The query is processed and the key to the first relation is hash coded to determine the pointer table index.
3		Disk Access 2	The partially inverted pointer table is read from disk (if not already resident).
4	Processor Task 2		The pointer is used to compute the disk address of the record.
5		Disk Access 3	The record is read.
6	Processor Task 3		The foreign key is removed from the record and is hash coded to determine the pointer table index.
7		Disk Access 4	This pointer table is read in from disk (if not already resident).
8	Processor Task 4		The pointer is used to compute the disk address of this record.
9		Disk Access 5	The second record is read.
10	Processor Task 5		The view is constructed and the query response is formatted.
11		Disk Access 6	The user is rolled out.

Figure 6.2-2 Detailed Data Access Process Model

MODEL PARAMETER	SYMBOL	UNITS	VM SYSTEM	NON-VM SYSTEM
Query Rate	λ	Queries/Sec/User	.05	.05
New Relation Rate	j	Relations/Query	1/30	-
Mean Disk Service Time	$1/\mu_D$	MS	40	40
Mean DBMS Processor Service Time	$1/\mu$	MS	12	20
Mean Access Control Node Service Time	$1/\mu$	MS	5	5
DBMS Processor Load	λ_P	Queries/Sec/Processor	0.25 K/N	0.15 K/N
Local Disk Load	λ_L	Queries/Sec/Processor	0.3 K/N	-
Central Data Band and Access Control Node Load	λ_C	Queries/Sec/Processor	0.01 K	0.2 K

K = Number of Users
N = Number of DBMS Processors

Figure 6.2-3 Summary of Traffic Model Parameters

These assumptions are formulated as

$$1/\mu_{NVM} = \frac{1}{2}(1/\mu_D) \text{ and } 1/\mu_{ACN} = 1/8(1/\mu_D)$$

and are reflected in the values of Figure 6.2-3.

6.2.3 M/M/m Queue Performance

The average normalized system response time for an M/M/m queue with infinite population can be analytically solved in terms of its utilization, i.e. percent of capacity loading. This in turn allows a maximum subsystem degradation limit to be converted into a subsystem loading limit. For our analysis of subsystem response time, the system degradation specification of 3 dB (Section 5.1) is used as the normal operating point. For a heavy load model, a 6 dB response time degradation is used.

Figure 6.2 - 4 presents the service time metric as a function of system utilization for several queue statistical models (M/M/1, M/D/1 etc.). The curves describe the system utilization factor for the desired 3 dB and 6 dB degradation of service response time for the DBMS. Figure 6.2 - 5 summarizes these results, showing fraction of system utilization (ρ) resulting from the specified degradation figures. (In addition, this figures also shows the normalized capacity of the system (λ_η) (for $m > 1$), as the capacity ofan m server system is m times the capacity of a single (m=1) server system.)

6.2.4 System Performance Analysis Conclusions

Based on an interactive single query (with key) benchmark and a maximum subsystem degradation of 3 dB the Virtual Memory (VM) Systems are clearly superior to the corresponding non-VM Systems in terms of total and local DBMS query capacity, in terms of minimum and average query response times, and in terms of potential central data base throughput. These conclusions are reached from the queue analyses for the benchmark model selected in Section 6.2.1 and shown graphically in Figure 6.2 -6 through 6.2 -8 It should be recalled that this model only predicts the performance for a maximum likelihood user query function. Other types of query processing would likely show less differences between VM and Non-VM systems although VM performance will still be superior.

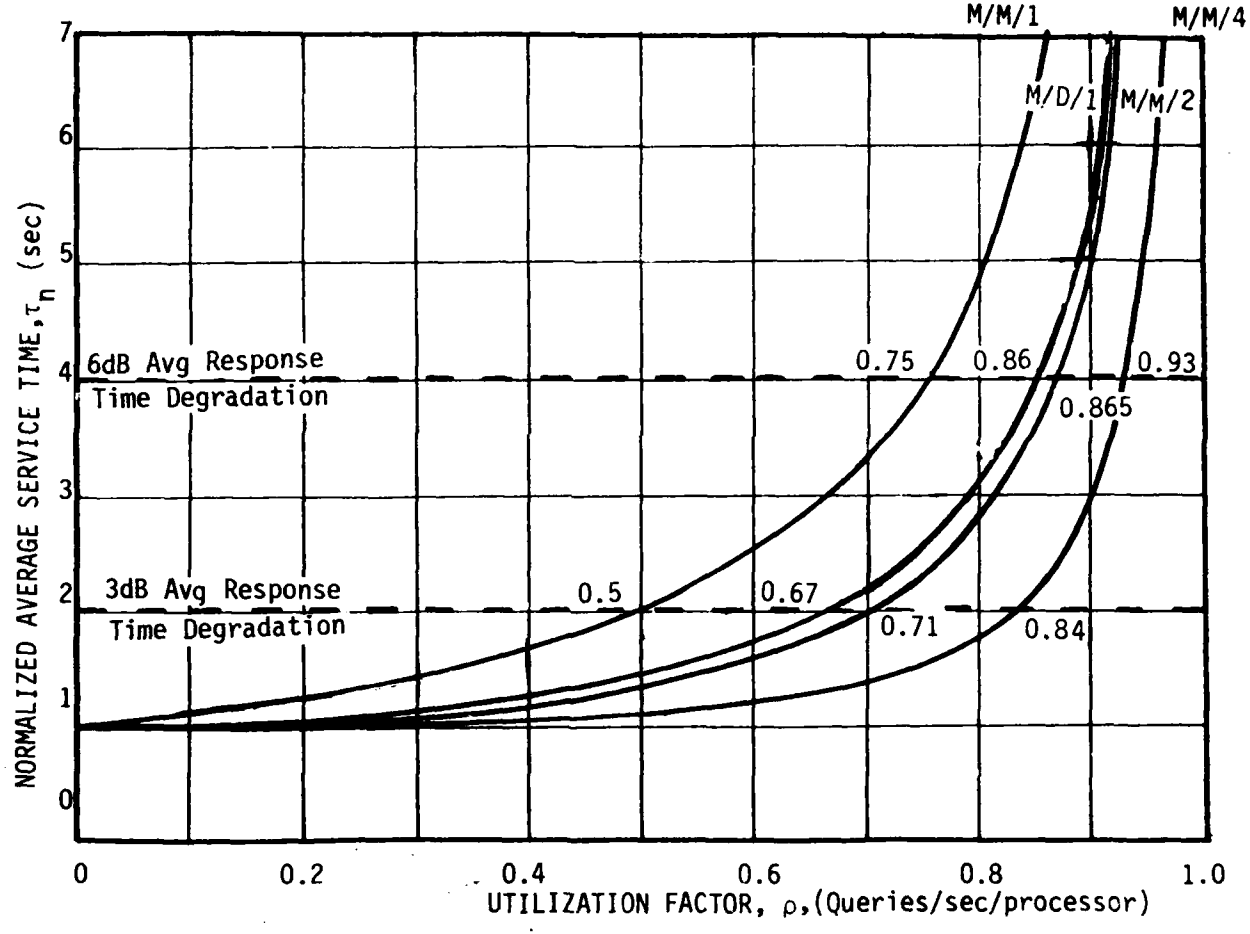


Figure 6.2-4 Average Service Time

QUEUE SYSTEM	PARAMETER	AVERAGE SYSTEM RESPONSE DEGRADATION	
		3 dB	6 dB
M/M/1	ρ	0.5	0.75
	λ_η	0.5	0.75
M/D/1	ρ	0.67	0.86
	λ_η	0.67	0.86
M/M/2	ρ	0.71	0.865
	λ_η	1.42	1.73
M/M/4	ρ	0.84	0.93
	λ_η	3.36	3.72

ρ = Utilization

λ_η = Normalized # of users supported by system assuming CM = MC_1

Figure 6.2-5 Summary of System Utilization for 3 dB and 6 dB System Response Degradation

The performance curves of Figures 6.2 -6 through 8 present the following information.

Figure 6.2 -6. VM System Performance

This curve shows the system response time for two VM systems: single and 8-processor DBMS. It should be noted that there is virtually no difference between the single and multiple processor configurations in terms of overall system response.

Figure 6.2 -7. Single Channel Non-VM System Performance

This curve shows that the system utilization factor declines relatively rapidly as more DBMS processors are added to the Non-VM system. This is due to saturation of the disk service process. The key conclusion here, is that the VM system permits additional DBMS processors to be added to the system with little or no system degradation. This figure also illustrates the inherent additional system latency of the non-VM system -0.4 sec versus 0.25 sec for VM systems at a system load of zero.

Figure 6.2 -8. Dual Channel Non-VM System Performance

This figure illustrates the system performance improvement achieved for the Non-VM system when the system is configured in dual channels when compared to the previous plot. The VM system with a single DBMS processor is also shown for reference purposes.

The two primary problems with the use of a central data base for multiple DBMS processors is dilution of disk throughput and the increase in access latency. The use of VM technology can largely offset these two disadvantages providing performance close to that of a single DBMS processor with a single disk channel. With VM technology the performance impact of a shared central data base is evident only when opening a file or closing it following an udpate session.

With a non-VM DBMS processor approach, each disk access associated with a query suffers from an additional latency of approximately 100ms due to delay in sending a request message to the Access Control Subsystem. This more than doubles effective disk access time and approximately doubles minimum query response time relative to a single DBMS processor with a single disk channel (performance reference). In addition, since this central resource is shared, the total query capacity of the system is limited by the central data base throughput. Further, the query capacity

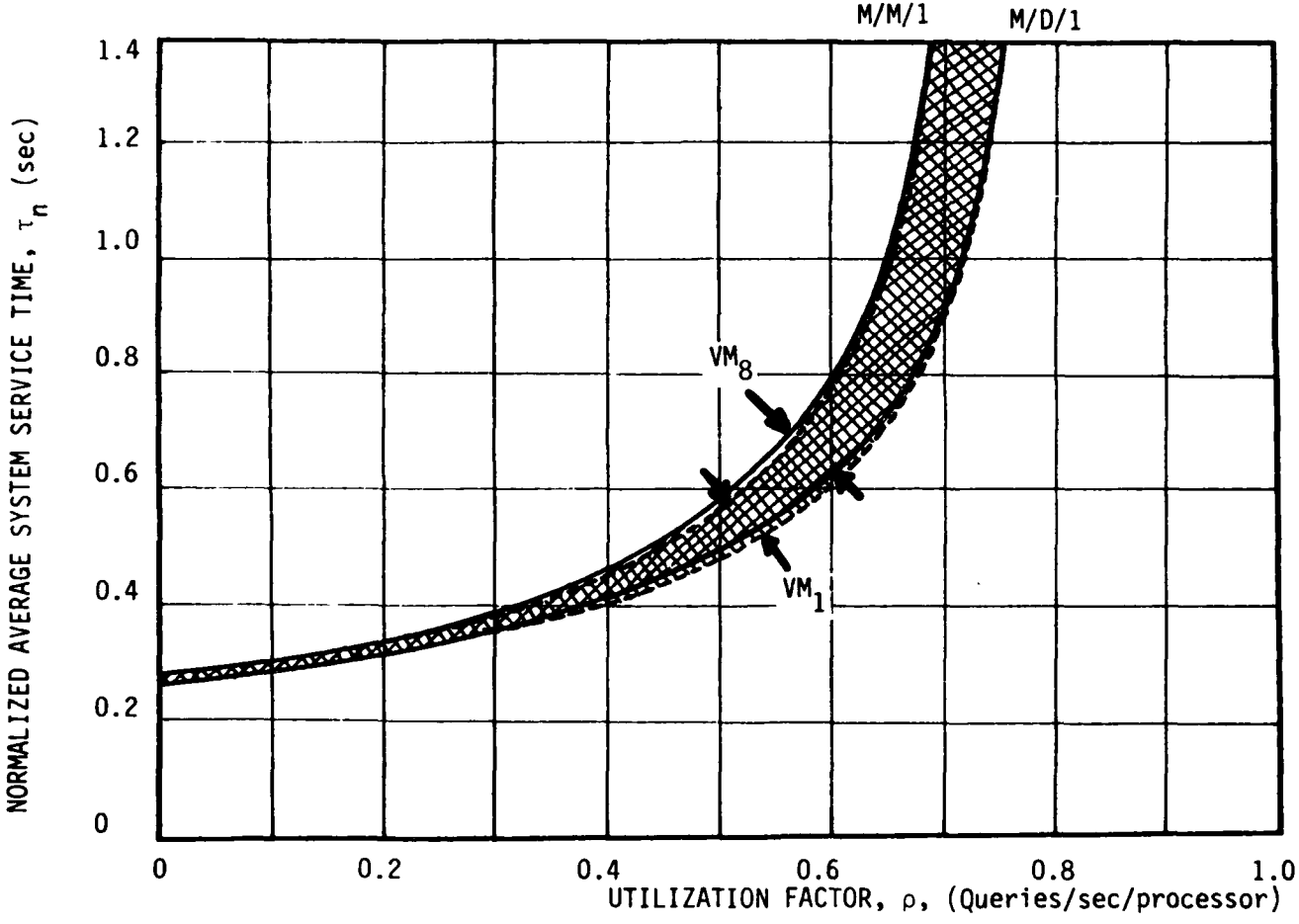


Figure 6.2-6 Average VM System Responses

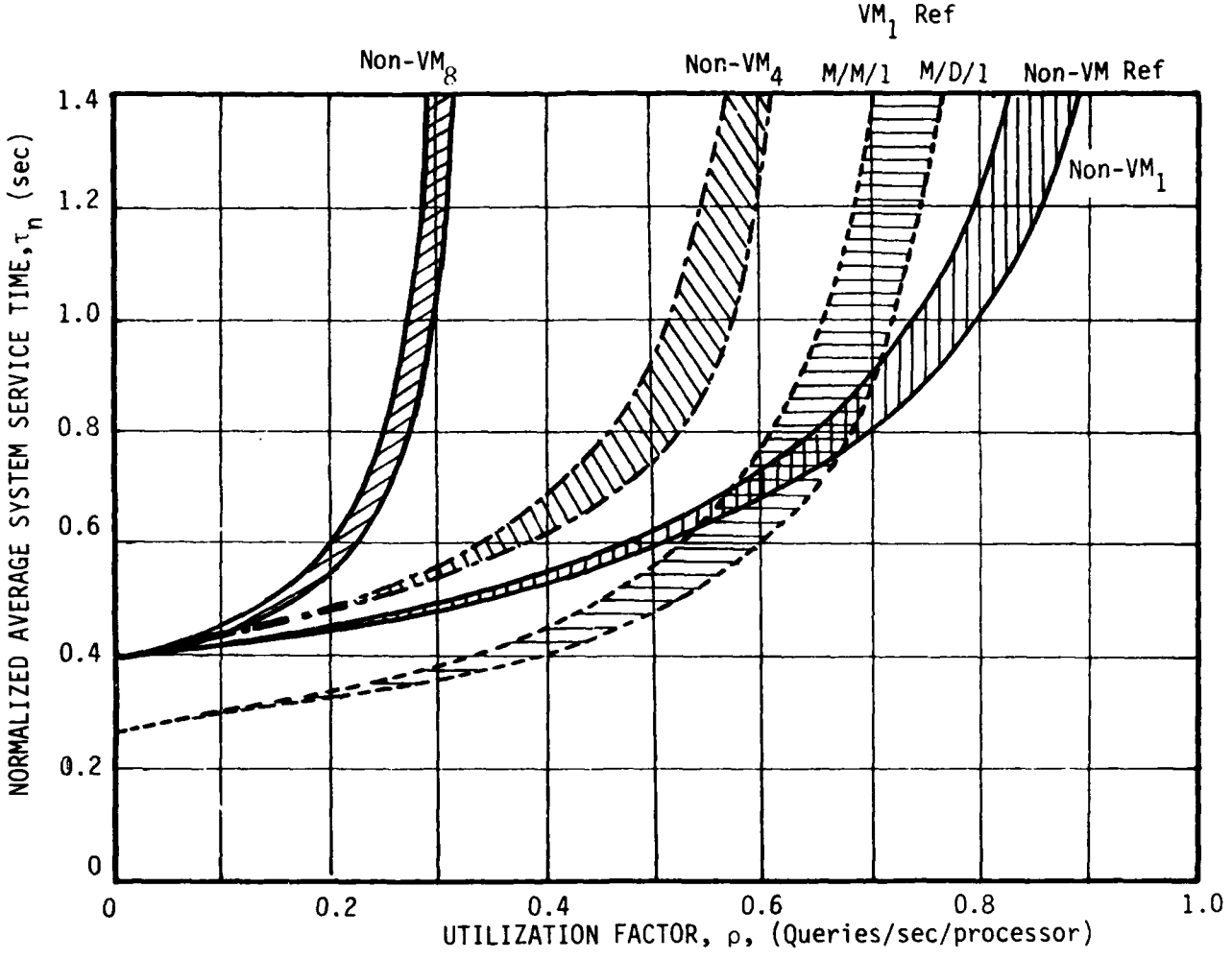


Figure 6.2-7 Average Non-VM System Response (Single Channel)

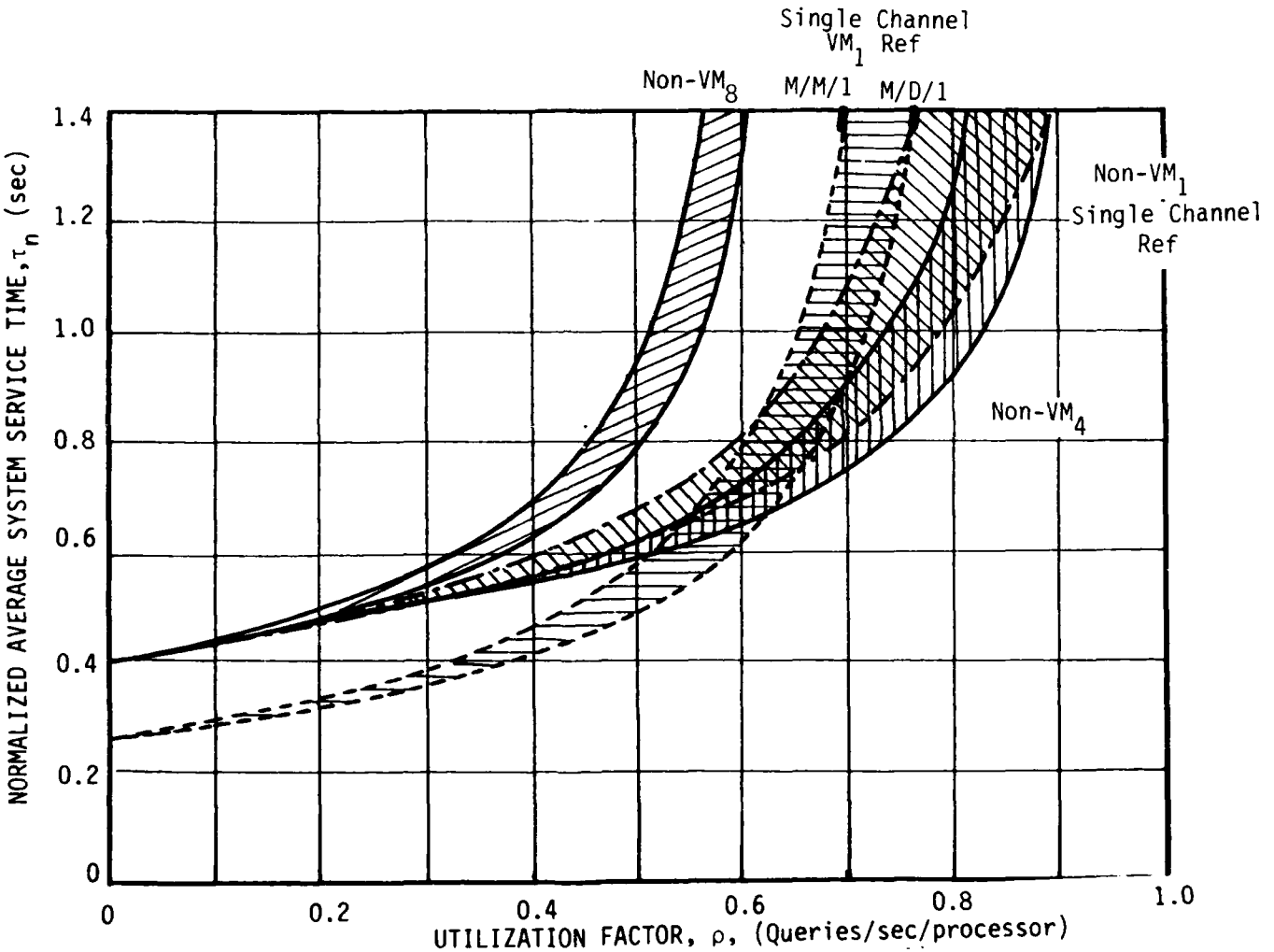


Figure 6.2-8 Average Dual-Channel Non-VM System Response

per DBMS processor is then inversely proportional to the number of processors and can only be increased by adding additional central data base disk channels, preferrably with dual ported disks.

A VM DBMS processor approach suffers no additional latency per query relative to a single DBMS processor with a single disk channel (performance reference) as its disk accesses are also local. Furthermore since the central data base is utilized only when opening or closing a file, the demand on the central data base resource is much lower than for a non-VM approach. Consequently, the query capacity per DBMS processor is limited by the local disk query capacity and total system query capacity will be greater than that of the non-VM system capacity. Further, the local VM disk capacity can easily be doubled with standard dual ported disk technology. A secondary benefit of the VM approach is that central data base throughput potential (Kbytes/sec) will be greater than for the non-VM approach. This is because the VM accesses will tend to be larger serial accesses while the non-VM accesses will tend to be shorter random access accesses.

It should be remembered that these performance conclusions are based upon a maximum likelihood type of benchmark and does not represent actual system performance. In particular, the small percentage of queries that involve multiple responses and/or do not use a relation's keys typically use the majority of a relational DBMS's resources. This type of processing is more likely to be processor bound than I/O bound and normally requires sequential access to all records of a relation. For this type of query the differences between a VM and non-VM approach will be less pronounced although VM performance will still be superior. The key conclusion is that a VM approach will allow additional DBMS processors to be added to the system with little loss to individual DBMS performance, while a non-VM approach will dilute individual DBMS performance unless the other subsystems are improved to compensate.

6.2.5 Relational DBMS Performance Characteristics

The compromise between performance and flexibility in the implementation of DBMS systems has generally been made to achieve rapid standard query access at the expense of flexibility. In contrast, flexibility is the key feature of a relational DBMS with the result that a relational DBMS requires more computer resources to respond to a query than other commonly used DBMS approaches. With a classical network approach only two or three random disk accesses are required for a query. For a standard query, a hash code is used to indicate the record in a partially inverted pointer structure to be scanned for the pointer to data requested. If secondary data is required the primary data record will have direct pointers to any secondary data records. With a relational DBMS approach, sequential scanning of all records and indirect linkages generated on the fly are often required. For a relational query, a sequential search of data records of a first relation may take place. From a selected record, a foreign key is extracted and is hash coded to index a partially inverted pointer structure. This pointer structure is subsequently scanned for the pointer for an associated second relation's data record. The information from all records thus selected is combined in a defined fashion view and compared to Boolean specifications of the query. Conversely, relational DBMS updates are easier to implement as fewer linkages need to be created or modified during the process.

In conclusion, a relational DBMS needs a large memory and a powerful processor to be efficient. The use of VM is advantageous for relational DBMS implementation because VM eliminates the overhead of multi-buffer management for each random data base access. Additionally, as potential enhancement to data base processing, newer processors are providing record addressing modes and string and queue manipulation instruction sets which are well tailored to DBMS support.

6.2.6 Processor/Microprocessor Performance Characteristics

As the Secure DBMS architectures described in this report replace a single large DBMS processor with multiple smaller DBMS processors, it is of interest to compare the performance of a range of mini and micro processors. Figure 6.2-9 indicates a typical FORTRAN benchmark for various sizes of processors. In particular, it shows the sensitivity to implementation, e.g., interpretive HOL versus assembly, and hardware support, e.g., with or without floating point coprocessor.

Super 32 bit Minicomputer (fp)	1000
Standard 16 bit Minicomputer (fp)	400
32 bit Microprocessor (fp)	200
16 bit Microprocessor (fp)	100
8 bit Microprocessor (assembly with floating point)	50
8 bit Microprocessor (assembly with out fp)	5
8 bit Microprocessor (interpretive higher order language-HOL without fp)	½

Figure 6.2 - 9 Whetstone* Comparison of Processors

* The term "WHETSTONE" refers to a composite machine level instruction used in comparing CPU speeds between various machines when executing typical higher level language statements. The term is used within the context of the "WHETSTONE BENCHMARK" program. The "WHETSTONE" program attempts to provide a true benchmark of higher level language usage by assigning relatively higher "weights" to those instruction types most frequently used in actual scientific programming applications. The program was developed by two British researchers, B.A. Wichmann and J.J. Curnow, based upon Wichmann's statistical analysis of 949 ALGOL 60 programs. The benchmark simulates the actual frequencies of "WHETSTONE INSTRUCTIONS" as recorded in Wichmann's statistical profile. The Whetstone Benchmark has now become the industry wide standard for measuring performance capabilities of various machines and/or compilers in executing higher level languages such as FORTRAN, ALGOL, AND PL/I. Speed is expressed in terms of thousands of "WHETSTONE INSTRUCTIONS" per second. One Whetstone instruction is approximately equal to two machine instructions.

Although this is not a good DBMS benchmark, it is indicative of relative performance differences. Hardware support consisting of large physical memories, VM, record addressing, string manipulation instructions, and queue manipulation instructions all offer potential DBMS performance improvements.

6.3 Reliability and Maintenance Impact

Reducing such as found in this architecture generally improves both reliability and maintenance, and is particularly beneficial in critical applications where fault tolerant, fail safe operation is required. Depending on the degree of redundancy and feasible recovery options one of three reliability results occurs:

- o If an ACS and HSF is placed between a DBMS processor and its data base, this increased quantity of hardware will obviously degrade the individual system reliability.
- o If a spare DBMS processor is available, or if higher priority users can preempt lower priority users from another DBMS, or if it is operationally acceptable to share another DBMS processor on a degraded period usage basis, then the reliability of the individual DBMS is effectively replaced with the composite reliability of the ACS and HSF plus an adjustment for the recovery time following a DBMS subsystem failure.
- o If a redundant ACS and HSF is supplied in a self-checking, hot standby or fault tolerant configuration, then there is essentially no reliability degradation introduced by those subsystems. Finally, if the central data base is configured in a parallel dual-ported manner and a "spare" DBMS is available then extremely high system availability and possible fault tolerant operation is achievable.

Maintaining a larger number of lower cost DBMS processor systems should be easier than maintaining an equivalent single DBMS processor system due to lower sparing cost. This advantage is largely offset by the additional requirements for maintaining the data base ACS and HSF. A common DBMS and ACS computer would obviously help this situation considerably.

6.3.1 Reliability Model

A system model of such a DBMS architecture in a partially redundant form is shown in Figure 6.3-1. The corresponding reliability model for this system is shown in Figure 6.3-2 consisting of four types of components designated

A,B,C, and D as described in Figure 6.3-3. As components B and C are the two potentially serial elements, they are the primary candidates for replication if enhanced availability is required.

Based on a 50% spare central data base storage requirement, 2 of every 3 drives (2/3M) must be available for the system to be available. Following the loss of any drive, a degraded mode exists while the files contained on that disk are reloaded. Unless a fault tolerant design keeps (writes) redundant files on separate drives, this reloading operation can require as much as an hour. Furthermore, this recovery operation requires the use of a DBMS processor (D component) during this period.

If both B components are down, the system as shown is down. If only one B component is down, the system is available in a degraded mode, unless the B component fails during a rekeying operation interval. If both B components are required for rekeying, then loss of a single B component during a rekeying interval can result in unavailability of the system until both B components are restored

If C is down the system is down.

Each D component supports a separate set of users. An operational policy in which higher priority users will preempt lower priority users is now assumed. Under these rules at least N-i of the N D-components must be available to support the i^{th} class (priority) of users (i = 0,1,...,N-1). Furthermore, when a particular D component fails, the system is unavailable to its users while they replace lower priority users on their processors and recover operation. This movement to a new DBMS processor of higher priority users is at least equivalent to a security level change of a period processed machine. If the system has a spare DBMS processor, there are not 0^{th} class users as described above.

In addition to the DBMS components described above, power and air conditioning are also required for operation. Also, if static encryption is employed, data base rekeying is required periodically. We therefore define three more reliability components:

P for power source
AC for air conditioning
RK for rekeying

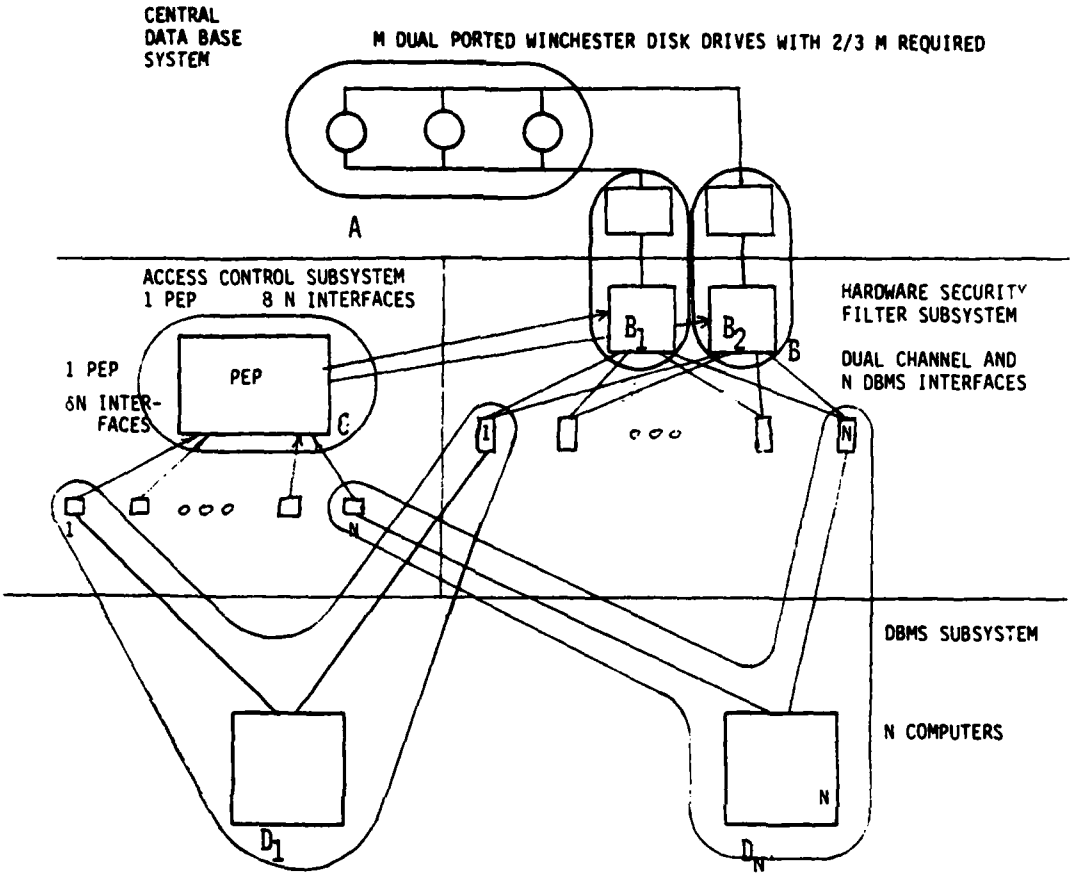


Figure 6.3-1 Subsystem Model

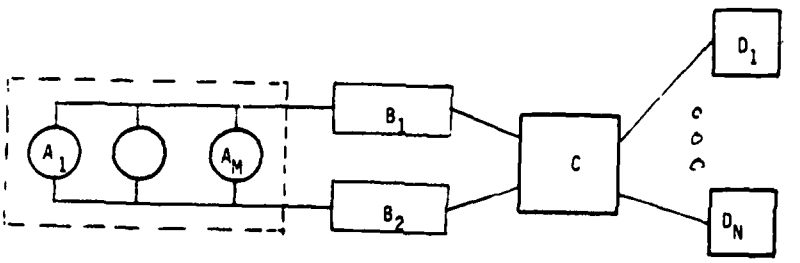


Figure 6.3-2 Reliability Model

COMPONENT DESIGNATOR	QUANTITY SYSTEM	DESCRIPTION
A	M	INDIVIDUAL DISK DRIVES IN CENTRAL DATA BASE
B	I	COMPOSITE OF DISK CONTROLLER IN CENTRAL DATA BASE, CORRESPONDING HSF CHANNEL (EXCLUSIVE OF INDIVIDUAL DBMS INTERFACES), AND STATIC KG SYSTEM (IF USED)
C	1	ACS (EXCLUSIVE OF INDIVIDUAL DBMS INTERFACES)
D	N	INDIVIDUAL DBMS SUBSYSTEM, THEIR COMMUNICATION LINES AND THEIR INTERFACES TO THE ACS AND HSF

Figure 6.3-3 Principal DBMS Reliability Model Components

For each reliability component, there is a mean time between failure (MTBF) and a mean time to restore (MTTR) designated as $MTBF_i$ and $MTTR_i$ where i = A,B,C,D,P,AC, or RK. By assuming that the MTBF for a component is much greater than its MTTR, the resulting equation for the system reliability model can be written, as presented in Figures 6.3-4 and 6.3-5. To simplify the form of the equations, we define the

$$\frac{MTTR_i}{MTBF_i} \equiv \alpha_i \qquad \text{(Eq 6-1)}$$

From Equations 6-2 and 6-3, the probability that the system is available to the i^{th} priority class user is approximated as

P(Total System Available to i^{th} Class User) $\approx$ 1-P(Degraded Operation) - P(System Not Available to i^{th} Class User) (Eq. 6-4)

To give an indication of actual reliability impacts, the availability of three system configurations was determined using Equation 6-4. Figure 6.3-6 presents the assumed parameters chosen for this sample evaluation. The resulting availabilities are presented in three figures for the systems considered:

- o An Independent System (single DBMS and Disk, with no ACS or HSF) - a single system reference — Figure 6.3-7
- o A non-redundant distributed system — Figure 6.3-8
- o A fault tolerant system — Figure 6.3-9

The results show that in the system approach described by Figure 6.3-8, the non-redundant system gives only a slight availability improvement over the reference system while the redundant system yields nearly an order of magnitude improvement. Figure 6.3-9 shows that addition of an ACS and HSF for fault tolerance provides somewhat lower system availabilities than did the user prioritized approach. Again it should be noted that these figures are only representative for the example system formulated for illustration. Obviously, actual system implementation and component choice greatly impacts the availability of a system.

6.3.2 Maintenance

Maintenance is multi-faceted, involving fault detection, fault isolation recovery, and repair operations. In addition to normal hardware maintenance,

$$P(\text{Degraded Operation}) \; M \; \alpha'_A + \left\{2\,\alpha_B\right\} \qquad \text{(Eq. 6-2)}$$

term id 1 2

where {·} indicates {·} an open term

Equation Description

Term 1: file reloading interval after a disk failure

Term 2: loss of single HSF channel or disk controller in a redundant, fault-tolerant configuration.

Figure 6.3-4 Evaluation of Probability of Degraded Operation

P(System Not Available to i^{th} Priority Class User) =

$$\alpha_p + \alpha_{AC} + \sum_{j=M/3+1}^{M} \binom{M}{j} (\alpha_A)^j + \left\{\alpha_B \text{ or } (\alpha_B)^2\right\}$$

Term id 1 2 3 4a 4b

$$+ \left\{2(\alpha_{RK})^2\, \alpha_B\right\} + \left\{\alpha_C \text{ or } (\alpha_C)^2 + \alpha'_C\right\}$$

Term id 4c 5a 5b 5c

$$+ \left\{\alpha_D \text{ or } \sum_{K=i+1}^{N} \binom{N}{K} (\alpha_D)^K + \alpha'_D\right\} \qquad \text{(Eq 6-3)}$$

Term id 6a 6b 6c

where {·} indicates · an optional term or

Figure 6.3 -5 Evaluation of Probability of System Availability to i^{th} Priority Class User

EQUATION DESCRIPTION

TERM 1	LOSS OF NON-REDUNDANT POWER SOURCE
TERM 2	LOSS OF NON-REDUNDANT AIR CONDITIONING SOURCE
TERM 3	LOSS OF MORE THAN 50% DISK SPARE CAPACITY
TERM 4a	LOSS OF THE HSF SUBSYSTEM OR DISK CONTROLLER (CENTRAL DATA BASE CHANNEL)
TERM 4b	LOSS OF BOTH CENTRAL DATA BASE CHANNELS IN A FAULT TOLERANT CONFIGURATION
TERM 4c	LOSS OF A SINGLE TERMPORARILY CRITICAL CENTRAL DATA BASE CHANNEL DURING REKEYING OF A FAULT TOLERANT STATICALLY ENCRYPTED SYSTEM.
TERM 5a	LOSS OF THE ACCESS CONTROL SUBSYSTEM
TERM 5b	LOSS OF BOTH ACS IN AN ACS STANDBY CONFIGURATION
TERM 5c	RECOVERY TIME REQUIRED TO SWITCH TO A STANDBY ACS WHILE THE PRIMARY ACS IS DOWN (A VERY SMALL TERM FOR A FAULT TOLERANT HOT STANDBY DESIGN)
TERM 6a	LOSS OF DBMS SUBSYSTEM OR ASSOCIATED COMMUNICATIONS LINKS
TERM 6b	LOSS OF i+1 OR MORE DBMS SUBSYSTEMS OR ASSOCIATED COMMUNICATION LINKS, THUS DENYING ACCESS TO THE i^{th} LEVEL USER
TERM 6c	RECOVERY PERIOD WHILE PRE-EMPTING LOWER PRIORITY USERS

Figure 6.3-5 (cont'd) Evaluation of Probability of System Availability to i^{th} Priority Class User

COMPONENT	MTBF (HRS) CASE I	CASE II	MTTR (HRS)
A	1000	10,000	1
B	500	5,000	1
C	1000	1,000	2
D	500	500	4 ($\frac{1}{2}$ hr. for $MTTR_D'$)

Number of individual disk drives M = 8

Number of individual DBMS processors, links and interfaces N = 0

Power and Air Conditioning and Rekeying Factors Were Not Used In The Sample Evaluation

Figure 6.3 -6 Parameters for Example System Availability Calculations

	CASE I	CASE II
DISK SYSTEM	0.996	0.9996
DBMS	0.992	0.9992

Figure 6.3-7 Component Availability

	INDEPENDENT	NON REDUNDANT	REDUNDANT	CLASS PRIORITY
CASE I	0.992	0.94 0.994 0.996	0.94 0.997 0.999	0 1 2 7
CASE II	0.9992	0.99 0.997 0.997	0.994 0.9999 0.9999	0 1 2 7

Figure 6.3-8 System Availability by Priority (Including Degraded Modes)

	INDEPENDENT SYSTEM	NON REDUNDANT SYSTEM	REDUNDANT SYSTEM	CLASS PRIORITY
CASE I	0.99	0.94 0.99 0.992	0.93 0.99 0.993	0 1 2-7
CASE II	0.999	0.99 0.997 0.997	0.992 0.998 0.998	0 1 2-7

Figure 6.3-9 Total System Availability (Non-Degraded Mode) By Priority

DBMS maintenance, which is primarily software related, is also needed to perform data base recovery operations. Such maintenance operations are typically built into DBMS systems. However, they may require modification to work in a distributed architecture, especially since full duplex interprocessor communication between single level machines is prohibited by the non-discretionary security policy.

Maintenance of the HSF should be very easy due to the self-checking required to satisfy SFA fail-safe requirements. Little additional cost is required to design in on-line fault detection and automatic fault isolation (AFI). Use of signature analysis* techniques should allow easy design of AFI with better than 95% confidence at the LRU level.

Maintaining a larger number of lower cost DBMS processor systems should result in appreciable less cost for complete sparing than for an equivalent single DBMS processor system. However, the impact on the skill level of maintenance personnel is not as clear. Many recent minicomputer systems have been designed for improved maintainability by offering near automatic fault detection and isolation capabilities. The most advanced offer a separate maintenance processor for performing diagnostics with an optional link to remote diagnostic centers. It is questionable to assume that lower cost DBMS processors will offer such trouble shooting capabilities. Thus, either a higher skill level or increased sparing with depot level repairs is likely to be needed. In either case, the mean time to repair will obviously be less critical for this redundant (distributed) architecture.

The largest maintenance problem is likely to be the ACS. The use of a standby ACS is recommended to avoid ACS maintenance impacting system availability. In addition, utility software to back-up and reload ACS data bases will be required. If the current ACS directory is lost, the entire central data base will have to be reloaded. To avoid this problem, it is recommended that the ACS operational software maintain duplicate copies of its data base on separate drives with the ability to perform from a single copy and later restore the other.

Distributed DBMS maintenance is largely a state of the art problem. At a minimum, a write interlock is required to keep two processors from making

*TM - HP

simultaneous incompatable changes to the data base. However, the simple security rule prohibits a lower level machine (which may make blind updates) from observing such a semaphore set by a higher level source (a potential covert channel). This can be handled by locating such semaphores in the system high ACS and thereby delaying a second write request to any file already engaged in the write mode. This solution provides a low speed covert timing channel. The speed of this channel can be further degraded through techniques suggested in Section 2.4 such as the delaying of the initial write of a write access to a file from a processor by a random length of time.

Distributed data base recovery is a greater challenge. Normally the entire data base is periodically backed-up and subsequent changes logged on magnetic tape. When a file or disk is lost, the most recent version of those files is reloaded and a utility package scans the change log to recreate the subsequent changes. An equivalent approach would require the system high DBMS (assuming that one is provided) to back-up the data base periodically and receive changes from all processors for update logging. A more recent approach that meets on-line DBMS maintenance requirements is to maintain duplicate copies of files on separate disks with the ability to operate from one and later reload the other. Ignoring the cost impact, this technique is preferrable because it can be performed transparently by the ACS(except for a small utility program to perform the actual reload)without modifications to the DBMS.

6.4 Cost Analysis

Cost can be considered from several perspectives. In particular, development costs, recurring costs, life cycle costs, direct costs, and indirect costs are all of interest. This distributed approach to secure multi-level DBMSs offers the potential for much lower development costs than that of a secure OS approach. Besides the lower direct development costs, the potential to use standard commercial software in a secure multi-level environment offers the possibility of even greater indirect savings in terms of other software development. While the secure OS approach has no direct recurring cost it is limited to a particular processor. In contrast to the cost of a new secure OS development effort required for a new processor, little cost is required to adapt this distributed architecture to

a new processor. Considering total life cycle costs, this secure multi-level DBMS architecture should recover its costs from the reduction of the quantity of high level clearances previously required for system high applications, increased availability over that previously available through period processing and improvements in mission efficiency by allowing personnel access when desirable rather than where required.

6.4.1 Development Costs

A wide range of development costs are possible depending on the particular system desired.

The lowest development cost is for a certified HSF with single-level file protection granularity and an ACS with trusted code that is transparent to a commercial processor, operating system, and DBMS. This system provides certified hardware enforcement of non-discretionary security but relies on the commercial OS for any discretionary security protection.

The highest development cost is for a back-end system with end-to-end encryption, certified single-level front-end (message switching) processors, certified distributed ACS, certified HSF with multi-level file support and static encryption, and a custom DBMS. This configuration provides external certified firmware enforcement of discretionary security in addition to non-discretionary security controls. The development cost of such a system would easily rival the development of a secure OS. The primary advantage of this back-end configuration is their certification is more technically feasible than for a secure OS.

Figure 6.4 -1 ranks development costs that might be found in particular system developments.

In addition to these possible direct development costs, TEMPEST, SFA, reliability, maintenance, fault tolerance, expandability, adaptability and other factors will have indirect development costs. Given the high cost of certification, it is recommended that a modular HSF system with standard options be developed that can be easily configured to meet a range of requirements. In particular, consideration to adaptability to different processors, modular expansion, and high reliability/fault tolerance is recommended. Such preplanned product im-

provements can be added to the ACS as needed if the ACS merely contains trusted code. If certified ACS code is deemed desirable, the initial design and preplanned upgrade support for the ACS will require increased care.

High:	Secure OS Custom DBMS Certified ACS End-to-End Encryption Certified Multi-level File HSF Certified Single Level File HSF Static Encryption Trusted ACS
Low:	Transparent I/O Handler
None:	Commercial DBMS Commercial OS Commercial Processor

Figure 6.4.1-1 Development Cost Ranking

6.4.2 Recurring Cost Estimate

It is estimated that the additional direct recurring acquisition cost required to provide a secure multi-level DBMS is less than 25% of total system acquisition cost excluding any additional costs required for physical security, Tempest, or KGs (which are Government owned equipment). The savings in operational cost over a 10 year life cycle will easily dominate this increase of initial acquisition cost. This estimate assumes that the DBMS processor subsystems and central data base subsystem or equivalent are required by the application with or without a secure multi-level DBMS requirement and that the direct additional cost for this capability is the recurring cost of the access control subsystem (ACS) and the hardware security filter (HSF) subsystem hardware. Figure 6.4-2 details the subsystem costs used in this estimate and Figure 6.4-3 describes the equipment assumed for each subsystem to arrive at these costs. From these figures, the total system cost for a four DBMS processor system would be around $1.4M of which less than $300K (21%) is directly related to multi-level security. For an eight DBMS processor system these numbers would be $2.5M and $300K (12%) respectively.

The dominant recurring cost of the secure multi-level DBMS is for the DBMS processor subsystems and is obviously very sensitive to the number of

SUBSYSTEM	UNIT COST $K	SYSTEM COST $K	
		4-DBMS CONFIGURATION	8-DBMS CONFIGURATION
DBMS PROCESSOR	250	1000	2000
CENTRAL DATA BASE	100	100	100
ACS	100	100	100
HSF*	200	200	200
TOTAL COST $K	--	1400	2400
% COST IN MULTI-LEVEL DBMS SECURITY	--	21%	12%

* cost is only a preliminary engineering estimate

Figure 6.4-2 Cost Estimates for a Multi-Level DBMS

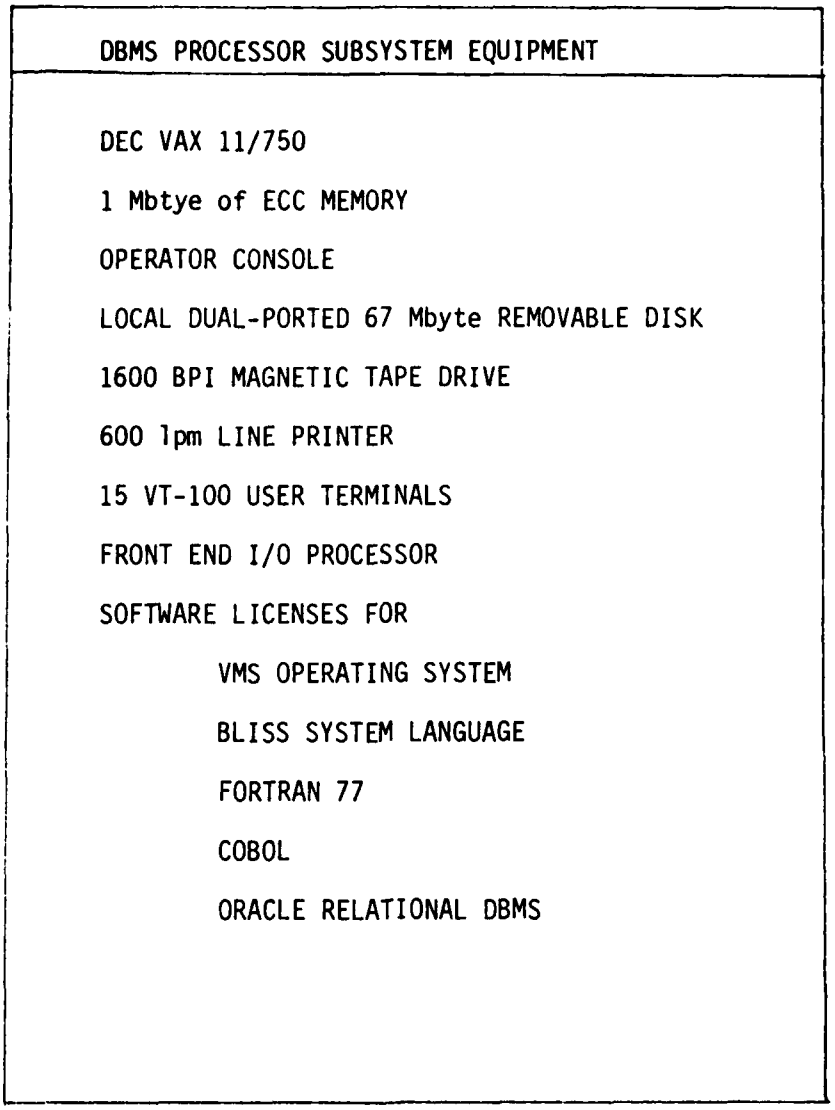


Figure 6.4 -3 Equipment Used to Estimate Subsystem Costs

CENTRAL DATA BASE SUBSYSTEM EQUIPMENT
Two Disk Controllers
Four Dual-Ported 124 Mbyte Winchester Disk Drives

ACS EQUIPMENT
PDP 11-44
256 Kbytes of Memory
Operator Console
Single Ported 67 Mbyte Disk Drive
1600 BPI Tape Drive
Front-End I/O Processor
RS 232-C Compatible Direct Wire Modems
Custom Hardware Security Filter Subsystem Interfaces

HSF EQUIPMENT
Custom Hardware
Single-Level File Support (File protection granularity)

Figure 6.4 -3 Cont'd Equipment Used to Estimate Subsystem Costs

such processors required and their cost. If an ACS/HSF security filter can be built so as to be adaptable and transparent to the operations of the DBMS processors, then a variety of DBMS processors can be considered for any given application, thereby permitting a trade-off between acquisition costs and performance. Considering technological developments (see Section 7), it is estimated that an 8-fold reduction in suitable DBMS subsystem price (to $40K each in constant dollars) is likely in the next five years. The ability of this architecture to exploit decreasing hardware costs and make multi-level security an add-on to new computer systems with no appreciable development costs are the key features of the recurring costs of this design.

6.5 Flexibility and Operational Impact

The largest operational impact of providing multi-level security will be the introduction of secure multi-level DBMS operation. If appropriately designed, the ACS/HSF with its hardware enforcement of non-discretionary security will be a transparent add-on to standard hardware which is compatible with standard software. Non-discretionary security rules will impose some inherent limitation on multi-level data base structures (schema restrictions). In addition, multi-level data entry and updates from a single-level processor may require custom DBMS and HSF support to be feasible.

Certified firmware enforcement of non-discretionary security will have large operational impacts and limitations unless personal DBMS processors are used. Rekeying of a central data base subsystem and maintenance of the data base itself can either be performed off-line or on-line, in a transparent manner. A fault tolerant design (as described in Section 6.3.1) can meet critical on-line reliability requirements.

There is a need for a system high processor and trusted subjects to perform downgrading operations. There may be a need to augment a standard relational DBMS with a write interlock enforcable in the ACS to handle multiprocessor data contention. If back-end DBMS machines are used, the user interface will be essentially limited to a query language and any user processing will require a separate host processor.

Providing a secure multi-level DBMS alleviates the requirement of clearing all personnel to system high, and thus indirectly induces operating costs in terms of fewer high-level clearances. In addition, the use of a secure multi-level DBMS increases functionality for many applications where it is beneficial but not necessary to allow lower security level personnel access to some low level portions of the data base concurrently with data base accesses by higher level personnel. Period processing may be used in conjunction with this architecture to reduce the number of concurrent compartments required.

7.0 TECHNOLOGY ACCESSMENT AND FORECAST

Within five years technical developments should allow a suitable DBMS processor system consisting of a high performance 32-bit microprocessor, 1 Mbyte of ECC memory, a local 50 Mbyte Winchester disk drive, high speed fiber optic central data base link, multiple relational DBMSs, and RS-232 links, to be configured with commercial, off-the-shelf hardware and software for as low as $40K. This development will make use of an HSF architecture economically feasible for multi-level/large multi-compartment applications.

The following is a detailed accessment of current applicable technology and a prediction of future developments over the next five years. The following topics are covered:

7.1 Processors/Microprocessors
7.2 Software
7.3 Computer Communications
7.4 Mass Storage
7.5 Memory

7.1 Processors/Microprocessors

The rate of major advances in this area is continuing at a rapid pace, allowing further significant decreases in cost and performance improvements to be expected in the next five years. There has been an explosion of super 32 bit minicomputers with previous mainframe performance (above 1000 Whetstone*) at low system cost (in the $150K to $500K range) in recent years. A similar explosion of 32 bit microprocessors now appears imminent. Intel, HP, Bell Labs, and Nippon have already produced 32-bit microprocessors (using VLSI technology) with the revolutionary Intel iAPX432 design advancing the state-of-the-art of computer systems. In addition, virtual memory (VM) technology is now available in top of the line minicomputers and microprocessors and is fast becoming standard. Equally important, instruction sets are becoming more powerful and specialized. Advanced string, record, and queue instructions in particular are well tailored to DBMS operation and offer large improvements in DBMS efficiency.

* Section 6.2.6 and Figure 6.2-9 discuss the Whetstone as a measure of system capability.

Better protection mechanisms are also appearing. Perhaps the most innovative are the object orientation approach (e.g., iAPX-286,432) and public double key password encryption. Another development of interest is the movement of large portions of the operating system into hardware and firmware.

Finally, the trend in computer architecture of functionally specialized distributed processing (e.g. I/O processor, array processor, etc.) has led to the concept of a data base machine attached to the back-end of a host computer or directly to terminals through a query language interface. There is currently a large amount of industry activity in this area with some systems already out.

7.2 Software

The preeminent upcoming change in the DoD area is the advent of ADA. Initial release of a validated DoD developed Ada Compiler is scheduled for early 1983 for the DEC VAX computer family. In addition, commercial interests are developing their own Ada compilers. The largest projected near term commercial software (S/W) impact is the large scale use and further development of Application Generators on the basis of their potential to dramatically increase S/W productivity and in some case allow the end user to directly build applications without programmers. (e.g. IBM's ADF, National's CSS Nomad, DEC's Admius 11, etc.) In the same vein, large scale use of query languages and their further development is also expected, thus allowing end users to directly interface with their application data bases without programming support or skills. In addition, the adoption of a standard query language offers the hope of compatibility and possible transportability between different DBMSs and computers. Along this line relational data bases are projected to be the data bases of the future, partially for human factors reasons, but primarily for their flexibility in creating new applications.

7.3 Computer Communications

The use of available low cost ($500 to $2K range) high speed (10Mbd to 100Mbd) fiber optic links will easily handle local point to point computer system links. These links are readily available in hybrid form and are standard on the Intel, DEC, and Xerox 10Mbd 2.5Km ETHERNET Standard. Development of even lower cost integrated ETHERNET transmitters and receivers has been underway

for some time. Commercial use of DES encryption for computer security should become widespread as low cost integrated circuit versions are developed.

7.4 Mass Storage

The development of the Winchester disk drives has solved the majority of previous mass storage problems. The result is low cost, high reliability mass storage with a standard dual port option allowing fault tolerant designs. The majority of the low end Winchester disk cost (≃ $2K) is in the disk controller. Disk controllers employing custom LSI chips are now appearing that reduce component counts, creating cost, reliability, and performance benefits. The next major advances in mass storage are already underway in the optical disk area. This development will greatly increase future mass storage capacity.

7.5 Memory

The forecast for memory is for large production of 64Kbit chips driving memory cost down further. This development is lagging earlier prodictions but Japanese firms are now introducing such chips. Because of reduced cell size, alpha particles have caused an increase in soft error problems further reinforcing the trend to error correcting coding. Another development of interest is serious development of EEROM (e.g. Intel 2816). Such electronically erasable ROM offers tremendous protection advantages for static security related data bases, i.e., by controlling associated write circuitry the security officer could control updates to EEROM resident security data.

8.0 CONCLUSIONS AND RECOMMENDATIONS

A distributed architectural approach to secure multi-level data base management has many benefits and is quite feasible using existing technology. The primary benefit is provable hardware enforcement of non-discretionary security. There are many possible implementations that allow trade-offs over a large range of complexity, cost, performance, reliability, flexibility, functionality, features, security granularity, etc. These trade-offs generally follow the rule of diminishing returns and with increasing complexity these sytems usually become more specialized and less generally applicable. For example, by providing a general design approach addressing some 80% of the requirements and design goals, and capable of satisfying maybe 80% of the potential applications, a net 64% solution is achieved. However, if instead, a design meeting 95% of all the requirements and design goals becomes so specialized as to apply to perhaps only 10% of all applications, a net solution of only 10% is achieved. Therefore, a general approach with some compromises is preferable to attain general usability, particularly if it is sufficiently flexible so as to be usable as a basis for more specialized design.

8.1 Conclusions

Feasibility

This architecture easily solves the multi-level security problem and is a partial solution to the multi-compartment problem limited only by financial considerations. The dominant acquisition cost is the DBMS processor subsystem which grows linearly with the number of concurrent dedicated levels and compartment combinations required for a given application. Thus, the availability of a suitable low cost DBMS processor is the key to using this architecture to handle applications with large numbers of concurrent compartments.

Non-Discretionary Protection

The primary advantage of this architecture is that it provides provable hardware enforcement of non-discretionary security for single level processors sharing a common multi-level data base. By physically isolating these single level processors and providing appropriate physical access controls, certifiable multi-level operation is feasible without certified or trusted software,

i.e., there is no requirement for a secure OS to enforce non-discretionary security.

Discretionary Protection and Integrity

If a standard OS is deemed adequate to enforce discretionary security and integrity (that is, the OS can be trusted), then this architecture will allow the use of standard commercial software, such as a standard DBMS to be used for multi-level applications. This offers the potential for large cost savings relative to development of custom system software for secure OS, DBMS, etc. This approach also provides the greatest flexibility, making non-discretionary multi-level security an add-on with no overhead except for increased central data base latency. Because of this flexibility it can be used to upgrade existing systems and support new applications.

Alternately, it is technically feasible to enforce discretionary security and integrity with certified software/firmware. However, this specialized software drastically reduces flexibility, e.g. compatible commercial software, and is unlikely to be provable.

Three technical approaches for providing certified discretionary protection were found:

- Use a secure OS and compatible DBMS in a host computer
- Use a certified front-end I/O processor, end-to-end encryption and a custom back-end DBMS; and
- Use individual personal computers and a modified OS.

For these three approaches, the distributed architecture of the secure DBMS designs offers the advantage that the discretionary access control software can be placed in the central access control subsystem, thus physically isolating this discretionary security software from potentially hostile user code environment. These last two approaches provide "external" enforcement of discretionary security, require less certified code, and can be placed in firmware, advantages not found with the traditional secure OS approach firmware. The hardware cost for these advantages, unfortunately is very high, in either individual encryption equipment or computers. Thus, providing certified discretionary security and integrity is considerably more difficult than providing certified non-discretionary security.

Data Base Selection

A relational DBMS approach was selected for the purpose of a greater detail system evaluation; however hierarchical or CODASYL-DBTG DBMS approaches are also feasible. The formal math model developed for a secure multi-level relational DBMS is generally applicable to these other approaches as well, although the impact is different. The requirements for both secure multi-level operation and physical implementation of a relational DBMS result in some fundamental limitations and make other operations difficult. In particular, the schema (description of data structure), primary keys (data that uniquely determines a record), and associated access structures (e.g. hash code tables, pointers, etc) must be at the lowest level and common compartment(s) of a multi-level relation. The simple and *-security rules combine to prohibit downgrading. Thus, a system high DBMS processor with trusted users and software is required by this architecture for downgrading.

Multi-Level Protection

Three feasible approaches for providing a secure multi-level relational DBMS were identified:

- o Multi-Relation View approach
- o Domain-Separation approach, and
- o Internal Bypass approach

The multi-relation view approach was the only technique permitting the use of standard relational DBMSs. In this approach relations are restricted to a single level with views of multiple relations used for multi-level operations. This approach requires segmenting a logical multi-level relation into multiple single level relations with the primary data base key(s) replicated in each relation. This approach to multi-level security increases DBMS overhead due to the processing requirements to write the view segments to form the users view of the data and the additional storage required for the replication of the data base keys at each level. This overhead is linearly proportional to the number of security classifications in a view and is unlikely to be excessive for most applications. Other difficulties with this approach are multi-level entries or updates cannot be made from a single level machine and partial logical multi-level records can exist in the data base.

To use a domain separation technique requires the development of a custom DBMS for secure multi-level DBMS operation. This approach incorporates classifications within the schema and segments domains by clearances into different single-level files. The overhead for this approach is similar to the multi-relation approach since a multi-level record must still be pieced together from different files. However, if a flat storage structure is used (i.e., no variable length physical fields), then it is possible to make multi-level entries and updates from a single-level machine operated at the lowest data level and common data compartment. Entries and deletions must be made from this level. In addition, because a custom DBMS is required, integrity controls can be provided at either the domain or data item level.

The third technique is the internal bypass approach. It requires an upgraded hardware security filter that handles multi-level files and provides a read-modify-write capability: internal bypass, buffer and merge hardware. In addition, a custom DBMS similar to that required by the Domain-separation approach to exploit this multi-level file capability is also required. Since multi-level records can be stored in a single multi-level file, this approach does not have the direct overhead of the other approaches. There is an indirect software overhead for this multi-level operation in terms of formatting security classifications within the file and possible hardware overhead in terms of effective disk storage and transfer rates. Again, integrity controls may be directly incorporated into this custom DBMS. The requirement for multi-level file support with this approach requires a significantly more complex and extensive HSF and is thus not recommended.

Hardware Security Filter

The hardware security filter (HSF) is the key to providing provable hardware enforcement of non-discretionary security. It acts as an entrance and exit guard separating data by classification and controlling which levels and compartments may be read and/or written by a given processor in accordance with an access control policy (ACP) supplied by the security officer. Four separation techniques were found, with static encryption being technically best and the most costly. The primary advantage of this technique is that failures in the central data base subsystem, such as that resulting in the wrong data being read, will not compromise security; i.e., with this approach the central data base is provided

a security fail safe guarantee. However, static encryption does not provide COMSEC link protection. Additionally, while the central data base stores encrypted rather than clear text data and hence is more secure, it will still be a sensitive, possibly system high, compartment. Hybrid data separation techniques are also feasible and beneficial. Comparisons of the four primary separation techniques are shown in Figure 8.1-1.

The hardware security filter can be built to handle either single-level or multi-level files. (Multi-level DBMS operation does not require multi-level files). The single-level files HSF is relatively easy to build and is estimated to involve 5 to 10 line replaceable unit (LRU) types and occupy less than a rack. Tempest, (security failure analysis, SFA) and possibly crypto requirements are expected to heavily influence the detailed design, e.g. reclocking, redundancy, self-checking, etc.

The multi-level files HSF is much more complex requiring on the fly screening of data items against access control policy and a read-modify-write capability. In addition this type of HSF must enforce that the dynamic, internal file security level structure can be created (written) only from the lowest level and common compartment(s) of the file to prevent a high speed covert channel. Hence a secondary file level separation technique is also needed. The read-modify-write capability requires an additional internal bypass buffer, and merge hardware which was not required for a single level files HSF. Obviously, a multi-level files HSF is much more complex than a single-level files HSF and correspondingly more costly to develop and certify. Custom software is required to support a multi-level files HSF as well.

Access Control Subsystem

Because the HSF is shared by numerous processors, a mechanism to handle this contention is required. Arbitration and queuing of these processors requests for central data base access is easily handled by preceding the HSF with an access control subsystem (ACS). This is also a convenient location for a central audit trail, security monitor, and possibly discretionary access and integrity control. It is estimated that a small minicomputer is quite suitable for performing these functions.

	SECURITY RANKING	MULTI-LEVEL FILE CAPABILITY	PERFORMANCE PENALTY	DYNAMIC STORAGE ALLOCATION	COMPLEXITY	HARDWARE SIZE	COST
CRYPTO KEY *	1	YES	NO**	YES	4	4	4
PHYSICAL TAG	2.5	YES	YES	YES	3	3	3
ADDRESS TAG	2.5	NO	NO	NO	2	2	2
DIRECTORY TAG	4	NO	NO	YES	1	1	1

Ranking - 1-. BEST; 4- WORST

* Has SFA Benefits
**Indirectly Yes for Multi-level Files - Security Key Format Must Be Provided

- o Directory Tag Required for Audit Trail and Central Discretionary Security and Integrity Firmware
- o Other Techniques Limited to HW Enforcement of Non-Discretionary Security at Processor Level
- o Multi-Level File Capability Also Requires Secondary File Separation to Eliminate Covert Channel
- o Hybrid Approaches Possible and Beneficial

Figure 8.1-1 Hardware Security Filter Data Separation Technique Tradeoffs

While the ACS contains security related software, the requests relayed through the ACS are internally audited by the HSF for compliance with ACP. Consequently, there is no requirement for certified or trusted code in the ACS to enforce non-discretionary security for single level processors. Because all central data base requests are funneled through the ACS, it is the ideal location for a central audit trail. The security monitor function provides the primary security check on multi-level processors, alerting the security officer to all downgrading (a write access requests) activity. Central discretionary access and integrity control software can also be located in this system.

All these security functions require the requesting DBMS subsystem processor OS to identify the actual user and logical object (file). Hence, a logical access request interface with these processors is preferable with translation to a physical access request to the HSF occuring within the ACS. To eliminate the possibility of covert channels, this logical access request interface should be unidirectional. The ACS must be run in a dedicated system high mode.

Another advantage of this ACS architecture is that the security related software and the central security data base in the ACS is physically isolated from potentially hostile user code. If certification of all or portions of the ACS software is desired, a distributed dedicated microprocessor/firmware approach is preferable. This approach can be used to segment the ACS software into smaller tasks and eliminate the requirement for concurrent processing within any microprocessor, thereby greatly simplifying the certification process. The static portion of the ACS security data base can be further protected by storing it in Electronically Erasable ROM (EEROM). By locating the support circuitry required to enable EEROM modification on a removable board, the security officer can physically control changes to the primary security data base such as user's clearances, and still be able to easily modify this data base.

DBMS Processor and Interface

A key technical challenge in this architecture is the sharing of a single central data base among numerous DBMS processors without limiting individual DBMS processor performance. The challenge is actually twofold: supplying data at disk rates to a locally distributed star network and providing enough total central data base throughput. Use of existing commercial fiber optic links with a block link

protocol will easily handle data distribution at disk rates up to several hundred meters. The obvious solution to the data base throughput problem is to increase the number of central data base channels. Unfortunately, that solution requires similar expansion of the HSF. A solution with more finesse is to reduce central data base demand. This can be accomplished through use of virtual memory (VM) technology in the DBMS processor and DBMS. The key is to read a relation's files once from the central data base when a user opens a relation and to use a local copy thereafter for query processing. This approach requires each DBMS processor to have a local system disk, i.e. a distributed hierarchical storage architecture. Specialized write interlock facilities are required to provide consistent multi-user views of the data base.

There are two fundamental architectural approaches to DBMS processor systems. One is to use a general purpose computer. More recently, the approach has been to use a functionally specialized data base machine (back-end DBMS). The data base machine's user interface is limited to a query language and thus another (host) processor is required to run user application programs, although directly connected interactive user terminals may be supported. This offers the advantage that no potentially hostile user code can exist in the DBMS processor. In addition, query languages generally reduce skill level requirements and often allow end users without a programming background to interactively make direct use of a DBMS.

Choice of Data Base Type : Relational

The two primary relational DBMS advantages are its good man-machine-interface (MMI) and its flexibility (Figure 8.2-1). Because a relational DBMS MMI can be thought of as a set of tables it is very easy for people to visualize. Additionally a user can join tables and select various columns, so as to create his view of the data base, again in tabular form. This flexibility supports rapid development of new applications, i.e., new applications do not require data base modifications. Conversely, the primary disadvantage of a relational DBMS is the large amount of memory and processor power utilized.

Reliability

Another potential architectural driver is reliability requirements, especially if fault tolerance is required. The three critical subsystems are the central

• EASE OF USE:	Two dimensional table representation of relations is simple
• FLEXIBILITY:	Relational Operators such as Project & Join permit relation formation in forms required by a variety of users and applications
• PRECISION:	Relations are precise in meaning and can be manipulated through mathematics of Relational Algebra or Relational Calculus
• SECUIRTY:	Security controls can be easily implemented at the relation level
• RELATABILITY:	Attributes from different sets of tuples or different "files" can be easily related
• EASE OF IMPLEMENTATION:	Physical storage of "flat files" can be less complex than physical storage of Tree and Plex structures. Use of normalization may be of benefit.
• DATA INDEPENDENCE:	If the data base is in a normalized form with data independence in the software, data can be restructured and the data base can grow without forcing rewriting of application programs. This feature results in considerable data base Management cost savings.
• DATA MANIPULATION LANGUAGE:	Data Manipulation Sublanguage can be based upon relational algebra or relational calculus.
• CLARITY:	Vast tree diagrams depicting the logical organization of the data base are never necessary. Lack of use of pointers-to-pointers-to-pointers reduces the logical complexity of the data base and permits easier expansion as users needs grow.

Figure 8.2-1 Advantages of a Relational Data Base

data base, the HSF and the ACS. Winchester disk technology is capable of providing the required central data base reliability. However, spare disk capacity and back-up DBMS utilities will be required for recovery operations. The key to high HSF reliability is a design with a minimal parts count consistent with (SFA) concerns. Good fault isolation and sparing or a standby are the primary mechanisms for good ACS reliability. If fault tolerance is required, a parallel disk controller, dual ported disk, dual channel HSF, hot standby ACS architecture is desirable. The central data base is off the shelf and the hot standby is standard design. The HSF however requires a common ACP but independent channels for SFA concerns. A solution is for the two channels to share a triple mode majority voting ACP system.

8.2 General Recommendations

It is recommended that a standardized general purpose ACS and HSF be designed as a security filter. Special, custom I/O handlers can then be designed to transparently interconnect this security filter to the desired DBMS processor and disk systems. In other words, the DBMS processor should merely "see" a set of single level "disks" with independent directories and should be unaware of the presence of any other DBMS processors, the ACS or the HSF. It is further recommended that the HSF provide certified non-discretionary security enforcement independent of other subsystems. Furthermore, it is recommended that the basic minimum design be a low cost system with standard modular options for expansion and upgrading for example enhancements for fault tolerance, etc.

8.3 Resulting Technical Recommendations

Based on the preceding general recommendation the following basic technical baseline is recommended:

- the basic HSF support single-level files using static encryption and address separation techniques
- the multi-relation view approach be used to process multi-level relations
- a commercial DBMS be used
- a commercial operating system be used

- o the ACS be initially implemented with a trusted microprocessor system and later upgraded to a certified, system through a validated, distributed microporcessor/firmware ACS architecture.
- o the HSF be expandable for a range of 4 to 32 DBMS processors.
- o the entire security system (ACS, HSF, central data base subsystem, etc) support a fault tolerant configuration based upon dual ported disk supporting parallel controllers with twice the potential data base throughput.
- o the HSF be designed to permit future add-ons (P^3I -preplanned product improvement) to provide multi-level file handling using fixed format security tags

8.4 *Development Recommendations*

It is recommended that a distributed secure multi-level DBMS architecture be developed in a staged, evolutionary fashion with initial emphasis on the hardware security filter.

The initial stage would consist of design of a modular HSF, development of a basic prototype and minimal ACS and evaluation and certification of the HSF.

The second stage would concentrate on the development of a more extensive ACS, certification of other HSF configurations, and integration into some operational test sites.

The third phase would concentrate on the certification of a revised ACS with subsequent widespread use of the system in suitable applications.

If commercial development of low cost data base machines parallels this HSF and ACS development, the resulting system is likely to be a relatively low cost approach to multi-level security.

APPENDIX A

Formal Mathematical Security Model

INTRODUCTION

This section details a formal mathematical proof of the requires for a secure system using the concepts of DoD security rules, need-to-know, an integrity policy. The discussions are presented in detail in this section reference purposes and are summarized in the discussion of security requires of Section 2.

The material presented here is divided into the following sections:

A.1 DATA SECURITY MATHEMATICAL MODEL

A data security mathematical model is an embodiment of security requirements into a logical and systematic collection of rules which can be proven to provide the desired degree of protection. There are several approaches which can be taken in the formulation of a security model. Historically, there are approaches described by Bell[6], Hinke and Schaefer[7], Bell and LaPadula[1], Grohn[2], and Robinson[8], and many others. The models of Bell-LaPadula and Grohn are adopted with alterations for the purpose of this study.

The data base security math model is used to generate the requirements of actual data base handlers. In order for the data base to be secure, two major conditions must be provably true: all software utilized to permit a user to access or modify any protected entities of a data base must be flawless, complete, and totally specific in the restrictions required for security, and all hardware used to enforce security must either be totally flawless or be sufficiently redundant to prevent a security failure. In this study, it will be assumed that all security enforcing software will be created in a secure environment by persons of adequate integrity and that sufficient code verification will be performed. To facilitate the ease of qualifying software, it will be further assumed that all software routines will be size restricted. With these assumptions, it will be concluded that the software can be created without any flaws. Hardware requirements for a secure data base system are detailed in a later section,3, wherein the system configuration and requirements are detailed.

It is the purpose of the data base model to ensure that the security-relevant enforcing software is complete and specifies all security enforcement requirements. In this study, mathematical proofs validate the data base model. General discussions of the security protection requirements are presented in Section A.3 of this report. Formal proofs are detailed in a later section, A.5.

Section A.2, which follows, prefaces the model discussions by presenting some of the mathematical concepts required in presentation of the math model.

Comparisons to Past Models

As mentioned above, the model discussed in the sections below are based upon the models described by Bell-LaPadula and Grohn. Grohn's model itself is based upon that of Bell and LaPadula. The model of Bell and LaPadula describes the requirements for a secure system and the requirements for preservation of a secure system during state transitions of the system. The secure system is defined to be secure in terms of DoD security requirements and consists of non-discretionary and discretionary access controls.

The model for data base security is described in terms of secure state transitions. State transitions define the changes in elements of a secure state. A transition is a secure transition if and only if it results in a new system state which is also a secure state.

Grohn's model uses the same definitions of security and adds an additional feature to the model called integrity. Integrity refers to the "trustworthiness" of information added to a data base. Security and integrity are subsequently combined into what is termed a protection level. In addition, Grohn defines a tranquility principle

which declares that entities in a system shall not alter their protection level.

The data accesses are controlled by a DBMS directory. The Bell-LaPadula model performs data accesses through a tree directory with certain dominance relationships between parent and child. The Grohn model utilizes a set of directories, each directory at a given protection level. Several other model differences exist. However, these are some of the principle elements of these previous models.

The model for this study will rely heavily upon the model described by Grohn, utilizing his extensions of the Bell-La-Padula model. However, the approach to the declaration and proof of the model is different from those of Grohn and Bell-LaPadula. For example, rather than emphasize the constituents of Protection (security and integrity) and their characteristics, the model emphasizes protection and treats its constituents (security and integrity) as secondary. Also, the proofs of Grohn lacked mathematical formality. Bell and La Padula's proofs are considerably stricter and therefore, their approach is used with the necessary modifications for this model. However, the Bell-LaPadula theorems are often difficult to comprehend and therefore the attempt here is to present these proofs in a better organized and more intelligible manner.

A.2 Mathematical Concepts

A number of mathematical tools and relationships are required for the discussion of the Secure DBMS and during the validation of the mathematical model. This section presents those concepts describing the data base model in a formal manner.

1) SET

A set is an unordered collection of mathematical elements (such as numbers, characters, etc) that are listed or are identified by a common characteristic or by a rule of formation. The enclosing brackets {.} denote a set. The symbol $\emptyset$ denotes the empty set, also called a null set.

Some examples of a set are:

A list of grouping A= {cat, dog, horse} set of animals identified by characteristic B={all prime numbers} identified by a formulation C={all numbers with integer square roots} ={1,4,9,16,25,...}

$$C=\{x_i\} : x_i^{\frac{1}{2}} \text{ is an integer}$$

Because a set is unordered, the sets {1,2,3} and {3,1,2} are equivalent.

Subdividing a set by retaining only portions of the original set generates a subset. From the above example, a subset D of set A above is D={cat, dog}. The null set is a subset of all sets. The symbol for the subset operation is $\subset$: $A \subset B$ means that A is a subset of B. Formally a subset is defined $A \subset B$ if $A \cap B = A$ where $\cap$ is the interseciton of the two sets and is defined as an operation below.

2) UNION

A union is a logical set operator which includes in a resulting set A all the elements of the two sets it operates upon. Duplicate elements are deleted from the resulting set. The symbol $\cup$ is used to denote the union.

For example:

Let B={1,2,3}

C={2,5,10}

D={1,10,25}

$A_1 = B \cup C = (1,2,3,5,10)$

$A_2 = C \cup D = (1,2,5,10,25)$

$A_3 = B \cup C \cup D = B \cup (C \cup D) = (1,2,3,5,10,25)$

The union operator is associatative and commutative.

3) INTERSECTION ∩.

An intersection is a logical set operator which Includes in a resulting set A only those items common to the two sets it operates upon. The intersection of the two sets with no common items is the null set, ∅.

For example: using the same sets A,B,C,D from the above union example.

$A_1 = B \cap C = \{2\}$ $\quad A_2 = C \cap D = \{10\}$

$A_3 = B \cap C \cap D = (B \cap C) \cap D = B \cap (C \cap D) = \emptyset$

The intersection operator is both associative and commutative.

4) SET SUM: +

The set sum is a logical operator identical to that of a set union. For example, if B and C are sets, then A = B+C = B∪ C. The set sum is commutative and associative.

5) SET DIFFERENCE: -

The set difference is a logical operator which includes in the resulting set those elements differing between the two sets it operates upon. If the two sets operated upon are identical, the set difference is the null set, ∅. For example, using sets B,C, and D from the Union discussion;

$A_1 = B-C = \{1,3,5,10\} = C-B$

$A_2 = C-D = \{1,2,5,25\} = D-C$

$A_3 = B-C-D = (B-C)-D = B-(C-D) = D-B-C = \{3,5,25\}$

$A_4 = B-B = \emptyset$

The set difference is commutative and associative.

6) CARTESIAN PRODUCT: X

A cartesian product (a vector product) of two sets creates a resulting set consisting of a set of ordered pairs. If A and B are two sets and a_i are elements of A and b_j are elements of B, then the cartesian product of AxB is the set $\{(a_i,b_j)\}$ for all i and j.

For example, let A ={a,b} and B ={1,2,3,4}.

Then, AxB = {(a,1), (a,2),(a,3),(a,4),(b,1),(b,2),(b,3), (b,4)}

The ordered elements of the cartesian product are referred to as tuples, or n-tuples where n is the number of elements. The cartesian product is associative but is not commutative due to the ordering of the tuple elements. The size of the tuples which are the elements of the resulting set is equal to the sum of tuples in the sets in the product.

For example, let A = $\{(a_1,\alpha_1),(a_2,\alpha_2),\ldots\}$, (2-tuple)

B = $\{(b_1,\beta_1),(b_2,\beta_2)\ldots\}$(2-tuple)and C = $\{(c_1,\gamma_1,1)$

$(c_2,\gamma_2,2),\ldots\}$(3-tuple) then R=AxBxC = $\{a_1,\alpha,b_1,\beta_1,c_1,\gamma_1,1)$,

$(a_1,\ _1,b_1,\beta_1,c_2,\gamma_2,2)$,etc} which is a 7-tuple.

7) TUPLE OR N-TUPLE

An n-tuple is an ordered group of N items. An n-tuple is formed as the result of a cartesian product. An example of tuples are (1,2,a) 3-tuple

(dog,cat,house,boy) 4 tuple

8) POWER NOTATION: A^B

Given that A and B are sets, the power notation A^B denotes the set of all functions from B to A. For example, let A= a,b , and B = 1,2,3 , then

$A^B=\{f_1,f_2,f_3,\ldots\}$such that

f_1= {(1,a),(2,a),(3,a)} f_5={(1,b),(2,a),(3,a)}

f_2={(1,a),(2,a),(3,b) f_6={(1,b),(2,a),(3,b)}

f_3={(1,a),(2,b),(3,a) f_7={(1,b),(2,b),(3,a)}

f_4={(1,a),(2,b),(3,b) f_8={(1,b),(2,b),(3,b)}

If n_A is the number of elements of A and n_B is the number of elements of B, then the number of functions in the set A^B is $(n_A)^{n_B}$.

A given function, f_i, defines the relation between set B and set A. For example, given f_3 is chosen, the element of A corresponding to the B element b is 2. That is, (2,b) is a tuple of f_i containing the B element b.

9) DOMINATES: $\succcurlyeq$

The dominates operator, $\succcurlyeq$, is a binary operator. A relation, R, such as $\succcurlyeq$, is a partial ordering relation if and only if the relation is reflexive, antisymmetric, and transitive. Grohn[4] proves these requirements for $\succcurlyeq$ and it will not be repeated here. For example, let

$M=\{L_1, L_2, L_3 \ldots L_r\}$ where $L_i = (a_i, b_i)$ and a_i is the element of A, and b_i is a subset of set B. Then L_i is said to dominate L_j if and only if their elements dominate.:

$L_i \succcurlyeq L_j$ iff

(i) $a_i \geq a_j$ and

(ii) $b_i \supseteq b_j$.

10) SET OF SUBSET OPERATORS: P(.)

P(.) is an operator which denotes the set of all subsets of the set which it operates upon. For example, if X is the set

X={A,B,2} then

P(X) = {A,B,2},{A,B},{A,2},{B,2}, Ø,{A},{B},{2}}

11) EXCEPTION OPERATOR: $\setminus$

The exception operator is a binary logical operator used to permit change of the contents of a set or proposition. The expression $A \setminus B$ means propositional A except as modified by B. For example, define a function (see A^B) f:

f = {(A,1), (B,2),(C,3)}. Now f(A)=1, f(B)=2, and

f(C)=3. However, if the function f is modified as

$f' = f \setminus (B,8)$, the new function is f'={(A,1),(B,8),(C,3)}

and f'(B)=8.

A second example using a matrix is of interest in the area of discretionary protection. Given two-dimensional matrix A with elements A_{ij}. The following operations are defined

$A \setminus A_{ij}=a$ change element A_{ij} to value a

$A \setminus A_{ij}+b$ add b to A_{ij}th element.

Finally, a third example wherein $\setminus$ operates on a set of elements. Given the set $z=\{1,2,3\}$, a new set z' can be formed as $z'=z \setminus 3 \leftarrow A,B$ whereupon $z'=\{1,2,A,B\}$ where the elements A,B replace the element 3.

The math model uses an extremely large number of symbols. Figure A-1, presented below, summarizes these symbols.

MODEL SYMBOLS

b	current access set	$b \in P(SxOxA)$
M	discretionary Access Matrix	$M_{ij} = \underline{x}$ where $\underline{x} \in A$ for O_j and S_i
P	Set of all Protection levels	$P = P_s \times P_o \times P_c$
F	Set of data base forms	$F = \{H,R,C,\ldots\}$
S	set of valid Subjects	$S = \{S_1, S_2, \ldots S_i, \ldots S_n\}$
O	set of valid objects	$O = \{O_1, O_2, \ldots O_j, \ldots O_m\}$
A	access mode	$A = \{\underline{r}, \underline{e}, \underline{w}, \underline{a}\}$
$\underline{r}$	read access	$\underline{r} \in A$
$\underline{e}$	execute access	$\underline{e} \in A$
$\underline{w}$	write access	$\underline{w} \in A$
$\underline{a}$	append access	$a \in A$
B	all possible access sets	$B = SxOxA$
MOD	modify access mode	$MOD \in \{\underline{w},\underline{a}\}$
OBS	observe access mode	$OBS \in \{\underline{r}, \underline{e}, \underline{w}\}$
S_i	an individual subject #i	$S_i \in S$
O_j	an individual object #$_j$	$O_j \in O$
$\underline{x}$	general access mode	$\underline{x} \in X$
X	set of subsets from access A	$X = P(A)$
$\rightsquigarrow$	*protection dominance*	$A \rightsquigarrow_p B \Leftrightarrow a_s \geq b_s,\ a_i \leq b_i,\ \alpha_s \supseteq \beta_s\ \alpha_i \subseteq \beta_i$
M_{ij}	set of accesses for subject i to object j	$M_{ij} \subseteq P(A)$
V	set of all possible system states	$V \subseteq BxMxPxF$
v	current system state	$v \in V \quad v = (b,M,P,F)$
v*	*next system state*	$v^* \in V$

Figure A-1 Math Model Symbol Definitions

ρ	set of all possible rules	$\rho \subseteq (DxV)^{(RxV)}$
ρ_i	a particular rule from ρ	$\rho_i \varepsilon\ \rho$
M*	next discretionary Access Matrix	$(D_m, v^*) = \rho_i(R_k,v)$, $v^* = (b,M^*,P,F)$
b*	next current access state	$(D_m,v^*) = \rho_i\ (R_k,v)$, $v^* = (b^*,M,P,F)$
R	set of Requests	R={get append access, get read access, add permission, etc.}
D	set of Decisions	$D \varepsilon$ {?, TRUE, FALSE}
R_k	a system Request	$R_k \varepsilon R$
D_m	a request Decision	$D_m \varepsilon D$
β	current Access set function	$\beta \varepsilon (OxA)^S$; $\beta(S_i) = \{O_1,X_1),\ldots(O_j,X_j)\ldots\}$
T	set of all Times	$T = \{t_0,t_1,t_2,t_3,\ldots t_\infty\}$
t	a specific time	$t \varepsilon T$; i.e, $t = t_0$, t_1, t_2, or t=0,1,2...
C_s	set of all Security Classifications	C_s = {TS, S, C, U}
c_s	Element of set of Security Classifications	$c_s \varepsilon C_s$
K_s	Set of all Security Categories	K_s = {NORFORN, NATO, ...etc.}
k_s	Subset of set of all Security Categories	$k_s \varepsilon P(K_s)$
C_i	Set of all Integrity Levels	C_i = {1,2,3,...} (for example)
c_i	Element of set of Integrity levels	$c_i \varepsilon C_i$
K_i	Set of all Integrity Categories	K_i = {A,B,C,D,...} (for example)
k_i	Subset of all Integrity Categories	$k_i \varepsilon P(K_i)$
Π	Set of all possible protection levels	$\Pi = C_s \times P(K_s) \times C_i \times P(K_i)$
π	Element of set of possible protection levels	$\pi \varepsilon \Pi$
P_s	Set of functions mapping subjects to maximum possible protection levels	$P_s = \Pi^S$
p_s	A function element of P_s	$p_s \varepsilon P_s$
P_o	Set of functions mapping Objects to possible protection levels	$P_o = \Pi^O$
p_o	A function element of P_o	$p_o \varepsilon P_o$
P_c	Set of functions mapping Subjects to current possible protection levels	$P_c = \Pi^S$

Figure A-1(Cont'd) Math Model Symbol Definitions

p_c	A function element of P_c	$p_c \in P_c$
P	System protection level function mapping max. subject level, object levels and current subject levels	$P = P_s \times P_o \times P_c = (p_s, p_o, p_c)$
W	The action of a rule, ρ	$W = (R_K, D_m, v^*, v)$
$W(\rho)$	A rule limited action set	$W(\rho) \subseteq W$
ζ	A mapping of states, V, into time	$\zeta \in Z$
Z	A functional mapping of all possible system states into time	$Z \subseteq V^T$
ζ_t	the t^{th} state in a state sequence	$\zeta_t = (b^t, M^t, P^t, F)$

$P(\ )$ subset operator

Figure A-1 (Cont'd) Math Model Symbol Definitions

A.3 Model Description

The following section discusses and defines the elements of the formal math model.

A.3.1 State Model Description

The current system state, v, consists of four items: the current access set,b; the discretionary access matrix,M; the non-discretionary protection level,P; and the data base form,F. The symbol used to represent the system state is v, where v is the 4-tuple v=(b,M,P,F). A secure system state is defined through valid specifications of its constituents, b,M,P,F. The following sections describe the system state in detail:

A. 3.2	Current Access Set	b
A. 3.3	Protection Levels	P
A. 3.4	Access Permission Matrix	M
A. 3.5	Data Base Structure	F

An additional section, A.3-6, discusses a group of subjects exempt from the security features imposed upon the secure system: Trusted Subjects. Trusted subjects are a necessity for many facets of the data base functions. Finally, the property of Rules in the Math Model is detailed in Section A.3-7.

A.3.2 Current Access Set:b

The Access Set

The current access set,b, is a set containing 3-tuples which define the modes in which objects in the system are currently accessed by the subjects:

$$b=\{(S_1,O_7,\underline{r}),\ (S_{10},O_3,\underline{e})\ \ldots\ (S_iO_j,\underline{x})\ \ldots\}.$$

A general element in the current access tuple is written $(S_i,O_j,\underline{x})$ where

S_i is the subject, $S_i \varepsilon S$ where S is the set of all subjects

O_j is the object which the subject S_i has accessed, $O_j \varepsilon$ O where O is the set of all objects.

and $\underline{x}$ is the mode in which O_j has been accessed, $\underline{x} \varepsilon$ A, where A is the set of all access modes.

The tuple $(S_i,O_j,\underline{x})$ means that subject S_i currently has $\underline{x}$ access to object O_j.

Changes to the system state which result in a user obtaining access to a file or other data base object or results in a user relinquishing access to such an item will cause an alteration in the current access set. The process which gives a user access of any type (read,write,etc.-see Section A.3.4) results in the addition of the appropriate tuple describing this new access to the current access set. Releases of access to an object by a user results in the removal of the appropriate tuple from the current access set. For example, if $b=\{(S_1,O_5,\underline{r}),\ (S_3,O_4,\underline{w}),\ (S_3,O_4,\underline{r})\}$ and subject #2 opens file object 3 in the append mode, the new current access set, b* becomes $b^*=b + \{(S_2,O_3,\underline{a})\} = \{(S_1,O_5,\underline{r}),(S_3,O_4,\underline{w}),\ (S_2,O_3,\underline{a})\}$. Furthermore, if subject #1 releases object 5, then the new access set is $b^*=b-\{(S_1,O_5,r)\} = \{(S_3,O_4,\underline{w}),\ (S_2O_3,\underline{a})\}$. A set of security specifying rules discussed later in this report govern the system state transitions, such as illustrated above.

We now describe the elements of the current access set.

Subjects

In this model, subjects are the active entities within the system. As such, subjects are either users of the facility or users of application programs. Inactive programs stored within the data base protection areas are considered objects (see below). As such, we make no restrictions regarding entities that may be both subjects and objects beyond the fact that when an entity is active, it is always treated as a subject. Note that only subjects can cause changes to the system state.

An arbitrary subject is referenced as S_i indicating the i^{th} subject. S_i is an element of the set of validated users and programs, S: $S_i \varepsilon S = \{S_1, S_2, \ldots, S_n\}$ where n is the total number of subjects. The number of validated subjects may increase or decrease only through the actions of the trusted subject or data base manager (See 2.2.6).

Objects

Objects are the passive entities in the secure DBMS model. As such, objects do not act upon other model entities and cannot generate any system state changes. As previously mentioned, some objects may be application programs and therefore are considered subjects while actively processing.

The referenced O_j indicates the arbitrary j^{th} object. O_j is an element of the set of all objects in the system,

O: $O_j \varepsilon O = \{O_1, O_2, O_3, \ldots, O_m\}$ where m is the total number of objects in the system. The number of objects may increase or decrease as a function of time as the result of the activities of subjects.

Accesses

In the course of interacting with data base objects, subjects must access the objects in one of the permitted access modes. In general, a subject can access an object in two general modes; 1) to observe (referenced as OBS) and 2) to modify (referenced MOD). These two basic access modes are further reduced to the familiar set of access modes A: A= {r,e,w,a} where;

r is read permission - observation but no modification

e is execute permission-no observation or modification

w is write permission-observation and destructive modification

and a is append permission - no observation and non-destructive modification

As such, MODϵ{w,a} and OBS ϵ{r,w}. For notational convenience, a general access mode is written as x, as in the tuple (S_i, O_j, x) where $x \epsilon A$.

Properties of b:

The current access set b is a subset of all possible access modes between subjects and objects, B: $b \subseteq B = S \times O \times A$. This cross product creates a set B consisting of all possible tuples with elements from the subject set,S, the object set,O, and the access set, A. The protection requirements (Section A.3.4) and the discretionary access controls (Section A.3.3) of this model restrict the set of accesses of b. This means that b is a subset of B because the security protection constraints of the model specify only certain access modes to objects by a given subject or because the subject responsible for the object does not wish to grant discretionary access of the object to the particular subject. Figure A-2 illustrates the subset b in the form of a Venn diagram.

The Current Access Function: $\beta(s)$

In the course of the data base security model proofs, a useful function is the current access set function, β. β is an element of the set of current access set functions, $\beta(S) \epsilon (O \times A)^S$ where O, A, and S are the set of objects, access modes and subjects as previously defined. β is used to map a specific subject, S_i, to a set of valid object-access mode tuples, (O_j, x), as $\beta(S_i) = (O_1, x_1), (O_2, x_2), \ldots (O_j, x_k), \ldots\}$ where subject i has access to object j in mode x_j and $S_i \epsilon S$, $O_j \epsilon O$, $x_{jk} \epsilon A$. This function determines the modes of access of all objects currently accessed by a specified subject,S_i. Formally, β is the current access set function if and only if the subject-object-access mode tuple (S_i, O_j, x) is an element of the current access set,b, then the tuple $(O_j x) \epsilon \beta(S_i)$: $(S_i, O_j x) \epsilon b \rightarrow (O_j, x) \epsilon \beta(S_i)$.

This function will be of use in Section A.4.2, Theorems.

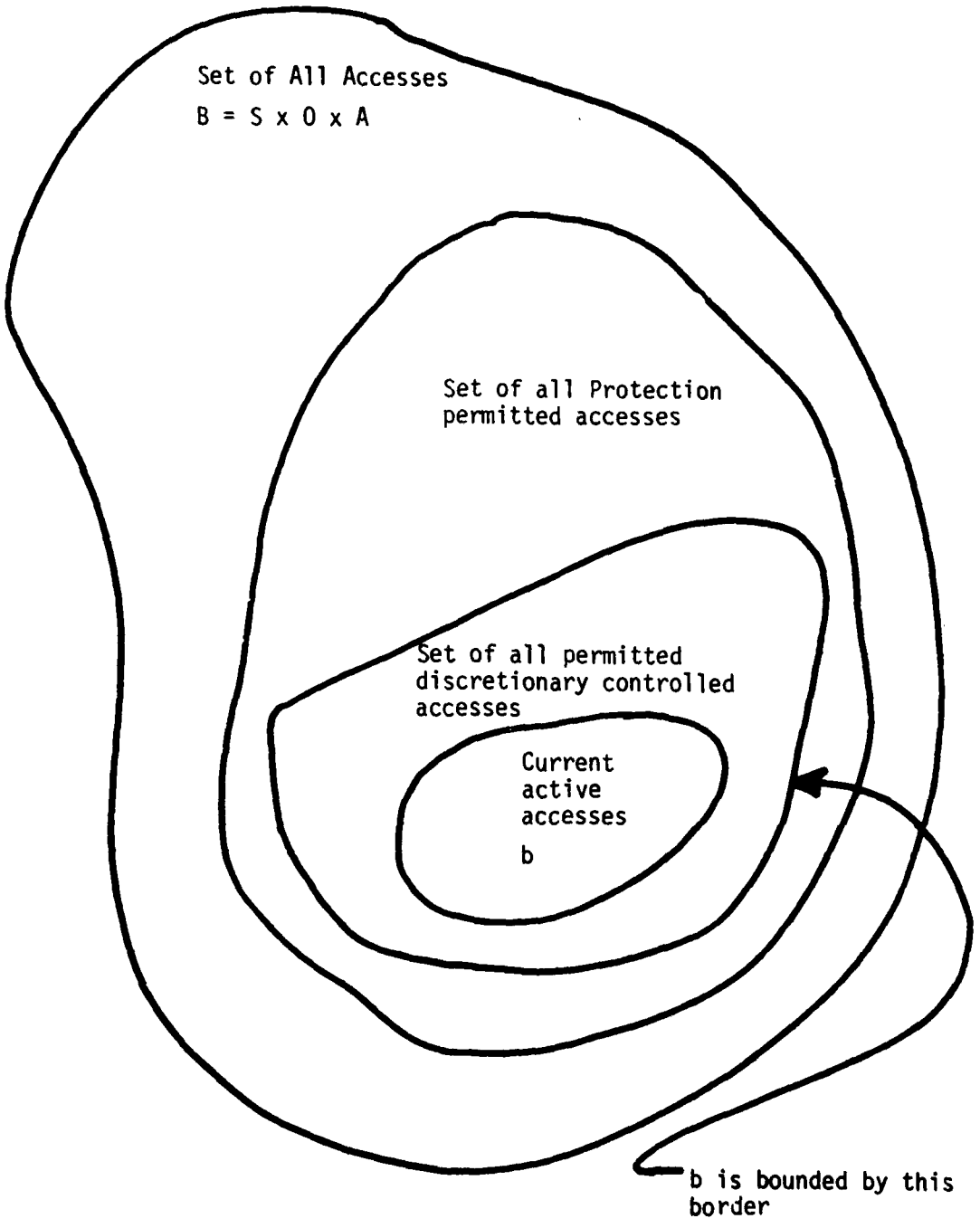


Figure A-2 The Current Access Set b

A. 3.3 Protection Levels: P - Non-discretionary Control

Introduction

The protection requirements of this model enforce two forms of security: discretionary and non-discretionary. This section covers the general principles, the reasoning and definition of protection, and the enforcement of non-discretionary access. Protection is an expanded form of the common practice of security classification and information categorization for purposes of controlled access of information by individuals with the required need to know and security clearances.

In most cases where the controlled access of secure defense data is required, the information is characterized in two ways. First, a classification is given based on the sensitivity of the information with regards to the defense of the United States. These classifications are typically Top Secret, Secret, Confidential and Unclassified, in order of descending sensitivity. In this model these elements will be grouped into a set, C_s, called the security classification set. An element of this set will be denoted by c_s; formally:

$$c_s \in C_s = \{TS,S,C,U\}$$

Categorization is the second type of information characterization. Although an individual may have the proper clearances for access to a particular piece of information, he may not need the access. Therefore a formal set of "need to know" categories are used as a second limitation on nondiscretionary access. These categories are elements of a set, K_s, called the security access category set. Subsets of this set, denoted k_s, will be used as a second characterization of protection in this model. Formally these subsets are elements of a set of all possible subsets of K_s or:

$$k_s \in P(K_s)$$

These characterizations of classification and security category form a security level similar to those given in most installations with controlled access to secure information.

The Grohn model extends data protection beyond security categorization to include integrity. Integrity is a static property of a dynamic system, which requires that the initial soundness of a system be maintained throughout its activities. Non-discretionary security is sufficient to control the access of information by observation but since modification is also an allowed form of information access, some non-discretionary constraint must control the data integrity of the modifications. Integrity classifications and a set of integrity categories similar to those of security characterize the level of integrity. An integrity classification indicates the "soundness" of information and is denoted by c_i, an element of the set of possible integrity levels C_i:

$c_i \varepsilon C_i$ Example of C_i: $C_i = (1, 2, 3, \ldots)$

Integrity categories form a set K_i, subsets of which form integrity category sets:

$k_i \varepsilon P(K_i)$ Example of K_i: $K_i = (A, B, C \ldots)$

PROTECTION LEVELS:

The protection level of an element of this model is described by the four characterizations presented: security classification, security category, integrity level, and integrity category. A protection level is an element of the cartesian product of these elements

$$\pi \varepsilon \Pi = C_s \times P(K_s) \times C_i \times P(K_i)$$

We must now define some relations and properties of protection levels that satisfy the system security requirements.

Ordering

To permit comparisons of protection levels, our protection model must provide for an ordering among the protection levels. This ordering is similar to that of security levels where Top Secret is the highest, and Secret, Confidential and Unclassified follow. This ordering can be expressed as: TS > S > C > U

For this model, we order protection levels in a similar manner: Given two protection levels, π_1 and π_2 from set Π, the set of protection levels Π,

where $\pi_1 = (c_s^1, k_s^1, c_i^1, k_i^1)$

$\pi_2 = (c_s^2, k_s^2, c_i^2, k_i^2)$

Then π_1 is said to dominate π_2 (i.e. π_1 is ordered ahead of π_2) if and only if

$$c_s^1 \geq c_3^2 \qquad (1)$$

$$k_s^1 \supseteq k_s^2 \qquad (2)$$

$$c_i^1 \leq c_i^2 \qquad (3)$$

$$k_i^1 \subseteq k_i^2 \qquad (4)$$

Condition (1) requires the security of the dominated level be lower than that of the dominating level. Using the ordering given above for security levels, if c^1 = TS and c_s^2 = S, then π_1 would satisfy the first condition for dominance of π_2 by π_1:

Condition (2) requires that the set of security categories of the dominated level be contained (i.e. be a subset of) the set of security categories of the dominating level.

Conditions (3) and (4) apply to integrity and are just the reverse of the corresponding conditions on security. The integrity of the dominating level must be less than or equal to that of the dominated level for Condition (3). Condition (4) requires the integrity categories of the dominating level be a subset of the integrity category set for the dominated level.

These conditions form a partial ordering on the set of security levels. Grohn[2] gives an adequate proof of this partial ordering. For this reason we do not repeat this proof in the next section.

The symbol $\succ$ denotes the dominance relationship and is read "dominates".

Protection Functions:

We must now define a function to map elements of the system (subjects and objects) into protection levels. This function will be denoted by P_x, where x can be s,o, or c corresponding to maximum subject protection, object protection, or current subject protection level mappings. These are defined as:

$$p_s \in P_s = \Pi^s = (C_s \times P(K_s) \times C_i \times P(K_i))^s$$

$$p_o \in P_o = \Pi^o = (C_s \times P(K_s) \times C_i \times P(K_i))^o$$

$$p_c \in P_c = \Pi^s = (C_s \times P(K_s) \times C_i \times P(K_i))^s.$$

The cross product of these functions form the system protection level function,P, the third element in the system state:

$$P = (p_s, p_o, p_c) = P_s \times P_o \times P_c$$

Properties of Protection

Protection levels characterize the element of this model. In this model, we define a secure access as one which relates the protection level of the subject (the accessor) to the protection level of the object (the accessed) through the three properties. These three properties provide for non-discretionary protection: the simple protection property, the *-protection property (pronounced "star"-protection property), and the tranquility property.

Simple Protection:

The simple protection property applies to observe type accesses where observe is either read or write: OBS $\in \{\underline{r}, \underline{w}\}$. The simple protection property holds for a given access of object O by subject S in either $\underline{r}$ or $\underline{w}$ made if and only if the current protection of the subject $p_c(S)$, dominates the protection level of the object, $p_o(O)$:

$$p_c(S) \succ p_o(O).$$

Note that the simple protection property applies only to observe (read or write) type accesses and this prevents subjects from observing objects which have higher security levels or lower integrity levels.

*-Protection Property

The protection model constrains modification accesses by the *-protection property. When subject S is accessing object O in either write or append mode, the *-protection property is satisfied if and only if the protection level of the object, $p_o(O)$, dominates the current protection level of the subject, $p_c(S)$:

$$p_o(o) \succ p_c(s).$$

The *-protection property applies only to modify (write or append) type accesses and prevents subjects from modifying objects with lower security or higher integrity. This prevents the posssibility of having high security information or lower integrity information written or appended to objects of lower security or higher integrity.

As detailed by Grohn[4], the satisfaction of the simple protection and *-properties form a required ordering by protection level of the objects simultaneously accessed by a given subject. Given S as a subject and O_1,O_2 and O_3 as three objects accessed by S in append, write and read modes respectively it is required that:

$p_c(S) \succ p_o(O_3)$ by simple protection,
$p_c(S) \succ p_o(O_2)$ by simple protection,
$p_o(O_2) \succ p_c(s)$ by *-property, and
$p_o(o_1) \succ p_c(s)$ by * property.

Therefore: $p_o(O_1) \succ p_o(O_2) \succ p_o(O_3)$.

It is shown that 1) append accessed objects must dominate read and write accessed objects; 2) write accessed objects must dominate read accessed objects and must be at a protection level equal to that of the accessing subject; and 3) read accessed objects must be dominated by the accessing subject.

Tranquility Property

The third protection property is called the tranquility property and it states that the protection level of an object never changes An extension of this property requires the current protection levels of all subjects be a constant or:

and $p_o(O)$ = constant for all objects
$p_c(S)$ = Constant for all subjects.

This property prevents the use of the classification level of an object as a covert communication path. In addition, the downgrading of information through alteration of the subject protection is prevented in this model -- only the trusted subject can downgrade the protection level of an object. Section A.3.6 describes the characteristics of the trusted subject.

The system which maintains these three properties (simple) protection, *-protection, and tranquility) satisfies the requirements for non-discretionary security.

A.3.4 Access Permission Matrix: M_{ij}-Discretionary Access Control

Protection, as described above, restricts subsets of B permitted to exist as the current access set, b. (See Figure A.3-1). In addition to protection which enforces a DoD security policy and a data quality integrity policy, the subject-object-access tuples of the current access set are further restricted by the access permission matrix, M, which provides the discretionary access control in the model. The access permission matrix consists of a table of access permissions as a function of subject object pairs and can be viewed as a table as illustrated in Figure A-3.. In this table, all valid subjects from set S appear on the rows and all objects within the system from set O appear on the columns. The matrix entry at the intersection of the j^{th} column and the i^{th} row is an element from the access set, A, defining the valid access modes for the i^{th} subject to the j^{th} subject. As such, the entries for M_{ij} are mathematically equal to $M_{ij} \subseteq P(A)$. That is, M_{ij} is a subset of the set of all subsets of accesses in set A.

	O_1	O_2 ...	... O_{m-1}	O_m
S_1	r,e	r,e	r	e
S_2	r,e,	r	r,w,e	ø
⋮	⋮	⋮		
S_{n-1}	r,w,e	a,r	r,w,e	e
S_n	e	r,w,e	r,e	r,w,e

FIGURE A-3 EXAMPLE ACCESS CONTROL MATRIX

Also M_{ij} forms a mapping function from subject-object pairs to valid accesses:

$$f(S_i,O_j) = x \text{ where } X=P(A)$$

Conversely stated, if the tuple $(S_i,O_j, \underline{x})$ is an element of the current access set b, then access $\underline{x}$ must have been recorded in M at location i, j:

$$(S_i,O_j,\underline{x}) \in b \quad \text{iff } \underline{x} \in M_{ij} \text{ where } \underline{x} \in X=P(A)$$

Entries into the access control matrix are protected through the same mechanisms of any protected object. To add or remove accesses from the matrix first requires that the subject be given the corresponding discretionary permission and that the subject be at the required protection level.

A.3.5 Data Base Structure: F

The fourth element in the state model vector is the data base structure, F. F specifies the form in which the data is stored and is an element of the set {H,R,C...} where H is hierarchial, R is relational, and C is CODASYL. It is not the purpose of this study to detail the characteristics of these or any other data base form. The important concept here, however, is that the protection provided by this model is independent of the type of data structure used in the data base. It is the implementation of the secure DBMS which most greatly impacts the math model.

The data base structure and the system configuration do require special considerations in the specification of the data base access utilities. The type data base and the system configuration may require that special interface handlers and directory management software be written, for example.

The relational view for data base access is used in the model constructed during this study. There are several advantages to use of the relational data base structure, including ease of usage, flexibility and data independence. Figure A-4 presents many of the reasons for use of a relational data base structure.

- **EASE OF USE:** Two dimensional table representation of relations is simple
- **FLEXIBILITY:** Relational Operators such as Project & Join permit relation formation in forms required by a variety of users and applications
- **PRECISION:** Relations are precise in meaning and can be manipulated through mathematics of Relational Algebra or Relational Calculus
- **SECUIRTY:** Security controls can be easily implemented at the relation level
- **RELATABILITY:** Attributes from different sets of tuples or different "files" can be easily related
- **EASE OF IMPLEMENTATION:** Physical storage of "flat files" can be less complex than physical storage of Tree and Plex structures. Use of normalization may be of benefit.
- **DATA INDEPENDENCE:** If the data base is in a normalized form with data independence in the software, data can be restructured and the data base can grow without forcing rewriting of application programs. This feature results in considerable data base Management cost savings.
- **DATA MANIPULATION LANGUAGE:** Data Manipulation Sublanguage can be based upon relational algebra or relational calculus.
- **CLARITY:** Vast tree diagrams depicting the logical organization of the data base are never necessary. Lack of use of pointers-to-pointers-to-pointers reduces the logical complexity of the data base and permits easier expansion as users needs grow.

Figure A-4 Advantages of a Relational Data Base

A.3.6 Trusted Entities

There are two entities within this data base model which are not constrained by the protection measures presented for the model above. Both entities can be termed trusted subjects. The first set of trusted subjects is the collection of data base security enforcing hardware and software. The second group of trusted subjects are those individuals entrusted with the management of the data base.

Both the hardware and the software of all data protection processors must be trusted, lest the protection be invalid. The hardware used in construction of the security enforcement system must either be accepted to be free of design flaws and trap doors or processed through a validation study. Implementation of the secure DBMS as a distributed architecture removes some of the risk from the hardware: the user of the system does not have direct programming access to the trusted hardware and consequently cannot attempt to exploit any of its possible weaknesses. This fact then places the burden of secure design upon the DBMS architect and the access rule programmers.

All software used in the enforcement processors must be certified to be free of flaws and trap doors and functionally correct. The validity of the software is probably more significant than the validity of the processor utilized because this software provides the interface between the user and the protection system. The software is the embodiment of the security enforcement rules of the DBMS security model of this document. Validation of the software requires proof that all modules implement only the desired operational functions. Development in a secure environment by trusted individuals of high personal integrity is a necessity to ensure that no trap doors or other unintended code is embedded in the process. (See Threats in Appendix B.) Correctness of the software requires detailed checking against the rules specifications to ensure the intended performance. In addition, validation requires actual testing of the procedures operation. Finally, to ensure against tampering with the software and to prevent subjects from creating their own code for execution within the security enforcement processors, all software must be embodied as firmware: software encoded in non-volatile, unalterable read-only memories (ROMS's).

Software certification is a difficult process but extremely critical. One method of certification consists of using only very small software modules (approximately 200 lines of code). Small routines lend themselves to detailed scrutiny and comparison to the intended process. It is necessary that the software be shown to accept only the expected requests and formats and to render only the desired decision based upon the security rule implemented. No other change in any system parameters or system state can be permitted. If the process is sufficiently small, and the system of processes properly defined in terms of any inter-module interaction, the software can be certified..

The second group of trusted subjectsis a single user or a group of users who are trusted and of high integrity who are empowered to by-pass all the protection elements of the secure DBMS. Such trusted users will function as the data base administrators (DBA's). The DBA has several duties which must be executed in order that the data base functions properly. The most obvious of these duties are those of adding users to the set of valid users and setting users maximum protection levels within the system. In addition, due to the characteristics of the protection rules, and in particular, due to the *-property, the only mechanism through which the protection level of an object can be modified is through the explicit action of the DBA. Figure A-5 presents some examples of the responsibilities of the DBA.

PRIMARY ROLES

DATA SECURITY	MONITORS DATA BASE SECURITY FEATURES AND REVIEWS ACCESS LOGS.
DATA BASE MAINTENANCE	ASSIGNS PROTECTION LEVELS TO SUBJECTS. ONLY PERSON WHO CAN REDUCE THE PROTECTION LEVEL
DATA BASE ACCURACY	DEFINES RULES AND FUNCTIONS TO ENSURE VALIDITY AND CONSISTENCY OF DATA
DATA BASE DESIGN AND DEVELOPMENT	DETERMINES ANY CHANGES TO DATA REQUIRED TO MEET USERS NEEDS
USER ACCESS AND VALIDATION	CREATES AND MAINTAINS USER PROTECTION LEVELS AND VERIFIES THE CONTENTS OF THE ACCESS CONTROL LISTS
STYSTEM MAINTENANCE	GENERATES BACK-UPS AND RESTORES DATA FILES IN EVENT OF DATA LOSS OR DESTRUCTION.

PARTICIPATORY ROLES

SYSTEM IMPLEMENTATION	OVERSEES SYSTEM PERFORMANCE GOALS
INFORMATION STRATEGY PLANNING	DEFINES POLICIES GOVERNING DATA STORAGE AND REMOVAL
PROJECT FEASIBILITY EVALUATION	REVIEWS USERS DATA BASE NEEDS FOR COMPATIBILITY WITH EXISTING CONFIGURATIONS
DATA BASE SOFTWARE SELECTIONS	ASSISTS USERS IN SELECTION OF DATA BASE UTILITIES TO BE USED FOR VARIOUS APPLICATIONS
APPLICATION PROGRAMMING AND DESIGN	RECOMMENDS DATA BASE PROCEDURES TO BE USED IN FORMULATION OF A DATA BASE RELATION.

Figure A-5 Responsibilities of the Data Base Adminstrator

A.3.7 Rules in The Data Base Model

The data base rules are the most important items in the secure DBMS model. Rules are the links between the formal model definition and the data base system: Rules are the data base process embodiment of the security restrictions. The data base designer must define a set of rules for each data base user process in such a manner that the formal data base model can validate the rules.

As described in Section A.3.2 a rule is a function which operates upon a request, R_k, and the present system state, v, to determine a decision, D_m, and a resulting new state v*, as

$$\rho_i(P_k y)=(D_m,v^*)$$

where ρ_i is a particular rule from the entire set of rules within the model, ρ,

R_k is the request, (i.e. get_append_permission),

v is the current system state, v=(b,M,P,F),

D_m is the decision generated by the rule

$D_m \in D = \{?, TRUE, FALSE\}$, and

v* is the new system state resulting from the decision, changing either the current access set, b→b*, the access control matrix, m→m*, or both. The protection level is invariant due to the tranquility principle and F is invariant under a system.

If D_m = TRUE, then v* is some new system state, v*=(b*,M*,P,F); if D_m = FALSE, then the system state does not change; if D_m = ?, the request R_k was not recognized and is considered invalid. Thus, a rule can be viewed as a mapping of a request and an initial secure state to a decision and a resulting secure state.

The rule ρ_i above is one of many valid rules existing in the secure DBMS model. The set of all valid rules is ρ, where $\rho_i \in \rho$.

The set of valid rules is formulated as $\rho \subseteq (DxV)^{(RxV)}$ where

RxV forms all possible request-system state tuples and

DxV forms all possible decision new state tuples.

The power notation (see Section A.2) generates a set of functions $\{\rho_1, \rho_2, \rho_3 \ldots \rho_i, \ldots \rho_n\}$ where n is equal to the number of elements in the set DxV raised to the power equal to the number of elements of set RxV. Because not all rules will permit transitions form a secure initial state to a secure resultant state, ρ is a subset of the set of all possible rules.

The set of valid rules is specified in accordance with the desired user access functions. Figure A-6 presents an example list of user functions for interaction with the data base.

A rule is secure state preserving rule if and only if the resulting state v* is a secure state when $\rho_i(R_k, v) = (D_m, v^*)$ and v is a secure state. Similarly, the secure state preserving rule can be shown also to be simple protection preserving, *-protection preserving, and discretionary protection preserving. Secure state preserving rules are the basis of the security model implementation.

RULE	RULE NAME	DESCRIPTION	STATE ELEMENT AFFECTED
1	get-read	requests read access to object	b
2	get-append	requests append access to object	b
3	get-execute	requests execute access to object	b
4	get-write	requests write access to object	b
5	release	release accesses: read, append, execute, write	b
6	permit	permit another subject discretionary access on object	M,B
7	rescind	remove discretionary access to an object	M,B
8	create	create an object in data base	B,M
9	delete	remove an object from data base	B,M

$b = (S_i, O_j, \underline{x}), \ldots \quad B = SxOxA$

Figure A-6 An Example Set of User Data Base Access Rules

A.4 Security Item Definitions

The following are a set of definitions pertinent to the proofs contained within this Appendix. These definitions, although brief, are presented to reiterate some explanations of model elements developed in Section A.3 and as such, should be considered as a reference to the proofs. For more information see Section A.3

1 State: A state of the system denoted $v=(b,M,P,F)$ is an element of $V \subset B \times M \times P \times F$ and represents the current accesses, the access permissions the subject and object protection levels and the data base form at any time.

2 State Sequence: A state sequence, denoted z, is a mapping of states, elements of V, into time or $z \in Z \subset V^T$ where T is the set of times. z_t is the t_{th} state in the sequence $z: z_t = (b^t, M^t, P^t, F)$.

3 Current Access Function, β: The current access function, $\beta(S_i)$ is an element of the set of current access functions, $\beta(S_i) \in (O \times A)^S$
where O,A,S are the set of all objects, set of all access modes, and set of all subjects, respectively. For a given subject, S_i, β returns the set of object-access mode tuples, $(O_j, \underline{x})$; i.e $\beta(S_i) = \{(O_1, \underline{x}_1), (O_2, \underline{x}_2) \ldots (O_j, \underline{x}_k) \ldots\}$ where subject i has accessed object j in mode $\underline{x}_k$ and $S_i \in S$, $O_j \in O$, and $\underline{x}_k \in A$

4 Simple Protection Property: A state $v=(b,M,P,F)$ satisfies the simple protection property if and only if $p_c(S_i) \rightharpoonup p_o(O_i)$ for every subject S_i, and all $(O_j, \underline{x}) \in \beta(S_i)$, where $\underline{x}$ is $\underline{r}$ or $\underline{w}$ and β is the current access function.

5 *Protection Property: A state $v=(b,M,P,F)$ satisfies the * protection property if and only if $p_o(O_j) \rightharpoonup p_c(S_i)$ for every subject S_i, and for all $(O_j, \underline{x}) \in \beta(S_i)$, where $\underline{x}$ is $\underline{w}$ or $\underline{a}$.

6 Discretionary Protection Property: A state $v=(b,M,P,F)$ satisfies the discretionary protection property if and only if $\underline{X} \in M_{ij}$ for every subject S_i, and for all $(O_j,\underline{x}) \in B(S_i)$.

7 Secure State: A state $v=(b,M,P,F)$ is a secure state if and only if v satisfies the simple protection, *-protection and discretionary protection properties.

8 Secure State Sequence: A state sequence z is a secure state sequence if and only if for every z_t, $t \in T$, z_t is a secure state.

9 Secure System: A secure system is one which is described by a secure state sequence.

10 Rule: A rule is a functional mapping of request-state pair into decision-state pairs. A rule $\rho \in (Dxv)^{(Rxv)}$.

11 Secure State Preserving Rule: A rule ρ_k is secure state preserving *if and only if* v^* *is a* secure state whenever $\rho_k(R_i,v)=(D_m,v^*)$ and v is a secure state. Secure state preserving rules preserve the simple protection property, the *-property and the discretionary property.

12 Action of a Rule: An action of a rule is an element of the set $W=(R_k,D_m,v^*,v)$. The element W is the action of a rule ρ_i if $\rho_i(R_k,v)=(D_m,v^*)$.

13 Rule Limited Action Set: A rule limited action set, $W(\rho) \subseteq W$, is defined over a set of rules, $\rho=\{\rho_1,\rho_2,\ldots\rho_n\}$, as $(R_k,D_m,v^*,v) \in W(\rho)$ iff

(i) $D_m \neq \underline{?}$ and

(ii) $(D_m,v^*)=\rho_i(R_k,v)$ for a unique i, $1 \leq i \leq n$.

condition i) requires that the request be a valid request rule.

condition ii) requires that for every rule and request R_i, there be a decision, D_m: $D_m \in$ {True, False}

A.5 Theorems

This next section contains a set of theorems and associated proofs necessary for the support of the validation of a set of rules necessary for data base operation. These rules, in actual implementation, support two functions, From the data base users point of view, the rules provide him with access to objects in the data base. From the security point of view, the rules must enforce the requirements of protection and thus be protection preserving rules.

A.5.1 Mathematical Proofs For a Secure DBMS

This section presents a set of theorems and proofs formally describing the requirements of a Secure DBMS. These proofs are presented in four major groupings:

I. Requirements of Actions for Security Preservation

II. Properties of Secure State Preserving Rules

III. Requirements for Secure State Preservation During Additions to the Current Access Set.

IV. Security preservation during actions of rules which do not add access, do not change protection or do not remove access permissions.

Group I Theorems (Theorems 1-3) present the conditions necessary to restrict the set of actions, w, to those actions which maintain a secure system.

Group II Theorems (Theorems 4-6) prove that an action set limited by a set of secure state preserving rules forces the system to be a secure system. As a result, these three theorems prove that security enforcement lies in the design and implementation of the rules themselves.

The Group III Theorems (Theorems 7-9) present the set of conditions which must be met to maintain a secure state under additions of subject, object, access tuples to the current access set.

The Group IV theorems present the conditions for the actions of a rule to be secure state preserving in the special cases where protection levels do not change, accesses are not removed, and permissions are not added.

GROUP I THEOREMS: REQUIREMENTS OF ACTIONS FOR SECURE SYSTEM PRESENTATION

THEOREM 1:

This theorem proves the completeness of the constraint conditions on the set of allowable actions which will maintain a system which satisfies the simple protection property.

Statement

The system satisfies the simple protection property for any initial state, z_0, which satisfies the simple protection property if and only if W, the set of actions, satisfies the folloiwng conditions for each action, $(R_k, D_m(b^*,M^*,P^*,F^*), (b,M,P,F,))$:

(i) for each $(S,O,\underline{x}) \in b^*-b$ and $\underline{x} = \underline{r}$ or $\underline{w}$ then $p_c^*(S) \rightarrowtail p_o^*(O)$

(ii) for each $(S,O,\underline{x}) \in b$ and $\underline{x} = \underline{r}$ or $\underline{w}$ and $p_c^*(S) \not\rightarrowtail p_o^*(O)$ then $(S,O,\underline{x}) \notin b^*$

PROOF

For the system to satisfy the simple protection property, every state in the state sequence must satisy this property, key definitions 8 and 9.

Given $z_0=(b^0,M^0,P^0,F^0)$, a state satisfying the simple protection property, at time O, we must show that $z_t= (b^t,M^t, P^t, F^t)$ is a simple protection satisfying state for any time $t \in T$.

First, we will show that $z = (b^1,M^1,P^1,F^1)$ satisfies the simple protection property. As the result of the state transition creating b^1, either in b^1-b^0 or $b^1 \cap b^0$. Elements in b^1-b^0 satisfy the simple protection property due to condition (i). Elements in $b^1 \cap b^0$ satisfy the simple protection property due to condition (ii).

Secondly, we must show that for any $z_{t-1}=(b^{t-1}, M^{t-1}, P^{t-1}, F^{t-1})$ which satisfies the simple protection property, z_t satisfies it also under conditions (i) and (ii). Similar to the previous argument, all elements of access set b^t are in either $b^t - b^{t-1}$ or $b^t \cap b^{t-1}$. Elements in $b^t - b^{t-1}$ satisfy the simple protection property due to condition (i). Elements in $b^t \cap b^{t-1}$ satisty the simple protection property due to (ii).

Since z satisfies the simple protection property for a secure initial state z_0 and z^t satisfies the simple protection property for a previous secure state z^{t-1} under conditions (i) and (ii) on W, then Z, the sequence of system states, is a simple protection preserving state sequence, and the system is a simple protection preserving system.

Finally we must prove the "only if" condition. Suppose the system satisfies the simple protection property and has an initial state z_0 which also satisfies the simple protection. Assume that the action $(R_k, D_m, (b^t, M^t, P^t, F^t), (b^{t-1}, M^{t-1}, P^{t-1}, F^{t-1}))$ exists such that either:

(iii) there exists some $(S,O,\underline{x}) \in b^t - b^{t-1}$ where $\underline{x}=\underline{r}$ or $\underline{w}$ such that

$$p_c^t(S) \not\vdash p_o^t(O) \qquad \text{or}$$

(iv) there exists some $(S,O,\underline{x}) \in b^{t-1}$ where $\underline{x}=\underline{r}$ or $\underline{w}$ such that

$$p_c^t(S) \not\vdash p_o^t(O) \text{ and } (S,O,\underline{x}) \in b^{t-1} \cap b^t.$$

Then there is some $(S,O,\underline{x})$ in $b^t - b^{t-1}$ which does not satisfy the simple protection property relative to P^t (by iii) or there is some $(S,O,\underline{x})$ in $b^{t-1} \cap b^t$ which does not satisfy the simple protection property relative to P^t (by iv). Since $b^t = (b^t - b^{t-1}) \cup (b^{t-1} \cap b^t)$ then there is some $(S,O,\underline{x}) \in b^t$ which does not satisfy the simple protection property relative to P^t. Therefore, z_t does not satisfy the simple protection property, and the system state sequence does not satisfy the simple protection property. Consequently, the system does not preserve simple protection property which contradicts our supposition and completes our proof by contradiction.

THEOREM 2:

This theorem proves the completeness of the constraint conditions on the set of allowable actions which maintain a system which satisfies the *-protection property.

Statement

The system satisfies the *-protection property for any initial state z_o which satisfies the *-protection property if and only if the set of actions, W, satisfies the following conditions for each action $(R_k, D_m, (b^*,M^*,P^*,F^*),(b,M,P,F))$.

(i) for each $(S,O,\underline{x}) \in b^*-b$ and $\underline{x} = \underline{w}$ or $\underline{a}$ then

$p_o^*(O) \rightharpoonup p_c^*(S)$

(ii) for each $(S,O,\underline{x}) \in b$ and $\underline{x}=\underline{w}$ or $\underline{a}$ and $p_o^*(O) \not\rightharpoonup p_c^*(S)$ then $(S,O,\underline{x}), \notin b^*$

Proof

For the system to satisfy the *-protection property, every state in the state sequence must satisfy this property by definitions 8 and 9.

Given $z_o=(b^o,M^o,P^o,F^o)$, a state satisfying the *-protection property, at time zero, we must show that the system state at time $t, z_t=(b^t,M^t,P^t,F^t)$, is a *-protection property satisfying state for any time $t \in T$.

To do this, we first show that $z_1=(b^1,M^1,P^1,F^1)$ satisfies the *-protection property. All elements of b^1 are either in b^1-b^0 or $b^1 \cap b^0$. Elements in $b^1 \cap b^0$ satisfy the property by (ii). Elements in b^1-b^0 satisfy the property under condition (i).

Next, we show that if some general time (t-1) the system state $z_{t-1}=(b^{t-1},M^{t-1},P^{t-1},F^{t-1})$ which satisfies the *-protection property then z_t satisfies it also under (i) and (ii). All elements of b^t are either in b^t-b^{t-1} or $b^t \cap b^{t-1}$. Elements in b^t-b^{t-1} satisfy the property under condition (i).

Therefore, for any time $t \in T$, the state z_t satisfies the *-protection property under the restricitons on the set of actions W

in (i) and (ii). So z is a *-protection preserving sequence and thus the system is also.

Finally, we must prove the "only if" condition. Suppose the system satisfies the *-protection property and has an initial state z_0 which also satisfies the *-protection property. Assume there is some action $(R_k, D_m, (b^t, M^t, P^t, F^t), (^{t-1}, M^{t-1}, P^{t-1}, F^{t-1}))$ such that either:

(iii) there exists some $(S,0,\underline{x}) \in b^t - b^{t-1}$ where $\underline{x}=\underline{w}$ or $\underline{a}$ such that $p_0^t(0) \not\subset p_c^t(S)$ or

(iv) there exists some $(S,0,\underline{x}) \in b^{t-1}$ where $\underline{x} = \underline{w}$ or $\underline{a}$ such that $p_0^t(0) \not\subset p_0^t(S)$ and $(S,0,\underline{x}) \in b^{t-1} \cap b^t$.

By our assumption on the action, there is some $(S,0\ \underline{x})$ in $b^t - b^{t-1}$ which does not satisfy the *-protection property relative to P^t (by iii) or there is some $(S,0,\underline{x})$ in $b^{t-1} \cap b^t$ which does not satisfy the *-protection property relative to P^t (by iv). Since $b^t = (b^t - b^{t-1}) \cup (b^{t-1} \cap b^t)$, then there is some $(S,0,\underline{x}) \in b^t$ which does not satisfy the *-protection property relative to P^t. Therefore z_t does not satisfy the *-protection property and z does not satisfy the property. Consequently, the system is not *-protection property preserving which contradicts our supposition and completes our proof by contradiction.

THEOREM 3:

This theorem proves the completeness of the constraint conditions on the set W of allowable actions which will maintain a system which satisfies the discretionary protection property.

Statement

The system satisfies the discretionary protection property for any initial state z_o which satisfies the descretionary protection property if and only if W, the set of actions, satisfies the following conditions for each action $(R_k, D_m, (b^*M^*, P^*, F^*), (b, M, P, F))$:

(i) for each S_i and every $(O_j, \underline{x}) \in \beta^*(S_i) - \beta(S_i)$ then $\underline{x} \in M^*_{ij}$

(ii) for each S_i and every $(O_j, \underline{x}) \in \beta(S_i)$ if $\underline{x} \notin M^*_{ij}$ then $(S_i, O_j, \underline{x}) \notin b^*$

PROOF

For the system to satisfy the discretionary protection property every state in the state sequence must satisfy this property by definitions 8 and 9.

Given $z_o = (b^0, M^0, P^0, F^0)$, a state satisfying the discretionary protection property at time 0, we must show that $z_t = (b^t, M^t, P^t, F^t)$ is a discretionary protection property satisfying state at any time t.

First, we show that $z_1 = (b^1, M^1, P^1, F^1)$ satisfies the simple protection property. All elements of b^1 are either in $b^1 - b^0$ or $b^1 \cap b^0$. Elements in $b^1 - b^0$ satisfy the discretionary protection property due to condition(i). Elements in $b^1 \cap b^0$ satisfy the property due to condition (ii).

Next, we show that for any state $z_{t-1} = (b^{t-1}, M^{t-1}, P^{t-1}, F^{t-1})$ which satisfies the discretionary protection property z_t satisfies it also under conditions (i) and (ii). Again, all elements of b^t are either in $b^t - b^{t-1}$ or $b^t \cap b^{t-1}$. Elements in $b^t - b^{t-1}$ satisfy the discretionary protection property due to condition (i). Elements in $b^t \cap b^{t-1}$ satisfy the property due to condition (ii).

Then z is a discretionary protection preserving state sequence and the system is discretion protection preserving.

Finally, we must prove the "only if" condition. Suppose the system satisfies the discretionary protection property and has an initial state z_o which also satisfies the property. Assume there is some action $(R_k, D_m, (b^t, M^t, P^t F^t), (b^{t-1}, M^{t-1}, P^{t-1}, F^{t-1}))$ such that either:

(iii) there exists some $(S,O,\underline{x}) \in b^t - b^{t-1}$ and $\underline{x} \notin M^*_{ij}$ or

(iv) there exists some $(S,O,\underline{x}) \in b^{t-1}$ such that $\underline{x} \notin M^t_{ij}$ and $(S,O,\underline{x}) \in b^{t-1} \cap b^t$.

Then there is some $(S,O,\underline{x})$ in $b^t - b^{t-1}$ which does not satisfy the discretionary protection property (by iii) or there is some $(S,O,\underline{x})$ in $b^{t-1} \cap b^t$ which does not satisfy the discretionary protection property (by iv). Since $b^t = (b^t - b^{t-1}) \cup (b^{t-1} \cap b^t)$, then there is some $(S,O,\underline{x}) \in b^t$ which does not satisfy the discretionary protection property at time t.

Thus, z_t does not satisfy the discretionary protection property and therefore z does not satisfy the property. Consequently the system is not discretionary property preserving which contradicts our supposition and completes our proof by contradiction.

COROLLARY 1:

The six conditions presented are sufficient as proved in Theorems 1, 2, and 3 to restrict the set of actions, W, to those actions which would maintain a secure system, i.e., a system which satisfies the simple protection property, the *protection property and the discretionary protection property.

Statement

The system is a secure system if and only if z_o, the initial state, is a secure state and the set of actions,W, satisfies the conditions of theorems 1, 2, and 3 or:

(i) for each $(S,O,\underline{x}) \in b^*-b$ and $\underline{x}=\underline{r}$ or $\underline{w}$ then $p_c^*(s) \rightsquigarrow p_o^*(o)$

(ii) for each $(S,O,\underline{x}) \in b$ and $\underline{x}=\underline{r}$ or $\underline{w}$ and $p_c^*(s) \rightsquigarrow p_o^*(o)$ then $(S,O,\underline{x}) \notin b^*$

(iii) for each $(S,O,\underline{x}) \in b^*-b$ and $\underline{x}=\underline{w}$ or $\underline{a}$ then $p_o^*(o) \rightsquigarrow p_c^*(s)$

(iv) for each $(S,O,\underline{x}) \in b$ and $\underline{x} = \underline{w}$ or $\underline{a}$ and $p_o^*(o) \rightsquigarrow p_c^*(s)$ then $(S,O,\underline{x}) \in b^*$

(v) for each $(S,O,\underline{x}) \in b^*-b$ then $\underline{x} \in M_{ij}$

(vi) for each $(S,O,\underline{x}) \in b$ and $\underline{x} \notin M_{ij}$ then $(S,O,\underline{x}) \in b^*$

Items i and ii above are necessary and sufficient for the simple protection property.

Items iii and iv are necessary and sufficient for the *-protection property.

Items v and vi are necessary and sufficient conditions for discretionary protection.

GROUP II LIMITATIONS OF ACTIONS

THEOREM 4:

Theorem 4 proves that under a set of actions limited by the operation of a specific set of rules which in this case maintain the simple protection property, the system also maintains the simple protection property. An important result of this proof is the realization that the simple protection property can be enforced through the specification of the set of valid rules permitted on the system.

Statement

Given ρ, a set of simple protection property preserving rules, and z_0, the initial state which also satisfies this property, then the system satisfies the simple protection property under $W(\rho)$, a rule limited action set.

Proof

We will prove this theorem by contradiction.

Suppose the system does not satisfy the simple protection property. Then there is some state z_t in the state sequence z which is not a simple protection preserving state. Since z_0 is simple protection property preserving, t must be greater than t_0. Choose z_{t-1} such that it is a simple protection property preserving state and z_{t-1} differs from z_t. By the definition of the rule limited action set $W(\rho)$there is some rule ρ_i in ρ such that $\rho_i(R_k, z_{t-1}) = (D_m, z_t)$. Because z_{t-1} is a simple protection preserving state and ρ_i, by definition, is a simple protection preserving rule, then z_t is a simple protection property preserving state. This fact contradicts the initial assumption and compltees the argument.

THEOREM 5:

Theorem 5 proves that under a set of actions limited by the operation of a specific set of *-protection property rules, the system maintains the *-property. An important result of this proof is the realization that the *-protection property can be enforced through the specification of the rules allowed on the system.

Statement

Given ρ, a set of *-protection property preserving rules and z_0, the initial state, which also satisfies the *-protection property, then the system satisfies the *-protection property under $W(\rho)$, a rule limited action set.

Proof

We will prove the theorem contradiction. This proof directly parallels that of Theorem 4.

Suppose the system does not satisfy the *-protection property. Then there is some state z_t in the state sequence z which is not a *-protection property preserving state. Since z_0 is a *-protection property preserving state, then t must be greater than t_0. Choose a state z_{t-1} such that z_{t-1} is a *-protection preserving state and z_{t-1} differs from z_t. By the definition of the rule limited action set $W(\rho)$ there is some rule ρ_i in ρ such that $\rho_i(R_k, z_{t-1}) = (D_m, z_t)$. Because z_{t-1} is a *-protection preserving state and the rule ρ_i, by definition, is a *-protection property preserving rule, then z_t is a *-protection property satisfying state. This contracts the initial assumption and completes the argument.

THEOREM 6:

Theorem 6 proves that under a set of actions limited by the operations of a specific set of discretionary protection rules which maintain discretionary protection, the system maintain the property. An important result is the realization that discretionary protection can be enforced through the specification of the rules allowed on the system.

Statement

Given ρ, a set of discretionary protection property preserving rules, and z_0, the initial discretionary protection property preserving state. Then the system satisfies the discretionary protection property under $W(\rho)$, a rule limited action set.

Proof

Again, this system does not satisfy the discretionary protection property. Then there is some state z_t in the state sequence z which is not a discretionary protection property preserving state. Since z_0 is discretionary protection property satisfying, then $t > t_0$. Choose z_{t-1} such that z_{t-1} satisfies the discretionary protection property and z_{t-1} differs from z_t. By the definition of the rule limited action set $W(\rho)$ there is some rule ρ_i in ρ such that $\rho_i(R_k, z_{t-1}) = (D_m, z_t)$. Because z_{t-1} is a discretionary protection property preserving state and ρ_i, by definition, is a discretionary protection preserving rule, then z_t is a discretionary protection preserving state. This contradicts our assumption.

COROLLARY 2:

This result follows naturally from theorems 4, 5, and 6 which prove that an action set limited by a set of secure state preserving rules forces the system to be secure. Therefore, security enforcement lies in the design and implementation of the rules.

Statement

Given ρ, a set of secure state preserving rules, and z_0, the initial state, which is a secure state, then the system is a secure system under $W(\rho)$, a rule limited action set.

GROUP III REQUIREMENTS FOR SECURE STATE PRESENTATION DURING ADDITIONS TO THE CURRENT ACCESS SET

THEOREM 7:

This theorem presents the conditions necessary for the satisfaction of the simple protection property under additions of subject, object, access tuples to the current access set b.

Statement

Let the following be given: v=(b,M,P,F), a state of the system which satisfies the simple protection property; $(S,O,\underline{x}) \notin b$; $b^* = b \cup \{(S,O,\underline{x})\}$; and $v^* = (b^*,M,P,F)$. Then v* satisfies the simple protection property if and only if:

(i) $\underline{x}=\underline{e}$ or $\underline{a}$

or (ii) $\underline{x}=\underline{r}$ or $\underline{w}$ and $p_s(S) \leq p_o(O)$

Proof

First, we must prove the "if" section of this statement. *Assume for* $b^* = b \cup \{(S,O,\underline{x})\}$ *that* $x=\underline{e}$ *or* $\underline{a}$ then v* satisfies the property since v does. Assume for $b^* = b \cup \{(S,O,\underline{x})\}$ that $x = \underline{r}$ or $\underline{w}$. Then by the definition of the simple protection property (Def. 4) v* satisfies the simple protection property.

Next, we must prove the "only if" clause. Given $v^*=(b^*,M,P,F)$ where $b^* = b \cup \{(S,O,\underline{x})\}$ and v and v* are simple protection satisfying states. Therefore if $x = \underline{r}$ or $\underline{w}$ then $p_s(S) \leq p_o(O)$. Since $\underline{x} \in \{\underline{e},\underline{a},\underline{r},\underline{w}\}$ *then either (i) or (ii) is satisfied and so the* statement is proven.

THEOREM 8:

This presents the conditions necessary for the satisfaction of the *-protection property under additions of subject, object, access tuples to the current access set b.

Statement

Let the following be given: $v=(b,M,P,F)$, a state of the system which satisfies the *-protection protection; $(S,O,\underline{x}) \in b$; $b^* = b \cup \{(S,O,\underline{x})\}$, and $v^* = (b^*,M,P,F)$. Then v^* satisfies the *-protection property if and only if:

(i) $x=\underline{r}$ or $\underline{e}$

(ii) $x=\underline{w}$ or $\underline{a}$ and $p_O(O) \rightarrowtail p_S(S)$

Proof

First we must prove the "if" section of the statement. Assume for $b^* = b \cup \{(S,O,\underline{x})\}$ that $\underline{x}=\underline{r}$ or $\underline{e}$ then v^* satisfies the *-protection property since v does. Secondly, assume for $b^* = b \cup \{(S,O,\underline{x})\}$ that $\underline{x} = \underline{w}$ or $\underline{a}$. Then, by the definition of the *-proection property (Def 5), v^* satisfies the *-protection property because (1) $p_O(O) \rightarrowtail p_S(S)$ and (2) v satisfied the *-protection property.

Next, we must prove the "only if" clause. Let it be given that $v^* = (b^*,M,P,F)$ where $b^* = b \cup \{(S,O,\underline{x})\}$ and, v and v^* are states which satisfy the *-protection property. If $\underline{x} = \underline{w}$ or $\underline{a}$ then $p_O(O) \rightarrowtail p_S(S)$, by Def 5.

Since $\underline{x} \in \{\underline{e},\underline{a},\underline{r},\underline{w}\}$ then either (i) or (ii) is satisfied and the statement is proven.

THEOREM 9:

This theorem presents the conditions necessary for the satisfaction of the discretionary-protection property under additions of subject, object, access tuples to the current access set b.

Statement

Let the following be given: v = (b,M,P,F), a state of the system which satisfies the discretionary protection property ; $(S_i, O_j, \underline{x}) \notin b$; $b^* = b \cup \{S_i, O_j, \underline{x})\}$;and $v^* = (b^*, M, P, F)$. Then v* satisfies the discretionary-protection property if and only if $\underline{x} \in M_{ij}$.

Proof

First, we must prove the "if" section of this statement. Assume $b^* = b \cup \{(S_i, O_j, \underline{x})\}$and $\underline{x} \in M_{ij}$. Since v satisfied the discretionary security property and $\underline{x} \in M_{ij}$ then v* satisfies the property also.

Next, we must prove the "only if" clause. Assume v* satisfies the discretionary protection property. Then for $b^* = b \cup \{(S_i, O_j, \underline{x})\}$, $\underline{x} \in M_{ij}$ by the definition of discretionary-protection (Def 6). The statement is satisfied and is proven.

COROLLARY 3:

Corollary 3 is a natural connection of the results of Theorems 7, 8 and 9. Therefore, Corollary 3 gives the set of conditions which must be met to maintain a secure state under additions of subject, object, access tuples, $(S_i, O_j, \underline{x})$ to the current access set.

Statement

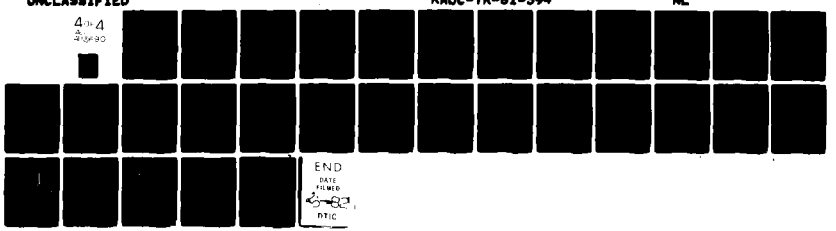
Let the following be given that:

$v = (b, M, P, F)$ is a state of the secure system;
$(S_i, O_j, \underline{x}) \notin b$ is a subject-object-access tuple not in v;
$b^* = b \cup \{(S_i, O_j, \underline{x})\}$ is a new access set; and
$v^* = (b^*, M, P, F)$ is a new system state.

Then v^* is a secure state if and only if:

(i) if $\underline{x} = \underline{e}$
or (ii) if $\underline{x} = \underline{r}$ and $p_s(S_i) \rightharpoonup p_o(O_j)$ (Simple protection)
or (iii) if $\underline{x} = \underline{w}$ and $p_s(S_i) = p_o(O_j)$ (Simple protection, *-protection)
or (iv) if $\underline{x} = \underline{a}$ and $p_o(O_j) \rightharpoonup p_s(S_i)$ (*-Protection)
and $\underline{x} \in M_{ij}$ (Discertionary protection)

AD-A113 690 HARRIS CORP MELBOURNE FL GOVERNMENT ELECTRONIC SYSTE--ETC F/G 9/2
SECURE DBMS.(U)
FEB 82 T D WORMINGTON, C E GIESLER F30602-80-C-0235
UNCLASSIFIED RADC-TR-81-394 NL



GROUP IV: SECURITY PRESERVATION DURING ACTIONS OF RULES WHICH DO NOT ADD ACCESS, DO NOT CHANGE PROTECTION OR DO NOT REMOVE ACCESS PERMISSIONS

THEOREM 10:

Theorem 10 states that if the action of a rule removes some element or elements from the current access set of a simple protection preserving state, then the resultant state also satisfies the simple protection property. Therefore, the rule is one which preserves the simple protection property.

Statement

Let the following be given: ρ_i, a rule, such that $\rho_i(R_k,v) = (D_m,v^*)$; $v = (b,M,P,F)$; and $v^* = (b^*,M^*,P^*,F^*)$. If $b^* \subseteq b$ and $P^* = P$ then ρ_i is a simple protection preserving rule.

Proof

Assume v satisfies the simple-protection property. Then, for every $(S,O,\underline{x})$ b where $\underline{x} = \underline{r}$ or $\underline{w}$, $p_s(S) \rightharpoonup p_o(O)$. Since $b^* \subseteq b$, every $(S,O,\underline{x})$ in b^* is also an element of b. Therefore, $p_s(S) \rightharpoonup p_o(O)$ for every $(S,O,\underline{x}) \in b^*$ where $\underline{x} = \underline{r}$ or $\underline{w}$. Since $P^* = P$ (tranquility principle), $p^*_s(S) = p_s(S)$ and $p^*_o(O) = p_o(O)$. Then for every $(S,O,\underline{x}) \in b^*$ where $\underline{x} = \underline{r}$ or $\underline{w}$, $p^*_s(S) \rightharpoonup p^*_{o(o)}$. Consequently, if v satisfies the simple-protection property, then v* satisfies the simple protection property and thus satisfies the condition for ρ_i to be a simple protection preserving rule.

THEOREM 11:

Theorem 11 states that if the action of a rule removes some element or elements from the current access set of a state which satisfies the *-protection property then the resultant state also satisfies the *-protection property. As a result, the rule is one which preserves the *-protection property.

Statement

Let the following be given: ρ_i is a rule such that $\rho_i(R_k,v) = (D_m,v^*)$; $v=(b,M,P,F)$; and $v^* = (b^*,M^*,P^*,F^*)$. If $b^* \supseteq b$ and $P^*=P$, then ρ_i is a *-protection property preserving rule.

Proof

Assume v satisfies the *-protection property. Then for every $(S,O,\underline{x}) \in b$ where $\underline{x} = \underline{a}$ or $\underline{w}$, $p_O(O) \rightarrowtail p_S(S)$. Now $b^* \subseteq b$ every $(S,O,\underline{x})$ in b^* is also an element of b. Thus, $p_O(O) \rightarrowtail p_S(S)$ for every $(S,O,\underline{x}) \in b^*$ where $\underline{x} = \underline{a}$ or $\underline{w}$. Since $P^* = P$ as the result of the tranquility principle, then $p^*_S(S) = p_S(S)$ and $p^*_O(O) = p_O(O)$ and for every $(S,O,\underline{x}) \in b^*$ where $\underline{x} = \underline{a}$ or $\underline{w}$, $p^*_O(O) \rightarrowtail p^*_S(S)$. Therefore, given that v satisfies the *-protection property, then v* satisfies the *-protection property which satisfies the conditions for ρ to be a *-protection preserving rule.

THEOREM 12:

Theorem 12 states that if the action of a rule adds no elements to the current access set (b) and removes no accesses from the current access matrix,(M), from a state which satisfies the discretionary protection property, then the resultant state satisfies the property. As a result, the rule is one which preserves the discretionary protection property.

Statement

Let the following be given: ρ_i is a rule such that $\rho_i(R_k,v)=(D_m,v^*)$; $v=(b,M,P,F)$; and $v^* = (b^*,M^*,P^*,F^*)$. If $b^* \subseteq b$ and $M^*_{ij} \supseteq M_{ij}$ for every i,j, then ρ_i is a discretionary-protection preserving rule.

Proof

Assume v satisfies the discretionary protection property. Then, for every $(S_i,O_j,\underline{x}) \in b$, $\underline{x} \in M_{ij}$. Since $b^* \subseteq b$, every $(S,O,\underline{x})$ in b* is also an element of b. Thus, $\underline{x} \in M_{ij}$ for every $(S_i,O_j,\underline{x}) \in b^*$. Since $\underline{x} \in M_{ij}$ and $M_{ij} \subseteq M^*_{ij}$ then $\underline{x} \in M^*_{ij}$. Therefore, given that v satisfies the discretionary-protection property v*, also satisfies the discretionary protection property which satisfies the condition for ρ_i to be a discretionary-protection preserving rule.

COROLLARY 4:

Statement

The Group IV Theorems and Corollary 4 form the basis for the support of the conditions for the actions of a rule to be secure state preserving. These conditions apply to the special case where the protection levels do not change, accesses are not removed and permissions are not added.

Let it be given that;
ρ_i is a rule such that $\rho_i(R_k,v)=(D_m,v^*)$;
$v=(b,M,P,F)$ is a current system state;
$v^*=(b^*,M^*,P^*,F^*)$ is a new system state.

Then if $b^* \subseteq b$ is the new access set,
$P^*=P$ is the new protection level, and
$M^*_{ij} \supseteq M_{ij}$ is the new Access Matrix,

Then ρ_i is a secure state preserving rule

APPENDIX B

SYSTEM THREATS

B.0 INTRODUCTION

This section discusses general computer threats. Most of these threats are present for all computer installations and therefore are not unique to a secure system.

This section is presented in support of the threat discussions in Section 2.3 of this report. The following subsections follow:

B.1 Computer System Threats
B.2 Physical Threats
B.3 System Threats
B.4 Data Threats
B.5 Security Threats

B.1 Computer System Threats

Threats to a computer system come from a variety of sources. These sources are divided into four major areas: Physical Threats, System Threats, Data Threats and Direct Security Threats. Physical threats are risks resulting from physical attacks to the computer itself, such as from the environment, fire or sabotage. System threats are threats which seek to destroy the computer operational integrity and these include operation system software flaws and hardware flaws. Data threats are attacks directly upon data and data protection facilities. Finally, direct security threats are those which include techniques of subverting protection such as use of covert communication channels. These threats include impersonation, tapping, and browsing, for example.

The following sections discuss these four areas of threats in mere detail. The discussions give extra emphasis for those threats for which development of a Secure DBMS as described in this report reduces or removes the threat risk.

B.2 Physical Threats

Physical threats to a computer system cause physical damage to the computer or physical impairment of proper functioning of the computer facility. Figure B-1 presents some of the major physical threats. Fire, explosion and sabotage are obvious threats. The environmental threat of power loss can force a system to "dead start", losing all processing active at the time of the outage. A power surge can cause circuit protection devices such as breakers, to trip, resulting in a temporary loss of computing ability and a loss of data (similar to that of a power outage). However, a power surge will generally result in a shorter outage and has a lower probability of causing any outage whatsoever.

Computers and other delicate electronic equipment are sensitive to the effects of temperature and humidity. Thermal problems can cause damage from merely an erroneous bit in memory to catastrophic destruction of the computer. Humidity is a more subtle threat in that it generally requires more time for humidity to damage a computer system. Corrosion of electrical contacts and the resulting subsystem failure is the risk of improper humidity control.

Water is an obvious threat to electrical systems and can cause a variety of system functions to fail. Naturally, direct application of water to an electrical system will generate an electrical short and failure. Water damage to disc and tape media, programs retained on paper tape or cards, or corrosion of the mechanical system such as motor bearings, and disc head activators are examples of water damage.

Particulates cause poor contact between electrical connectors and fatigue of moving parts. Additionally, particulates generate excessive wear to tape heads, disc heads, magnetic tape, and disc media.

Although a failure of a communication system will not "crash" a system totally, it will cause a loss of data and control to users at remote sites for that data of a temporary nature and not permanently retained on the system (i.e. on disc or tape). Finally,

- o FIRE
- o EXPLOSION
- o ENVIRONMENT
  - POWER LOSS/SURGES
  - TEMPERATURE/HUMIDITY EFFECTS
  - WATER
  - PARTICULATES: SMOKE, DUST, DIRT
- o COMMUNICATION SYSTEMS FAILURE
- o SABOTAGE
- o THEFT

Figure B-1 Physical Threats

- o SOFTWARE TRAP DOORS
- o HARDWARE FLAWS
- o INFORMATION HANDLING PRACTICES
- o OPERATING SYSTEM UPDATE PROTECTIONS

Figure B-2 System Threats

theft of hardware or software can disable a computer system if the stolen elements are essential to system operation (i.e. the operating system disc packs).

There are many means to prevent physical damage to a computer system. Development of Secure DBMS is not among them. Physical protection generally encompasses such precautions as perimeter fencing, power filtering, power back-up, systems, and air conditioning (for temperature, humidity, and particulate control).

B.3 System Threats

System threats are somewhat less obvious than physical threats. System threats are methods of attacks which reduce the security integrity of the computer system. Figure B-2 lists some of these threats.

Software and hardware threats are similar. Software trapdoors are sections of code inserted into an operating system or other security process which are normally inactive. However, when triggered by the data thief, these trapdoor processes enable the intruder to acquire system control or access to normally protected system elements or data. Hardware flaws can permit a thief to acquire system control similar to means employed for software trapdoors. Hardware flaws can be intentionally created (like software trapdoors) by the data thief, or they may be created through incomplete system design or incomplete verification of all possible machine states. Information handling practices refers to those means customarily employed by the systems operation personnel and management to safeguard the computer equipment, system software, and protected data. Operating system updates present a particularly vulnerable period for an operating system. Illegal access to the update tapes can permit unauthorized alterations which can potentionally violate security.

Prevention of system threats is generally accomplished through proper system software generation and modification control, development and certification of system main-frames for a secure operating environment, use of proper security containers for system

software storage and proper disposal of all system dumps.

Use of a Secure DBMS can partially reduce some of the above system threats. The distributed architecture developed as a result of this study permits the isolation of the user work space from the security enforcing processors ACS and HSF and switches. This physical isolation increases the difficulty of use of trap doors and hardware flaws. Use of firmware (software encoded into read-only memories - ROM's) for data handling and security enforcement subsystems concurrent with software development by trusted programmers with the proper security clearances (i.e., individuals who will provide software of high integrity) is a must for any secure system. Additionally, use of static encryption for data and programs residing in mass storage strengthens information handling security.

The above security practices are therefore to be a part of the Secure DBMS, namely:

(i) all security related protection enforcement equipment shall provide provable enforcement of simple security and *-security principles.

(ii) all security related software shall be developed in a secure environment by properly cleared and trusted programmers. It shall be certified, as deemed necessary.

(iii) all system updates and modifications shall be created and added to the system in a similar secure environment

(iv) all system software shall be unalterable through storage in read-only memories.

B.4 Data Threats

Data threats are means for potential security violations through which data, application programs or other protected items are compromised. Data threats differ from the system threats above in that the latter attempts to defeat the security protection while the former is generally a direct attack upon the data. Figure B-3 presents these data threats.

Impersonation is a computer attack whereby a user acquires data by masquerading as a legitimate user or device. In one such technique, the data thief acquires valid user's log-on codes and passwords. In another instance, the thief exploits system flaws to bypass proper data input/output access controls. The Secure DBMS developed in this study reduces this threat by physically isolating the user from the processor which contains the log-on authenticator. In addition, the special authentication processes, such as encrypted log-ons may be used to enhance security.

Foible is the system access through unintentional means, through flaws in either hardware or software. Incomplete parameter checking on data requests or other input/output operations can result in a security compromise. The only solution to this threat is dilligence in the design of the data handling hardware and software. Development of these entities in a secure environment does not guarantee a flaw-free system but will prevent deliberate trap doors, as discussed in Section B.4.

Artifice is the introduction of either clandestine code or hardware changes which permit unauthorized data and program access.

As such, artifice is the same as trapdoors. When activated, the illegal user bypasses all system controls on data access. The trapdoor code can be supplied during the creation of the operating system or later through a gift program supplied at a later date (Trojan Horse Attack). A secure DBMS cannot validate software integrity. However, again, the isolation of the user work station from the data management software and the maintenance of the latter in firmware prevents the use of user software to trigger trapdoors. Creation of data processor and secure DBMS protection software and hardware in a trusted (i.e. in a secure environment and by personnel of high integrity)environments and certifying necessary code can alleviate any intentional trapdoor threats.

Another data base threat is that of browsing. Browsing is the searching of systems residues for unauthorized information. Browsing can be as simple as searching wastebaskets for sensitive information or passwords, dumping core of security protection utilities, dumping system core, or searching input/output (I/O) buffers for residues belonging to other users. The distributed architecture of the secure DBMS described in this study can eliminate or greatly reduce browsing. The currently feasible design wherein a separate DBMS processor is used at each security level restricts browsing to within the browser's own clearance level. Browsing in this manner constitues only a need-to-know violation at worst. With the advent of relatively inexpensive DBMS processors (See Technology discussions of Section 7) it may be possible to provide a completely independent DBMS processor microcomputer for each individual. Under these conditions, browsing can be eliminated.

Wire tapping is a significant threat. Tapping can be either active or passive. Active tapping consists of attachment of an unauthorized device, such as a computer terminal, to a communications circuit for the purpose of obtaining access to data by generation of false messages or control information or by altering the communications of authorized users. Passive tapping consists of monitoring and/or recording of data while it is being transmitted over a communications link. It is assumed that all data and query links are contained within a protected facility at the proper security level or that any line not provided suitable physical security protection is properly encrypted using dynamic encryption techniques (See Section 4.2)

The secure DBMS developed through this study cannot prevent tapping. We recommend the use of encryption on all non-secured lines between user sites and the central data base unit to prevent theft of data on data links. Encryption acts to reduce database transactions on communication lines to an unintelligible state. Any changes or additions to messages in an active tape will be made unintelligible to the processing equipment when the message is decoded. Proper installation of the data base, protection equipment, and physical protection of the site will preclude direct taps to the data base equipment.

The final data base threat is one common to all electronic equipment: radiation. All electronic equipment, and especially devices with switching currents, generate electromagnetic radiaton. In a computer, the higher the processor speed, the higher the radiated frequency and the greater the harmonic content of the RF signature will be. Intelligent interception of these emanations can provide illegal access to sensitive data. In addition to any electromagnetic signal, electromechanical devices also radiate signatures in the form of acoustical emanations. Alleviation of the risk incurred from the threat of radiation must be designed into system hardware. In addition, screening the room for RF signals, properly filtering all power and communication lines, and proper use of acoustical absorptive materials can reduce the risk of security loss.

- IMPERSONATION
  - ACQUIRE INFORMATION
  - ACQUIRE ACCESS
- FOIBLE
  - ACCIDENTAL SYSTEM ACCESS
- ARTIFICE
  - TRAP DOOR CREATION
  - HARDWARE MODIFICATIONS
- BROWSING
  - CORE DUMPS
  - I/O BUFFER EXAMINATION
- TAPPING
  - ACTIVE: between-the-lines; piggy-back entry
  - PASSIVE: eavesdropping
- RADIATION
  - ELECTROMAGNETIC
  - ACOUSTICAL

Figure B-3 Data Threats

- COVERT CHANNELS
- TRUSTED SUBJECTS, PROCESSES
- ILLEGAL RE-CLASSIFICATION OF OBJECTS
- ILLEGAL ACCESS TO OBJECTS

Figure B-4 Direct Security Threats

B.5 Security Threats

In addition to the direct threats indicated above, there are threats to data security. These threats normally involve at least one validated system user. Figure B-4 describes typical threats of this type.

Principle among threats of this type is that of a covert communication channel through which a system user is able to send information to an illegal accomplice or is able to move sensitive data to a position in the data base system where it can be illegally observed (i.e., intentional divulgence of sensitive information). Covert channels may exist in several forms in a data base system. A covert channel becomes a threat when a valid subject uses it to directly transmit information to another person or to bypass the restrictions of the *-protection property. In the data base design, we must use every means possible to prevent the creation of potential covert channels.

There are probably many unprotectable covert channels in any system. Any multi-level resource or system control register (such as write reservations) constitutes a potential covert channel. The goal of the secure DBMS is to minimize the utility of any covert channel by lowering the effective data rate to an unacceptably low level. Section 2.4 describes some techniques for reducing the effectiveness of timing channels. In addition, security monitoring and audit trails are useful for detecting attempts at unusual system activity possibly associated with a covert channel. Section 2.6 discusses the value of security monitors in detection of possible covert communicaitons.

Trusted subjects and processes are required in order to provide the capability of downgrading classified information, such as in the case of "sanitized" summaries which are often desired from sensitive material. Special means within the secure DBMS can be taken to ensure that only designated equipment can be used for such downgrading operations. In addition, the security officer can be alerted to each write from such a process which represents a downgrade operation and thereby screen this process in real time, if desired. Nonetheless, trusted system entities remain one of the greatest security risks. Section 2.5 discusses the

trusted processes and personnel in more detail.

The final two examples of direct security threats, illegal re-classification and illegal access to objects, are well addressed by the secure DBMS. The access control subsystem and Hardware Security Filter strictly enforce the tranquility principle by denying re-classification of objects, and strictly enforce the simple security rule and *-security property to prevent illegal object accesses.

APPENDIX C

Processing Requirements for Conversion of a PN Sequence Index to a PN State

This appendix supports Section 4.4.1.5 by describing the mathematical process by which a PN index can be converted to a PN state. The object of this exercise is to establish the processing requirements of the process for use in a static encryption mechanism for user, security level and compartment, and data storage isolation.

To support high speed, random access, statically encrypted, disk I/O, there is a need for rapid conversion of random access disk location (i.e., disk number, track number, sector number) (PN sequence index) into the associated LFSR PN state This involves first converting the disk drive number, track number, and sector number into a logical data storage bit address which is used as the PN sequence index. Then this PN sequence index is converted to the associated PN state of an LFSR. The first operation is very simple, requiring three integer multiplies and two integer additions:

LFSR Index = Disk Drive Number x Conversion Factor 1 +
Track Address x Conversion Factor 2 +
Sector Address x Conversion Factor 3

Conversion of the LSFR index to a PN state is considerably more complicated. This process is based on the representation in a Galois Field as shown with a root α^n, as:

$$P(X) = X^n + A_{n-1}X^{n-1} + \ldots + A_1 X + 1 \text{ where } A_i = 0 \text{ or } 1$$
$$\text{with root } \alpha^n = A_{n-1}\alpha^{n-1} + \ldots + A_1 \alpha + 1 \qquad \text{(Eq C-2)}$$

From this polynomial, two tables can be constructed. Table 1 is the representation of α^i, i = 0 to 2n-1 as a linear sum (exclusive or) or α^k, k=0, n-1. That is the i^{th} entry of Table 1 is a vector of coefficients a_{ik} such that

$$\alpha^i = \sum_{k=0}^{n-1} a_{ik}\ \alpha^k. \qquad \text{(Eq C-3)}$$

Table 2 is the representation of $(\alpha^n)^{2^j}$, j = 0 to n-1 as a linear sum of α^k, k = 0, n-1. Therefore, the j^{th} entry of Table 2 is a vector of coefficients b_{jk} such that

$$(\alpha^n)^{2^j} = \sum_{k=0}^{n-1} b_{jk}$$ are either 0 or 1. Therefore, these two sets

can be represented by linear sums of the first n states of the PN sequence. As these tables can be pre-computed then stored, they represent only a memory requirement of 3n by n bits.

To determine the m^{th} state, m is divided by n and the remainder kept. This allows α^m to be factored as

$$\alpha^m = \alpha^r\ \alpha^{qn} \text{ where } q = \sum_{i=0}^{n-1} C_i 2^n \text{ and } C_1 = 0 \text{ or } 1 \qquad \text{(Eq C-5)}$$

Eq C-5 can be further factored into quantities stored in Tables 1 and 2 as

$$\alpha^m = \alpha^r\ \alpha^{\sum_{i=0}^{n-1} C_i Z^n} = \alpha^r \prod_{c_i=1} (\alpha^n)^{2^i} \qquad \text{(Eq C-6)}$$

These quantities are represented as polynomials of degree n-1, hence each multiplication results in a polynominal of degree 2n-2. Before proceeding to the next product, all terms of degree n to 2n-2 are themselves reduced to polynominals of degree n-1, i.e. after each multiplication of polynominals, the higher order terms are reduced giving a resulting polynominal of degree n-1. The implementation of this reduction involves exclusive OR-ing of the lower order terms with a table 1 entry for each higher order term. The end result is the representation of the m^{th} state as a sum (exclusive OR) of the first n states of the LFSR sequence. These n nbits states can be stored in a third table.

In summary, the requirements for conversion of LFSR sequence index to LFSR state is 4n by n bits of memory, 1 integer divide with remainder, up to n Table 2 look ups, up to n nbit by nbit integer multiplies, up to n · (n-1) table 1 look ups and exclusive ORs, and up to n Table 3 look ups and exclusive ORs. Worst case and average computation required for conversion are summarized in Figure C-1.

Category	Worst Case	Average
Memory	4n by n bits	4n by n bits
Integer Multiplies	n - n bit by n bit	n/2 - n bit by n bit
Integer Divides	1 - n bit by n	1 - n bit by n
Exclusive Ors	n^2 - n bit XORs	$n^2/4$ - n bit XORs

Figure C-1 LFSR Index to State Conversion Requirements

APPENDIX D

PERFORMANCE ANALYSIS DETAILS

D.1 Introduction

This appendix describes the analysis performed to estimate system performances. Additional detail, particularly in the areas of queuing theory, models, and analysis, is provided to supplement Section 6.2.

This section consists of the following discussions:

D.2 System Model

D.3 Detailed Traffic Models

D.4 M/M/m Queue Performance

D.2 System Model

Analysis of system performance requires a system model, a performance metric and a traffic model. These first two items can be handled relatively vigorously and in detail if desired; however, the traffic model is, to a large degree, arbitrary and is best thought of as a benchmark.

The DBMS system model shown in Figure D-1 can be analyzed as a queueing network. To a first approximation it can be modeled as a network of M/M/m* queues[9,10] with an infinite population. This model has numerous advantages versus more detailed models. The three most important advantages of this approximation are:

*Notation M/M/m queues: In general, queues are described in terms of three parameters, P1/P2/P3

P1 = distribution type of interarrival time
P2 = distribution of type of service time
P3 = number of servers

Common distribution types used in queueing theory are:

M - Markov
G - General
D - Deterministic

Thus, an M/D/m queue is described as one with Markovian interarrival time, Deterministic service time, and m servers.

- o The analysis of a M/M/m queuing network can be decomposed into the analysis of individual M/M/m queues.
- o Individual M/M/m queues can be analytically analyzed. This model will provide allower bound on system performance.
- o Furthermore, a more detailed system model would be much more heavily tied to system details which are usually difficult or impossible to quantify, such as, requires considerable more effort to analyze, and is incapable of improving the performance prediction as this is usually dominated by the traffic model sensitivity.

Thus modeling this system requires quantifying m, the number of servers, and $\frac{1}{\mu}$, the average service time, for each subsystem.

The proposed system metric is multi-dimensional as shown in Figure D-2 with the average system response time determined from this metric through Equation D-1. This metric consists of the average service time, $\frac{1}{\mu}$, capacity, i.e., the maximum number of users the sytem can support, and the average normalized response time as a function of the normalized system load, τ_n. Since the system model can be decomposed, Equation 6-1 can be rewritten in terms of subsystem metrics as shown in Equation D-2. A more optimistic performance prediction can be made by relaxing the requirement that these subsystem metrics come from a M/M/m queue model, resulting in approximation of Equation D-3.

As previously stated a traffic model is needed for performance analysis but cannot be rigorously determined and hence is somewhat arbitrary. As a portion of the system/subsystem metric is capacity in terms of number of users, a user traffic model is required. Furthermore the system model chosen assumes only one type of "user", and hence this "user" model should be the most likely type. For our purposes this will be defined to be an on-line user making a query by key that results in a single response every 20 seconds. To increase the load, it will be assumed that the query involves two relations with the first relation containing the foreign key for the second relation. Thus, the network traffic flow is deterministic.

Details of the traffic model are present in the following section (Figure D-3). The process breakdown of the traffic model permits viewing the model consisting of a finite number of users, K, with an individual query rate, λ, as an infinite number of users with a combined query rate of $k\lambda$ queries per second.

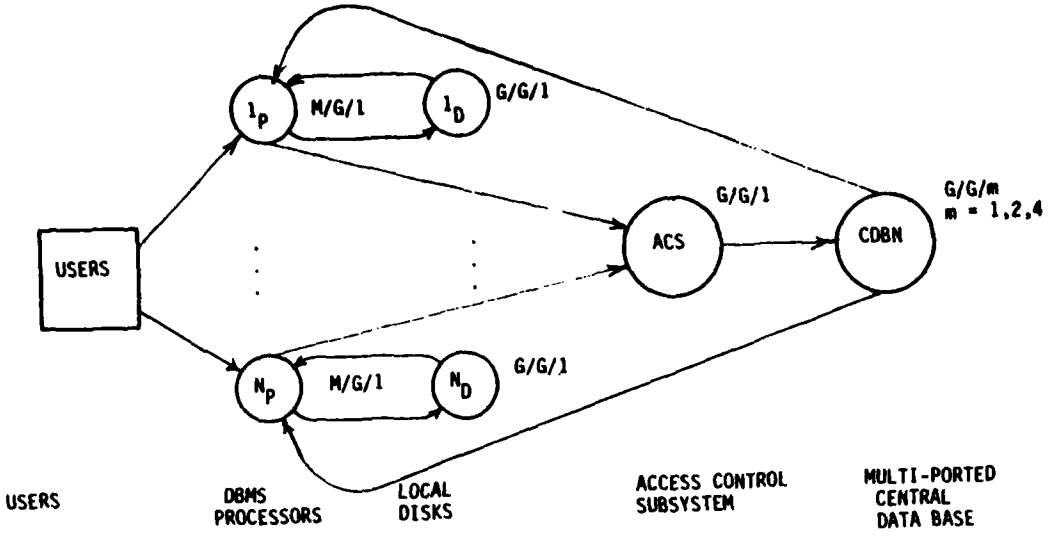


Figure D-1 2N+2 Node Queuing Network System Model

System Metric $(1/\mu, C, \tau_n(\rho))$
where $1/\mu = \overline{X}$ = average system time
C = Capacity = Maximum Number of Users

$\mathcal{T}^n(\rho) = \mu\mathcal{T}(\rho)$ = Average Normalized System Response Time

$$\mathcal{T}_s \leq (1/\mu)\mathcal{T}^n\left(\frac{\#\ users}{C}\right) \qquad \text{EQ D-1}$$

$$\mathcal{T}_s \leq \sum_{k=o}^{m} (1/\mu_{i_k})\mathcal{T}^n_{M/M/m}\left(\frac{\#\ users}{C}\right) \qquad \text{EQ D-2}$$

$$\mathcal{T}_s \simeq \sum_{k=o}^{m} (1/\mu_{i_k})\mathcal{T}^{n/i_k}\left(\frac{\#\ users}{C}\right) \qquad \text{EQ D-3}$$

Figure D-2 System/Subsystem Metrics and Average System Response Performance

Each query involves a fixed sequence of processor tasks and disk accesses. It is assumed that each type of processor task has the same avearge service time and that service time has a negative exponential probability density function (PDF). The same assumption is made for disk accesses, although the mean processor service time and mean disk access time can be different for each processor and disk system.

D.3 Detailed Traffic Models

Based on the simple benchmark traffic model previously defined, a detailed network traffic model is derived for both virtual memory and non-virtual memory systems. A benchmark traffic model of an on-line user making a query of a two relation view with the query containing the key to the first relation and the associated record containing the foreign key to the second record, resulting in a single response was chosen. This benchmark must be converted into a physical network traffic model, i.e., processor tasks and disk accesses. Figure D-3 details the processes of the model as a series of requested events consisting of 5 processor tasks and 6 disk accesses. This model ignores I/O and OS overhead and assumes multiprocessing versus true time sharing operation. Note that disk access 6 is not part of the system response timeline and is important only in that it increased disk loading and thus only indirectly increases system response time.

Based on this analysis, the $K\lambda$ queries/second system load can be converted into subsystem interarrival rates. For a system consisting of N DBMS processors, this results in a load of $\frac{5K\lambda}{N}$ processor tasks/seconds. If a VM system is used then there is a local disk load of $\frac{6K\lambda}{N}$ data base access/second. For a VM system it is also important to know how many "new relations" per second are required. This can be expressed as 1 new relation per j queries or a load of $\frac{K\lambda}{j}$ on the central data base. The size of these central disk accesses will be much larger than the local paging disk accesses. For our VM benchmark/analysis parameter values of λ = 0.05hz (avg. query per user every 20 sec) and j=30 (new relation used every 30 queries) are chosen.

For the non VM (NVM) benchmark it is assumed that disk accesses 2 and 4 are not required, and hence there are $\frac{3K\lambda}{N}$ processor tasks/second and $4K\lambda$ central disk accesses/second. While we are optimistically assuming that fewer disk accesses are required, we will assume the same amount of processing is required,

Event	Task #	Disk #	Description
1		Disk Access 1	Upon arrival of a user query, the user is rolled into the DBMS processor.
2	Processor Task 1		The query is processed and the key to the first relation hash is coede to determine the pointer table index.
3		Disk Access 2	The partially inverted pointer table is read from disk (if not already resident).
4	Processor Task 2		The pointer is used to compute the desk the disk address of the record.
5		Disk Access 3	The record is read.
6	Processor Task 3		The foreign key is removed from the record and is hash coded to determine the pointer table index.
7		Disk Access 4	This pointer table is read in from disk (if not already resident).
8	Processor Task 4		The pointer is used to compute the disk address of this record.
9		Disk Access 5	The second record is read.
10	Processor Task 5		The view is constructed and the query response is formatted.
11		Disk Access 6	The user is rolled out.

Figure D-3 Detailed Data Access Process Model

i.e., $\frac{3K\lambda}{N}\frac{1}{\mu_{nvm}} = \frac{5K\lambda}{N}\frac{1}{\mu_{vm}}$ or $\mu_{nvm} = 0.6\mu_{vm}$. If we had made comparable traffic assumptions the disk load would be 50% higher. For our NVM benchmark/analysis, $\lambda = 0.5$hz was also used.

In addition to these arrival rates, average service times for these subsystems are required. This is both a function of the load and subsystem rate. For the average disk 2.5 revolutions at 16.6 ms/rev is a reasonable performance estimate, resulting in an average access time, μ_D, of 41.5ms. DBMS processor task service time can vary widely. For this benchmark we will assume $1/\mu_{nvm} = \frac{1}{2}\mu_D$, i.e that these DBMS systems are I/O bound, and that the average access control node service time, $1/\mu_{ACN}$, equals $\frac{1}{8}(1/\mu_D)$, i.e., the access control node is very fast relative to disk performance.

The above traffic model parameters for the system model are summarized in Figure D-4.

D.4 M/M/m Queue Performance

The average normalized system response time for an M/M/m queue with infinite population can be analytically solved in terms of its utilization, i.e., percent of capacity loading. This in turn allows a maximum subsystem degradation limit to be converted into a subsystem loading limit. For our analysis of subsystem response time degradation, a 3dB limit will be used for normal operation and a 6dB limit will be used as a heavy load model.

The average normalized system response time for an M/M/m queue with infinite population is given in Figure D-5. This function is plotted for a subsystem utilization (ρ) range of 0 to 1 for various choices of m (number of servers) in Figures D-6 and 7. It should be noted that the capacity of an M/M/m queue, C_m, is m x C_1. If capacity were held constant then average response time vs ρ would be as shown in Figure D-8. For our analysis we are interested in capacity for 3dB (τ_n=2) and 6dB (τ_n=4) subsystem degradation for M/M/1, M/M/2, and M/M/4. These values are given in Figure D-9.

In addition, it may be argued some service times are closer to deterministic than exponential, e.g. constant processor service time and/or single

MODEL PARAMETER	SYMBOL	UNITS	VM SYSTEM	NON-VM SYSTEM
Query Rate	λ	Queries/Sec/User	.05	.05
New Relation Rate	$\dot{j}$	Relations/Query	1/30	-
Mean Disk Service Time	$1/\mu_D$	MS	40	40
Mean DBMS Processor Service Time	$1/\mu$	MS	12	20
Mean Access Control Node Service Time	$1/\mu$	MS	5	5
DBMS Processor Load	λ_P	Queries/Sec/Processor	0.25 K/N	0.15 K/N
Local Disk Load	λ_L	Queries/Sec/Processor	0.3 K/N	-
Central Data Band and Access Control Node Load	λ_C	Queries/Sec/Processor	0.01 K	0.2 K

K = Number of Users
N = Number of DBMS Processors

Figure D-4 Summary of Traffic Model Parameters

record accesses with disk optimizer. Thus the M/D/m queue is also of interest. The average normalized response for the M/G/1 and M/D/1 is given in Figure D-10.

$$\tau_n = \frac{\mu c \tau}{m} = 1 + \frac{\rho_m}{m\,(1-\rho)}$$

Where $\rho m = \dfrac{\rho_o\,(m\rho)^m}{(1-\rho)m!}$

And $\rho_o = \dfrac{1}{\sum_{k=o}^{m-1} \dfrac{(m\rho)^k}{K!} + \dfrac{(m\rho)^m}{(1-\rho)m!}}$

Figure D-5 Average Normalized Response Time vs Utilization

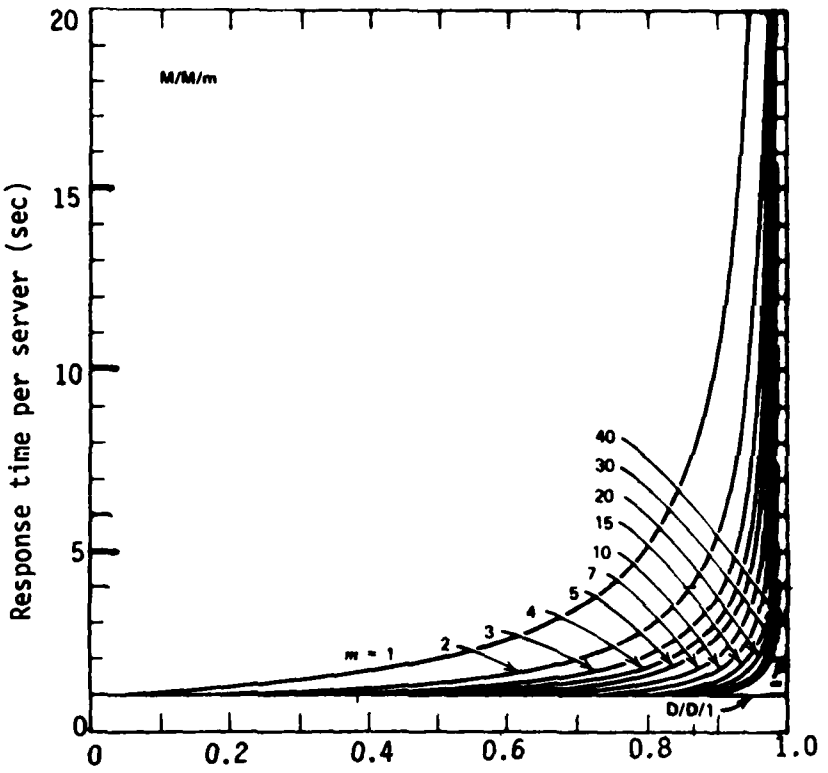


Figure D-6

Normalized Average Response Time

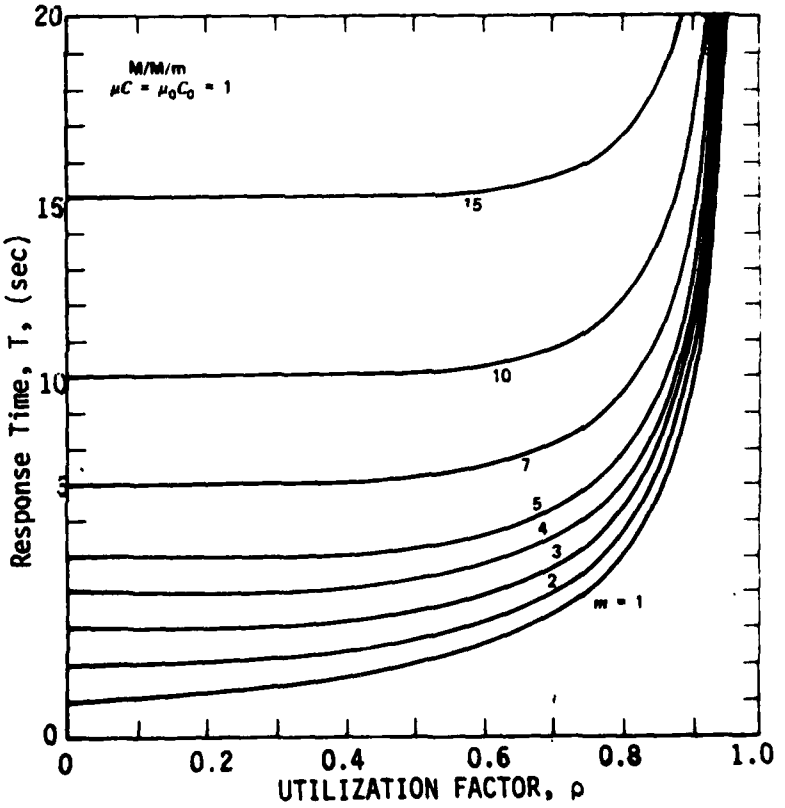


Figure D-7

Average Response Time With Fixed Capacity

QUEUE SYSTEM	PARAMETER	AVERAGE SYSTEM RESPONSE DEGRADATION	
		3 dB	6 dB
M/M/1	ρ	0.5	0.75
	λ_η	0.5	0.75
M/D/1	ρ	0.67	0.86
	λ_η	0.67	0.86
M/M/2	ρ	0.71	0.865
	λ_η	1.42	1.73
M/M/4	ρ	0.84	0.93
	λ_η	3.36	3.72

ρ = Utilization

λ_η = Normalized # of users supported by system assuming CM = MC_1

Figure D-8 Summary of System Utilization for 3 dB and 6 dB System Response Degradation

$$T^n_{M/G/1}(\rho) = 1 + \rho + \frac{\lambda^2 E\{\rho^2\}}{2(1-\rho)}$$

$$T^n_{M/D/1}(\rho) = 1 + \rho + \frac{\lambda^2 (1/\mu)^2}{2(1-\rho)} = \frac{1 - \rho^2/2}{(1-\rho)}$$

Figure D-9 Normalize System Time for M/G/1 and M/D/1 Queues (from Little's Result and Pohlaczek - Khinchin Mean Value Formula)

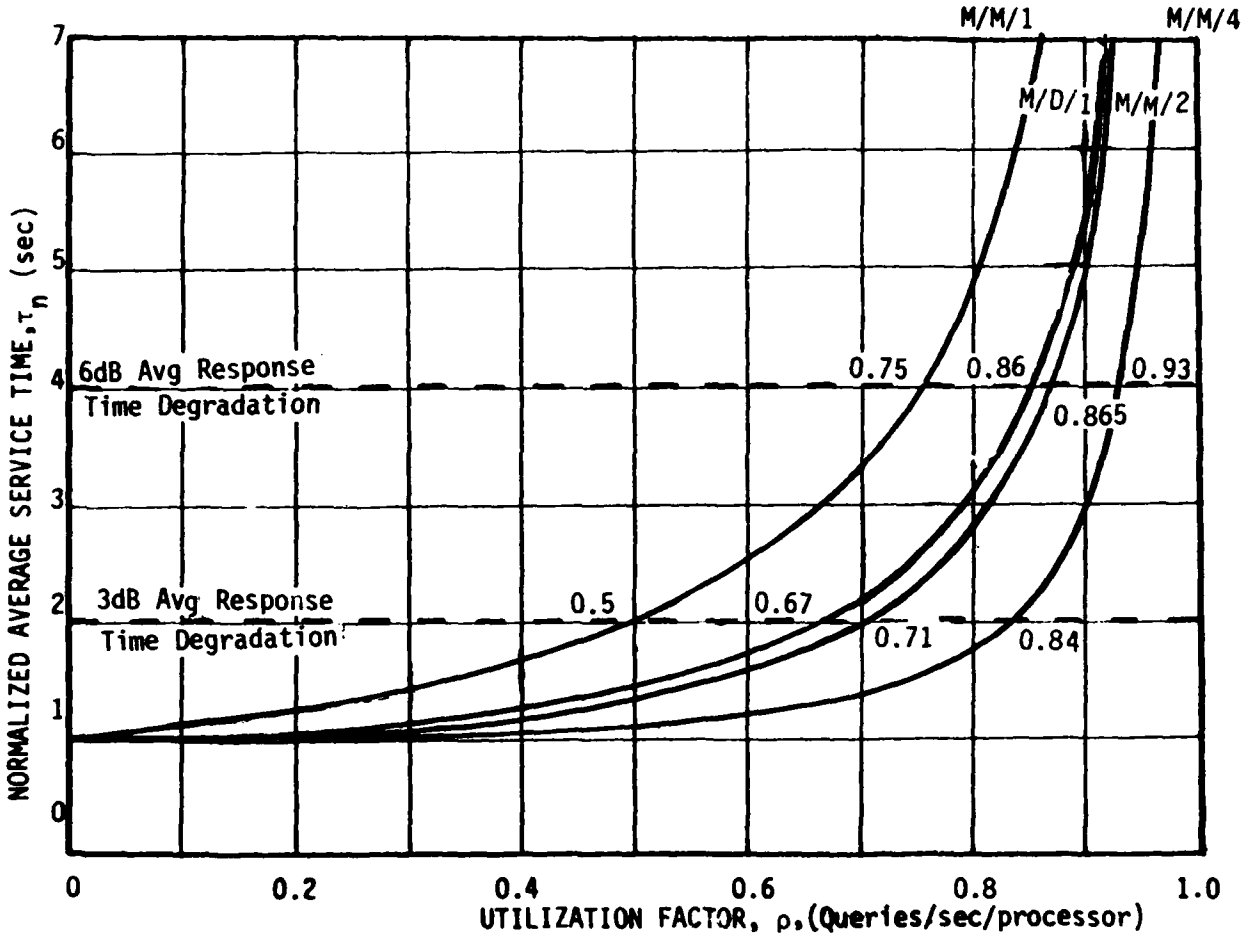


Figure D-10 Average Normalized System Response Time

APPENDIX E

REFERENCES

1. Bell, D.E., and LaPadula, L.J., Secure Computer System: Unified Exposition and Multics Interpretation, MITRE Corp., RPT. ESD-TR-75-306, Jan 1976.

2. Grohn, M.J., A Model of a Protected Data Management System, I.P. Sharp Associates, Ltd., Rpt ESD-TR-76-289, June 1976.

3. Martin, J., Principles of Data-Base Management, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976.

4. Data, C.J., An Introduction to Data Base Systems, Addison-Wesley Publishing Co., Inc., Reading, Mass., 1977.

5. Atre, S., Data Base: Standard Techniques for Design, Performance, and Management, John Wiley and Sons, Inc., NY, 1980.

6. Bell, D.E., Secure Computer Systems: A Refinement of the Mathematical Model", MITRE Corp., Final Tech. Rpt ESD-TR-73-278, Nov 1975.

7. Hinke, T.H. and Schaefer, M., Secure Data Management System, Systems Development Corporation, Final Tech. Rpt., RADC-TR-75-266, Nov 1975, AD#A019201.

8. Robinson, L., The HDM Handbook, SRI International, Project 4828, Contract N00123-76-C-0195, June 1979.

9. Kleinrock, L., Queueing Systems, Vol I., Wiley and Sons, NY, 1976.

10. Jaiswal, N.K., Priority Queues, Academic Press, NY, 1968.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


F